



Руководство по эксплуатации

версия 1.1.1

ООО «Веб-Сервер»

сент. 24, 2026

Оглавление

1	Аннотация	1
2	Общие сведения	2
3	Первоначальная настройка	3
3.1	Общая настройка	3
3.1.1	Базовая настройка системы	3
3.1.2	Расширенная настройка	4
3.1.3	Мониторинг	4
3.1.4	Логирование	4
3.1.5	Примеры типовых схем развертывания Angie ADC	4
3.2	Настройка в мастере	5
3.2.1	Меню мастера	6
3.2.2	Настройка сетевых интерфейсов	6
3.2.3	Настройка интерфейса управления	7
3.2.4	Создание бондов	7
3.2.5	Создание VLAN-интерфейсов	8
3.2.6	Просмотр конфигурации Angie ADC и сетевых настроек	8
3.2.7	Смена портов для веб-интерфейса Angie ADC	9
3.2.8	Изменение имени устройства Angie ADC	9
3.2.9	Проверка доступности узла	9
3.3	Настройка сетевых интерфейсов	9
3.3.1	Изменение параметров сетевого интерфейса	9
3.3.2	Просмотр параметров сетевого интерфейса	10
3.4	Подключение внешних пользователей	10
3.4.1	Подключение внешних пользователей через LDAP	10
3.4.1.1	Шаги	11
3.4.2	Подключение внешних пользователей через TACACS	13
3.4.2.1	Шаги	14
3.4.3	Подключение внешних пользователей через RADIUS	15
3.4.3.1	Шаги	16
4	Балансировщик нагрузки	18
4.1	Настройка конфигурации	19
4.1.1	Общая информация	19
4.1.2	Перезапуск балансировщика	20
4.1.3	Логи балансировщика	21
4.1.4	Принцип конфигурации	21
4.1.5	Использование портов	21
4.1.6	Настройка простого прокси-сервера	22
4.1.7	Настройка балансировки нагрузки	23

4.1.8	Раширенная настройка	23
4.2	Управление резервными конфигурациями	24
4.2.1	Просмотр списка сохраненных конфигураций	24
4.2.2	Изменение рабочего статуса хранимой конфигурации	24
4.2.3	Восстановление последней рабочей версии конфигурации	24
4.2.4	Восстановление конфигурации из резервной копии	24
4.2.5	Выбор репозитория для хранения конфигураций	25
4.3	Методы балансировки	26
4.3.1	Примеры HTTP-балансировки нагрузки	27
4.3.2	Примеры TCP/UDP-балансировки нагрузки	31
4.4	Предоставление доступов на работу с конфигурацией	34
4.4.1	Предоставление прав на просмотр и редактирование	34
4.4.2	Просмотр выданных доступов	35
4.4.3	Редактирование выданных доступов	35
4.4.4	Удаление выданных доступов	35
4.5	Проверки работоспособности серверов	35
4.5.1	Пассивные проверки	36
4.5.2	Активные проверки	36
4.5.2.1	Пример для HTTP	36
4.5.2.2	Пример для TCP/UDP	37
4.5.2.3	Пример ICMP echo-запроса для HTTP	38
4.6	Резервирование проксируемых серверов и групп	38
4.6.1	Запасные серверы (spare-серверы)	39
4.6.2	Резервные группы серверов (backup-группы)	39
4.7	Загрузка и раздача статических файлов	41
4.7.1	Раздача файлов из разных локальных каталогов	41
4.8	Аутентификация клиентов	42
4.8.1	Настройка аутентификации клиентов	43
4.8.2	Изменение параметров портала аутентификации	44
4.8.3	Удаление портала аутентификации	45
4.9	Делегирование сервиса оператору	45
4.9.1	Настройка со стороны администратора	45
4.9.2	Пример работы оператора	46
4.10	Справочная информация	47
4.10.1	Общие сведения	47
4.10.2	Основной модуль	47
4.10.3	HTTP-модули	47
4.10.4	Потоковые модули	47
4.10.5	Почтовые модули	48
4.10.6	Сторонние модули	48
4.10.7	Инструкции	48
4.10.8	Указатель директив	48
4.10.8.1	Конфигурационные файлы	48
4.10.8.2	Соединения, сессии, запросы, логи	50
4.10.8.3	Основной модуль	59
4.10.8.4	HTTP-модули	69
4.10.8.5	Потоковые модули	420
4.10.8.6	Почтовые модули	520
4.10.8.7	Сторонние модули	543
4.10.8.8	Настройка ACME	606
4.10.8.9	Настройка пользовательских метрик	623
4.10.8.10	Режим отладки	624
5	Глобальная балансировка	627
5.1	Настройка сервиса GSLB	628
5.2	Настройка конфигурации	629
5.2.1	Добавление домена и создание записей	629

6	IP-маршрутизация	634
6.1	BGP-маршрутизация	634
6.1.1	Резервирование с помощью протокола BGP (Active-Standby)	635
6.1.1.1	Введение	635
6.1.1.2	BGP-сессии	636
6.1.1.3	Обработка типовых сбоев	636
6.1.1.4	Настройка маршрутизации	637
6.1.1.5	Проверка работы	640
6.1.1.6	Применение с IP transparency	643
6.1.2	Распределение нагрузки с помощью протокола BGP (Active-Active)	643
6.1.2.1	Введение	643
6.1.2.2	BGP-сессии	644
6.1.2.3	Обработка типовых сбоев	645
6.1.2.4	Настройка маршрутизации	646
6.1.2.5	Топология #1	646
6.1.2.6	Топология #2	650
6.1.2.7	Топология #3	653
6.1.2.8	Применение с IP transparency	655
6.1.2.9	Заключение	655
6.2	OSPF-маршрутизация	655
6.2.1	Резервирование с помощью протокола OSPF (Active-Standby)	655
6.2.1.1	Введение	655
6.2.1.2	Обработка типовых сбоев	657
6.2.1.3	Настройки маршрутизации	658
6.2.1.4	Заключение	663
6.2.2	Распределение нагрузки с помощью протокола OSPF (Active-Active)	663
6.2.2.1	Введение	663
6.2.2.2	Обработка типовых сбоев	664
6.2.2.3	Настройки маршрутизации	666
6.2.2.4	Проверка работы	668
6.2.2.5	Применение с IP transparency	670
6.3	VRRP-маршрутизация	670
6.3.1	Предварительная настройка для VMware ESXi	670
6.3.1.1	Настройка VMware ESXi до версии 6.7	670
6.3.1.2	Настройка VMware ESXi версии 6.7+	671
6.3.2	Режим резервирования (Active-Standby)	672
6.3.2.1	Введение	673
6.3.2.2	Обработка типовых сбоев	674
6.3.2.3	Настройка VRRP	676
6.3.2.4	Проверка работы	677
6.3.2.5	Заключение	678
6.3.3	Режим распределения нагрузки (Active-Active)	679
6.3.3.1	Введение	679
6.3.3.2	Подключение Angie ADC с маршрутизатором	679
6.3.3.3	Подключение Angie ADC без маршрутизатора	682
6.3.3.4	Настройка VRRP	682
6.3.3.5	Проверка работы	683
6.3.3.6	Заключение	684
6.3.4	Команды VRRP	684
6.3.4.1	Команды настройки	686
6.3.4.2	Команды удаления и отмены	687
6.3.4.3	Дефолтные команды для настройки и отмены	688
6.3.4.4	Команды для просмотра информации	688
6.3.5	Создание и удаление VRRP-групп	689
6.3.5.1	Создание VRRP-группы	689
6.3.5.2	Удаление VRRP-группы	689
6.4	Использование протокола BFD для уменьшения времени реакции	690
6.4.1	Введение	690

6.4.2	Настройка с OSPF	690
6.4.3	Настройка с BGP	692
6.4.4	Настройка с VRRP	693
6.4.5	Настройка со статическими маршрутами	693
6.4.6	Настройка независимого пира	694
6.4.7	Профиль	695
6.5	Настройка RHI	695
6.5.1	Настройка BGP	696
6.5.1.1	Пример	696
6.5.1.2	Проверка	696
6.5.1.3	Отзыв префиксов	697
6.5.1.4	Объявление префиксов после восстановления работы апстрима	697
6.5.2	Настройка OSPF	697
6.5.2.1	Пример	697
6.5.2.2	Проверка	698
6.5.2.3	Отзыв префиксов	698
6.5.2.4	Объявление префиксов после восстановления работы апстрима	699
6.5.3	Настройка балансировщика нагрузки	699
6.5.3.1	Пример	699
6.5.4	Настройка конфигурации RHI	700
6.5.4.1	Удаление префиксов	700
7	Высокая доступность	701
7.1	Обзор работы	702
7.1.1	Режим Active / Standby	702
7.1.2	Развертывание	702
7.1.3	Обновление конфигурации и синхронизация	703
7.2	Создание пары высокой доступности	703
7.2.1	Требования	703
7.2.2	Предварительные действия	703
7.2.3	Создание пары высокой доступности в режиме Active / Standby	704
7.3	Настройка высокой доступности в паре	705
7.3.1	VRRP	705
7.3.1.1	Назначение адреса сервиса на виртуальный VRRP-интерфейс	707
7.3.1.2	Назначение адреса сервиса на интерфейс lo	707
7.3.2	OSPF	707
7.3.2.1	Пример	708
7.3.2.2	Проверка	709
7.3.3	BGP	711
7.3.3.1	Пример	711
7.3.3.2	Проверка	712
7.4	Управление парой	714
7.4.1	Просмотр информации о паре	714
7.4.2	Обновление конфигурации и синхронизация	715
7.4.3	Настройка протокола опроса доступности узлов	716
7.4.4	Удаление пары	716
7.5	Привязка клиентских сессий	717
8	Кластеризация	718
8.1	Создание кластера	718
8.1.1	Создание точки входа в кластер	719
8.1.2	Добавление узлов в кластер	720
8.2	Просмотр состояния кластера	721
9	Мониторинг и статистика	723
9.1	Мониторинг балансировщика нагрузки	723
9.1.1	Виджет "Серверные зоны"	724
9.1.2	Виджет "Зоны апстримов"	725
9.2	Мониторинг системы	726

9.2.1	Сводная иконографика	726
9.2.2	Центральный процессор	727
9.2.3	Обработка данных	727
9.2.4	Сетевые интерфейсы	727
9.2.5	ОЗУ	727
9.2.6	Файловая система	728
9.2.7	Накопители (ROM)	728
9.2.8	S.M.A.R.T. атрибуты диска	728
9.2.9	Датчики	729
9.3	Мониторинг и статистика в Console Light	729
9.3.1	Переход в Console Light	729
9.3.2	Интерфейс	730
9.3.3	Вкладка "Angie"	730
9.3.3.1	Виджет "<версия>"	730
9.3.3.2	Виджет "Соединения"	730
9.3.3.3	Виджет "HTTP-зоны"	731
9.3.3.4	Виджет "HTTP-апстримы"	731
9.3.3.5	Виджет "TCP/UDP-зоны"	731
9.3.3.6	Виджет "TCP/UDP-апстримы"	732
9.3.4	Вкладка "HTTP-зоны"	732
9.3.4.1	Раздел "Серверные зоны"	732
9.3.4.2	Раздел "Зоны путей (Location)"	733
9.3.4.3	Раздел "Зоны ограничения соединений (Limit Conn)"	733
9.3.4.4	Раздел "Зоны ограничения запросов (Limit Req)"	734
9.3.5	Вкладка "HTTP-апстримы"	734
9.3.6	Вкладка "TCP/UDP-зоны"	735
9.3.6.1	Раздел "TCP/UDP-зоны"	736
9.3.6.2	Раздел "Зоны ограничения соединений (Limit Conn)"	736
9.3.7	Вкладка "TCP/UDP-апстримы"	737
9.3.8	Вкладка "Кэши"	738
9.3.9	Вкладка "Общие зоны"	739
9.3.10	Вкладка "DNS-резолверы"	740
9.3.11	Виджет "Настройки"	740
9.3.12	Панель управления консолью	741
9.4	Экспорт метрик	741
9.4.1	Экспорт метрик Node Exporter	741
9.4.2	Экспорт метрик балансировщика нагрузок	741
9.4.3	Экспорт системных метрик Angie ADC	742
9.5	SNMP	742
9.5.1	Пользователи SNMP	743
9.5.2	Метрики операционной системы	744
9.5.3	Собственные метрики Angie ADC	744
10	Администрирование и безопасность	746
10.1	Настройка системы	746
10.1.1	Аутентификация	747
10.1.1.1	Типы аутентификации	747
10.1.1.2	Включение внешней аутентификации	747
10.1.1.3	Изменение параметров внешней аутентификации	748
10.1.1.4	Отключение внешней аутентификации	748
10.1.2	Авторизация	749
10.1.2.1	LDAP-авторизация	750
10.1.2.2	TACACS-авторизация	751
10.1.2.3	RADIUS-авторизация	752
10.1.2.4	Выключение внешней авторизации	753
10.1.3	Настройка HTTPS	753
10.1.3.1	Загрузка сертификата управления и включение защищенного соединения	753

10.1.4	Настройка NTP	754
10.1.4.1	Настройка конфигурации NTP	754
10.1.4.2	CLI	755
10.1.5	Настройка конфигурации	755
10.1.5.1	Hostname	755
10.1.5.2	Сетевые интерфейсы	756
10.1.6	Управление пользователями	757
10.1.6.1	Локальные пользователи	757
10.1.6.2	Внешние пользователи	758
10.1.6.3	Добавление локального пользователя	758
10.1.6.4	Изменение учетных данных и статуса локального пользователя	759
10.1.6.5	Сброс пароля локального пользователя	759
10.1.6.6	Сброс настроек двухфакторной аутентификации локального пользователя	760
10.1.6.7	Удаление локального пользователя	760
10.2	Управление сертификатами	760
10.2.1	Загрузка сертификата данных	760
10.3	Журналы событий и экспорт	761
10.3.1	Просмотр журналов событий	761
10.3.2	Экспорт логов	763
10.3.2.1	Создание нового syslog-сервера	763
10.3.2.2	Просмотр настроенных syslog-серверов	763
10.3.2.3	Изменение параметров syslog-сервера	763
10.3.2.4	CLI	764
10.4	Аудит действий пользователей	764
10.5	Диагностика сети	765
10.6	Резервное копирование и восстановление системы	766
10.6.1	Статусы узла	767
10.6.2	Создание резервной копии	767
10.6.3	Обновление резервной копии	767
10.6.4	Скачивание резервной копии	768
10.6.5	Восстановление текущего узла из резервной копии	768
10.6.6	Восстановление из резервной копии на новом узле	768
10.7	Справочник команд (CLI)	769
10.7.1	Поддерживаемые команды	770
10.7.2	Синтаксис	772
10.7.3	certificates	772
10.7.3.1	list	773
10.7.3.2	info	773
10.7.3.3	upload	773
10.7.3.4	remove	774
10.7.4	configuration	774
10.7.4.1	commit	775
10.7.4.2	end	775
10.7.4.3	rollback	775
10.7.4.4	set	775
10.7.4.5	show	777
10.7.4.6	unset	777
10.7.5	diagnostics	777
10.7.5.1	arping	778
10.7.5.2	diag	778
10.7.5.3	nslookup	778
10.7.5.4	ping	780
10.7.5.5	pingc	780
10.7.5.6	ss	780
10.7.5.7	ssldump	782
10.7.5.8	tcpdump	783
10.7.5.9	traceroute	783

10.7.6	firewall	784
10.7.6.1	list	784
10.7.6.2	no open	785
10.7.6.3	open	785
10.7.6.4	save	786
10.7.7	ip	786
10.7.8	logs	786
10.7.8.1	Фильтрация вывода	788
10.7.9	modules	789
10.7.9.1	info	790
10.7.9.2	list	790
10.7.10	ps	790
10.7.11	settings	791
10.7.11.1	angie debug	791
10.7.11.2	no angie debug	792
10.7.11.3	reload	792
10.7.11.4	sysctl	792
10.7.11.5	syslog	792
10.7.12	system	794
10.7.12.1	ntp	794
10.7.12.2	reboot	794
10.7.12.3	stat	794
10.7.12.4	time	795
10.7.12.5	timezone	795
10.7.12.6	upgrade	795
10.7.12.7	version	795
10.7.13	transfer	795
10.7.13.1	edit	796
10.7.13.2	ls	796
10.7.13.3	rm	796
10.7.13.4	view	796
10.7.14	vttysh	796
10.7.15	web	796
10.7.15.1	upload	796
10.7.15.2	tls	797
10.8	Веб-интерфейс Angie ADC	797
10.8.1	Экран "Вход"	798
10.8.2	Элементы настройки веб-интерфейса	799
10.8.3	Безопасность учетной записи	799
10.8.3.1	Блок "Изменить пароль"	800
10.8.3.2	Блок "Включить TOTP"	801
10.8.3.3	Блок "Отключить TOTP"	801
10.8.4	Раздел "Система"	801
10.8.4.1	Вкладка "Конфигурация"	801
10.8.4.2	Вкладка "Пользователи"	804
10.8.4.3	Вкладка "Аутентификация и авторизация"	806
10.8.4.4	Вкладка "Парольная политика"	813
10.8.4.5	Вкладка "Сертификаты"	815
10.8.4.6	Вкладка "Пользователи SNMP"	816
10.8.4.7	Вкладка "Резервное копирование и восстановление"	817
10.8.4.8	Вкладка "NTP"	820
10.8.4.9	Вкладка "Логирование"	822
10.8.4.10	Вкладка "О решении"	823
10.8.5	Раздел "Мониторинг"	824
10.8.5.1	Вкладка "Балансировщик"	824
10.8.5.2	Вкладка "Система"	826
10.8.6	Раздел "Управление трафиком"	832
10.8.6.1	Вкладка "Балансировщик нагрузки"	832

10.8.6.2	Вкладка "Глобальная балансировка"	836
10.8.6.3	Вкладка "Аутентификация клиентов"	842
10.8.6.4	Вкладка "Конфигурация RHI"	845
10.8.7	Раздел "Высокая доступность и кластеризация"	846
10.8.7.1	Вкладка "Высокая доступность"	846
10.8.7.2	Вкладка "Кластер"	849
10.8.8	Раздел "Диагностика"	852
10.8.8.1	Вкладка "Сеть"	852
10.9	Парольная политика	853
10.9.1	Настройка парольной политики	854
10.9.2	Сброс истории паролей для пользователей	856
10.9.3	Сброс пароля локального пользователя	856
10.9.4	Сброс настроек двухфакторной аутентификации локального пользователя	856
10.10	Настройка учетной записи	856
10.10.1	Смена пароля учетной записи	858
10.10.2	Включение двухфакторной аутентификации	858
10.10.3	Отключение двухфакторной аутентификации	859
11	Типовые задачи и примеры	860
11.1	Настройка IPv6	860
11.1.1	Доступ к веб-интерфейсу Angie ADC	860
11.1.2	Настройка сетевых протоколов маршрутизации	860
11.1.3	Передача и обработка клиентского трафика	861
11.1.4	Поддержка смешанных подключений IPv4 и IPv6 в Angie ADC	861
11.1.5	Настройка ip6tables	863
11.2	Настройка пула SNAT (SNAT Pool)	864
11.2.1	Введение	864
11.2.2	Ручная настройка SNAT-пулов	864
11.2.2.1	1. Выбор пула адресов и их настройка	864
11.2.2.2	2. Настройка маршрутизации	865
11.2.2.3	3. Настройка правил балансировки	866
11.2.3	Заключение	867
11.3	Настройка Transparent Proxy для TCP- и UDP-трафика	868
11.3.1	IP Transparency	868
11.3.1.1	Предусловия	869
11.3.1.2	Пример настройки балансировщика	869
11.3.1.3	Настройка маршрутизации	869
11.3.2	Direct Server Return (DSR)	870
11.3.2.1	Пример настройки балансировщика	871
11.3.2.2	Переписывание IP-адреса и порта при ответе	871
11.4	Балансировка трафика на основе набора шифров	873
11.4.1	Настройка	873
11.4.1.1	Случай 1	874
11.4.1.2	Случай 2	874
11.4.2	Проверка	875
11.4.2.1	Случай 1	875
11.4.2.2	Случай 2	875
12	Миграция с других решений	877
12.1	Keepalive как аналог OneConnect profile от F5	877
12.1.1	Настройка	877
12.2	Передача IP-адреса клиента как аналог Use Source IP Mode (USIP) от Citrix NetScaler	879
13	Права на интеллектуальную собственность	880
	Алфавитный указатель	881

ГЛАВА 1

Аннотация

Настоящий документ содержит информацию, необходимую для эксплуатации программного обеспечения Angie ADC.

Angie ADC — программное обеспечение класса "контроллер доставки приложений", которое представляет собой систему балансировки, включающее DNS-балансировку, а также позволяющее маршрутизировать и балансировать сетевые запросы, используя протоколы маршрутизации внешнего и внутреннего шлюза.

глава 2

Общие сведения

Angie ADC — комплексное программное обеспечение для балансировки нагрузки и управления сетевым трафиком для создания гибкой, производительной и безопасной инфраструктуры.

Особенности:

- Балансировщик нагрузки на уровнях L4-L7.
- Глобальная DNS-балансировка (GSLB).
- IP-маршрутизация.
- Решения для обеспечения высокой доступности.
- Присутствие в реестре российского ПО.

Angie ADC имеет удобный веб-интерфейс, командную строку (CLI) и API для интеграции с внешними системами, что обеспечивает понятный и надежный мониторинг и управление функциями.

Angie ADC поставляется как виртуальное устройство (Virtual Appliance).

ГЛАВА 3

Первоначальная настройка

После установки и запуска виртуального образа Angie ADC рекомендуется выполнить общую настройку системы:

- настроить сеть;
- получить доступ к веб-интерфейсу;
- настроить точку входа в кластер;
- добавить пользователей и настроить авторизацию;
- настроить синхронизацию времени и конфигурацию балансировщика нагрузки.

Дополнительно можно настроить высокую доступность и глобальную балансировку.

В этом разделе:

- *Общая настройка*
- *Настройка в мастере*
- *Настройка сетевых интерфейсов*
- *Внешние пользователи*

3.1 Общая настройка

После установки и запуска виртуального образа Angie ADC необходимо создать кластер и выполнить общую настройку системы, а затем настроить конфигурацию балансировщика нагрузки.

3.1.1 Базовая настройка системы

1. Задайте сетевую конфигурацию Angie ADC через *мастер настройки*. Минимально необходимо настроить хотя бы один сетевой интерфейс (вручную или с помощью DHCP).

После этого у вас появится доступ к веб-интерфейсу Angie ADC по адресу `http://<IP-адрес веб-интерфейса>:8080`. IP-адрес веб-интерфейса можно посмотреть в консоли ВМ или по команде `sudo virsh net-dhcp-leases default`.

2. Перейдите на страницу веб-интерфейса в браузере и создайте *точку входа в кластер Angie ADC*.

В дальнейшем, если необходимо, вы сможете подключить другие узлы Angie ADC к этой точке входа.

3. Авторизуйтесь в системе. Реквизиты для первого входа предоставляются после покупки решения.
4. Задайте новый пароль администратора и войдите в веб-интерфейс.
Если необходимо, в дальнейшем можно настроить *парольную политику Angie ADC*.
5. Донастройте сетевые интерфейсы *через веб-интерфейс*, если это необходимо.
6. Настройте *HTTPS* для защищенного подключения к веб-интерфейсу.
7. Добавьте пользователей системы и настройте для них аутентификацию и авторизацию (поддерживаются *локальные и внешние пользователи*).
8. Настройте *синхронизацию времени по NTP*.
9. Настройте *конфигурацию балансировщика нагрузки* и использование *сертификатов*.

3.1.2 Расширенная настройка

После выполнения общей настройки системы дополнительно можно настроить:

- *кластеризацию Angie ADC*;
- высокую доступность для *standalone-устройства Angie ADC*;
- пару высокой доступности *для двух устройств Angie ADC*;
- *глобальную балансировку*.

3.1.3 Мониторинг

В веб-интерфейсе Angie ADC вам доступны:

- *Мониторинг работы балансировщика нагрузки*;
- *Мониторинг системных метрик Angie ADC*.

3.1.4 Логирование

Поддерживаются:

- просмотр *журналов событий*;
- подключение *syslog-сервера* для экспорта логов;
- отправка *SNMP-метрик*.

3.1.5 Примеры типовых схем развертывания Angie ADC

- В рамках одного дата-центра (DC) для локальной балансировки трафика:

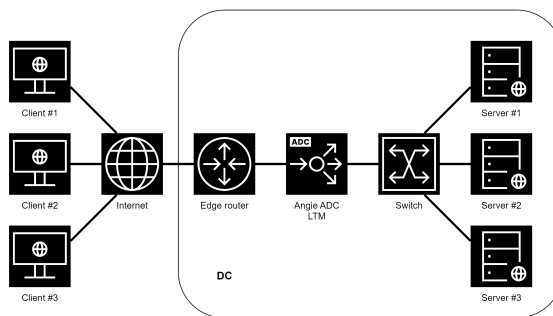


Рис. 1

- В двух и более дата-центрах с поддержкой резервирования и глобального управления трафиком (GSLB):

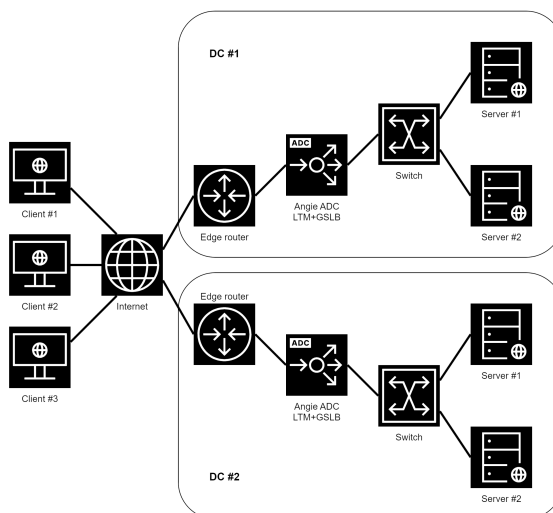


Рис. 2

См. также:

- `adc_installation`;
- *Балансировщик нагрузки.*

3.2 Настройка в мастере

Мастер настройки позволяет управлять конфигурацией Angie ADC. Также в мастере можно посмотреть сетевую конфигурацию и проверить доступность узла, выполнив `ping`-запрос.

При первом запуске Angie ADC вам необходимо настроить хотя бы один *сетевой интерфейс* для доступа к веб-интерфейсу Angie ADC.

При расширенной настройке вы можете дополнительно настроить другие интерфейсы, бонды и VLAN-интерфейсы, назначить выделенный интерфейс управления, изменить имя устройства и порт для веб-интерфейса.

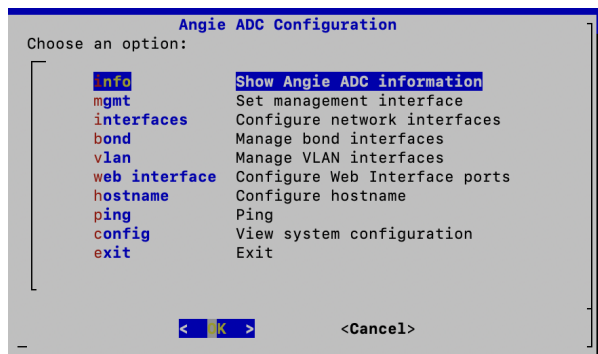
Для входа в мастер необходимо перейти в консоль виртуальной машины и в окне входа в мастер ввести предустановленные логин `setup` и пароль `setup`. Чтобы выйти из мастера настройки Angie ADC, нажмите `exit` в основном меню.

В этой статье:

- *Меню мастера*;
- *Настройка сетевых интерфейсов*;

- *Настройка интерфейса управления;*
- *Смена портов для веб-интерфейса Angie ADC;*
- *Создание бондов;*
- *Создание VLAN-интерфейсов;*
- *Просмотр конфигурации Angie ADC и сетевых настроек;*
- *Изменение имени устройства Angie ADC;*
- *Проверка доступности узла.*

3.2.1 Меню мастера



Основное меню мастера настройки Angie ADC содержит следующие команды:

- **info** — просмотр общей информации об Angie ADC.
- **mgmt** — настройка интерфейса управления Angie ADC.
- **interfaces** — переход к просмотру и настройке сетевых интерфейсов, бондов и VLAN-интерфейсов.
- **bond** — переход к созданию, просмотру и удалению сетевых бондов.
- **vlan** — переход к созданию, просмотру и удалению VLAN-интерфейсов.
- **web interface** — изменение порта для веб-интерфейса Angie ADC.
- **hostname** — смена имени устройства.
- **config** — просмотр конфигурации Angie ADC и сетевых настроек.
- **ping** — проверка доступности узла.
- **exit** — выход из мастера.

3.2.2 Настройка сетевых интерфейсов

Чтобы настроить сетевой интерфейс Angie ADC, выполните следующие действия:

1. Выберите в основном меню пункт **interfaces**.

Откроется список доступных интерфейсов.

2. Выберите интерфейс и перейдите в его меню.

Откроется список доступных действий:

- **show** — просмотр информации об интерфейсе.
- **config** — переход к ручной настройке IP-адреса и шлюза (Gateway).
- **dhcsp** — настройка автоматического получения IP-адреса от DHCP-сервера.

- **mtu** — переход к настройке MTU.
- **clear** — сброс всех заданных настроек интерфейса, кроме MTU. Значение MTU, заданное вручную, не сбрасываются.
- **back** — выход из меню интерфейса.

3. Выберите **config** или **dhcp** в зависимости от ваших целей и задайте необходимые настройки.

3.2.3 Настройка интерфейса управления

Рекомендуется использовать для управления отдельную сеть, в которой нет служебных сервисов, иначе эти сервисы перестанут быть доступными для других компонентов Angie ADC. Если вы хотите сохранить доступ к этим сервисам, после выделения интерфейса управления настройте для них маршруты-исключения.

Чтобы выделить отдельный интерфейс для управления Angie ADC, выполните следующие действия:

1. Выберите в основном меню пункт **mgmt**.

Откроется список доступных действий:

- **setup** — переход к настройке интерфейса управления.
- **status** — просмотр текущего статуса интерфейса управления.
- **leak** — переход к настройке маршрутов-исключений для сервисов внутри сети управления (пункт отображается, если интерфейс управления настроен).
- **back** — выход из меню **mgmt**.

2. В открывшемся меню выберите **setup**.

Откроется список доступных интерфейсов.

3. Выберите нужный интерфейс. Этот интерфейс будет использоваться для управления системой через отдельную сеть и будет иметь собственную таблицу маршрутизации.

4. Подтвердите операцию.

Предупреждение

Служебные сервисы, расположенные в этой сети, перестанут быть доступными для других компонентов Angie ADC.

5. Если необходимо, настройте маршруты-исключения для служебных сервисов. Для этого перейдите в меню **mgmt** → **leak** → **add** и добавьте сеть назначения, из которой будет производиться доступ к сервисам, находящимся в сети управления.

3.2.4 Создание бондов

При создании бондов используется протокол LACP (802.3ad) и режим **active** (интерфейс инициирует создание агрегированного канала).

Чтобы объединить несколько физических интерфейсов в один логический интерфейс (бонд), выполните следующие действия:

1. Выберите в основном меню пункт **bond**.

Откроется список доступных действий:

- **list** — просмотр списка бондов.
- **create** — переход к созданию нового бонда.
- **back** — выход из меню **bond**.

2. Выберите **create**, укажите номер бонда (по умолчанию будет предложен первый свободный номер по возрастанию) и нажмите **OK**.
3. Выберите интерфейсы, которые необходимо объединить (максимум 8 интерфейсов), и нажмите **OK**.

Примечание

Рекомендуется объединять в бонд ненастроенные сетевые интерфейсы.

4. В открывшемся окне с конфигурацией подтвердите создание нового бонда.

Просмотр свойств бонда и его удаление доступны через основное меню **bond** → **list** → **<bond_name>**.

Настройка бонда доступна через основное меню настройки интерфейсов: **interfaces** → **<bond_name>**.

3.2.5 Создание VLAN-интерфейсов

Чтобы настроить VLAN-интерфейс, выполните следующие действия:

1. Выберите в основном меню пункт **vlan**.

Откроется список доступных действий:

- **list** — просмотр списка VLAN-интерфейсов.
- **create** — переход к созданию нового VLAN-интерфейса.
- **back** — выход из меню **vlan**.

2. Выберите **create**, укажите родительский интерфейс, VLAN ID для нового VLAN и нажмите **OK**.

Будет создан новый VLAN-интерфейс.

Просмотр свойств VLAN и его удаление доступны через основное меню **vlan** → **list** → **<vlan_name>**.

Настройка VLAN доступна через основное меню настройки интерфейсов: **interfaces** → **<vlan_name>**.

3.2.6 Просмотр конфигурации Angie ADC и сетевых настроек

Чтобы просмотреть текущую конфигурацию Angie ADC или конфигурацию **systemd-networkd** для заданных сетей, выполните следующие действия:

1. Выберите в основном меню пункт **config**.

Откроется список доступных действий:

- **system** — просмотр сетевой конфигурации Angie ADC.
- **network** — просмотр конфигурации **systemd-networkd** для заданных сетей.
- **back** — выход из меню **config**.

2. Выберите **system**, чтобы просмотреть текущую конфигурацию Angie ADC.

3. Выберите **network** → **<имя сети>**, чтобы просмотреть настройки **systemd-networkd** для этой сети.

3.2.7 Смена портов для веб-интерфейса Angie ADC

Вы можете назначить для веб-интерфейса Angie ADC любой незанятый порт. Чтобы изменить порты по умолчанию, выполните следующие действия:

1. Выберите в основном меню пункт **web interface**.

Откроется список доступных действий:

- **set** — назначение новых портов.
- **reset** — восстановление настроек по умолчанию.
- **logs** — просмотр журнала.
- **back** — выход из меню **web interface**.

2. Выберите **set**, укажите новые значения портов и примените настройки.

3.2.8 Изменение имени устройства Angie ADC

Чтобы изменить имя устройства Angie ADC, выполните следующие действия:

1. Выберите в основном меню пункт **hostname**.
2. В открывшемся окне введите новое имя устройства и сохраните изменения.

3.2.9 Проверка доступности узла

Чтобы проверить доступность узла с помощью ping-запроса, выполните следующие действия:

1. Выберите в основном меню пункт **ping**.
2. В открывшемся окне введите имя или IP-адрес узла, доступность которого нужно проверить.

3.3 Настройка сетевых интерфейсов

Вы можете просматривать и изменять параметры сетевых интерфейсов в веб-интерфейсе Angie ADC (**Система** → **Конфигурация**).

Предупреждение

Не рекомендуется изменять настройки интерфейсов узлов, объединенных в пару высокой доступности, иначе пара может потерять связность и работоспособность.

3.3.1 Изменение параметров сетевого интерфейса

1. В веб-интерфейсе Angie ADC перейдите **Система** → **Конфигурация**.
2. В блоке **Сетевые интерфейсы** нажмите на сетевой интерфейс, параметры которого вы хотите изменить.

Откроется окно с параметрами выбранного сетевого интерфейса.

Примечание

Если для интерфейса был указан способ получения настроек по DHCP, то в окне редактирования будет отображаться подсказка с DHCP-параметрами интерфейса.

3. Внесите необходимые изменения и нажмите **Сохранить**.

Также изменение параметров сетевых интерфейсов доступно через *CLI* и *в мастере*.

3.3.2 Просмотр параметров сетевого интерфейса

В блоке **Сетевые интерфейсы** отображается список настроенных сетевых интерфейсов.

Для каждого интерфейса отображаются следующие параметры:

- Имя сетевого интерфейса.
- MAC-адрес (формат: "XX:XX:XX:XX:XX:XX").
- Тип конфигурации: **static**, **DHCP**.
- Массив IPv4-адресов в формате CIDR.
- Массив IPv6-адресов в формате CIDR.
- IP-адрес шлюза по умолчанию (глобальный для всех интерфейсов), если установлен.
- Массив IP-адресов DNS-серверов.
- Массив доменов поиска DNS.

3.4 Подключение внешних пользователей

При использовании внешнего провайдера аутентификации и авторизации данные о пользователях хранятся на внешнем сервере. Вы можете настроить аутентификацию и авторизацию внешних пользователей в Angie ADC через провайдеров RADIUS, TACACS и LDAP.

Пользователям можно назначить одну из следующих ролей:

- **Администратор**: обладает полными правами на все действия в системе;
- **Супервизор**: имеет ограниченные права, может только просматривать информацию в системе, например, проводить аудит;
- **Оператор**: имеет ограниченные права, может редактировать файлы конфигурации балансировщика, к которым ему предоставлен доступ, и просматривать статистику зон, к которым ему предоставлен доступ. Доступы предоставляет администратор. Остальные настройки Angie ADC оператору не доступны и не отображаются в веб-интерфейсе.

Для включения внешней аутентификации и авторизации необходимо добавить в систему предварительно настроенный сервер. После этого при входе в веб-интерфейс будет доступен выбор типа аутентификации.

В этом разделе:

- [RADIUS](#)
- [LDAP](#)
- [TACACS+](#)

3.4.1 Подключение внешних пользователей через LDAP

Вы можете включить и настроить аутентификацию и авторизацию для внешних пользователей через LDAP. Пользователям может быть назначена одна из следующих ролей:

- **Администратор**: обладает полными правами на все действия в системе;
- **Супервизор**: имеет ограниченные права, может только просматривать информацию в системе, например, проводить аудит;
- **Оператор**: имеет ограниченные права, может редактировать файлы конфигурации балансировщика, к которым ему предоставлен доступ, и просматривать статистику зон, к которым ему предоставлен доступ. Доступы предоставляет администратор. Остальные настройки Angie ADC оператору не доступны и не отображаются в веб-интерфейсе.

Роли внешних пользователей определяются на основе принадлежности к LDAP-группам.

Для подключения внешних пользователей к системе вам необходимо:

1. включить внешнюю аутентификацию через LDAP и задать настройки сервера;
2. включить внешнюю авторизацию через LDAP и задать настройки сервера;
3. задать базовый DN и фильтры для поиска пользователей и групп на LDAP-сервере;
4. настроить соответствие LDAP-групп и ролей пользователей в Angie ADC.

На стороне LDAP-сервера также должны быть настроены пользователи, группы и сервисная учетная запись для Angie ADC.

⚠ Предупреждение

Необходимо настроить и включить оба блока: аутентификацию и авторизацию. Если один из блоков не настроен или выключен, внешние пользователи не смогут получить доступ к системе.

3.4.1.1 Шаги

1. Откройте веб-интерфейс Angie ADC. В панели навигации перейдите в раздел Система → Аутентификация и авторизация.

Откроется страница с настройками аутентификации и авторизации для внешних пользователей.

The screenshot displays two tabs: 'Внешняя аутентификация' (External authentication) and 'Внешняя авторизация' (External authorization). The 'Внешняя аутентификация' tab is active, showing fields for LDAP server connection (Server, Port, DN of service account), search settings (Base DN, Filter), and user/group mapping. The 'Внешняя авторизация' tab is also visible, showing similar fields for authorization settings.

2. Включите переключатель Включить внешнюю аутентификацию и выберите провайдера аутентификации LDAP.

i Примечание

Одновременно поддерживается использование только одного провайдера внешней аутентификации. При добавлении нового провайдера аутентификации настройки предыдущего провайдера сбрасываются.

3. Укажите следующие параметры:

- Сервер: IP-адрес или доменное имя LDAP-сервера.
- Порт: порт подключения к LDAP-серверу (по умолчанию 389).

- Режим защиты соединения (по умолчанию Без TLS, также доступны варианты LDAPS и StartTLS для защищенного соединения).

Примечание

При выборе режима защиты соединения LDAPS или StartTLS становится доступен флажок Проверять TLS-сертификат. По умолчанию флажок установлен.

- DN сервисной учетной записи: логин сервисной учетной записи, от имени которой система подключается к LDAP.
- Пароль сервисной учетной записи: пароль сервисной учетной записи для подключения к LDAP-серверу.
- Базовый DN пользователя: задает ветку на LDAP-сервере, в которой система будет искать пользователя, например ou=users,dc=example,dc=com.
- Идентификатор логина: поле в LDAP, которое содержит логин, например uid, sAMAccountName или userPrincipalName.
- Фильтр поиска: LDAP-фильтр для поиска пользователя, например (&(objectClass=user)(sAMAccountName=%s)).

Примечание

Фильтр LDAP определяет, в каком формате пользователь должен вводить логин при входе в систему:

- если используется фильтр (&(objectClass=user)(sAMAccountName=%s)), то логин будет без доменного имени, например ivanov;
- если используется фильтр (&(objectClass=user)(userPrincipalName=%s)), то логин будет полным: ivanov@domain.com.

4. Нажмите Сохранить.

Внешняя аутентификация с использованием провайдера LDAP будет включена.

5. В блоке Внешняя авторизация переведите переключатель Включить внешнюю авторизацию в положение "включено" и выберите провайдера авторизации LDAP.

6. Задайте параметры подключения к серверу:

- Сервер: IP-адрес или доменное имя LDAP-сервера.
- Порт: порт подключения к LDAP-серверу (по умолчанию 389).

7. Укажите параметры защиты соединения: Без TLS (по умолчанию) или TLS (LDAPS)/startTLS, если необходимо использовать безопасное соединение при подключении к серверу.

Примечание

При выборе режима защиты соединения LDAPS или StartTLS становится доступен флажок Проверять TLS-сертификат. По умолчанию флажок установлен.

8. Укажите DN учетной записи, с которой будет выполняться подключение к LDAP-серверу (DN сервисной учетная запись (bind)). Можно использовать учетную запись администратора LDAP-сервера, например cn=admin,dc=example,dc=com.

9. Укажите пароль сервисной учетной записи (Пароль пользователя bind).

10. В блоке Поиск пользователя укажите следующие параметры:

- Базовый DN пользователей, например `ou=users,dc=example,dc=com`;
- Фильтр поиска пользователей, например `(uid=%s)`.

11. В блоке Поиск групп укажите следующие параметры:

- Базовый DN групп, например `ou=groups,dc=example,dc=com`;
- Фильтр поиска групп, например `(member=%s)`.

12. В блоке Мapping ролей LDAP нажмите **Добавить правило** и задайте соответствие LDAP-групп пользователей ролям в Angie ADC. Например, если на LDAP-сервере для администраторов создана группа `dn: cn=<group_name>,ou=groups,dc=example,dc=com`, то укажите значение `cn=<group_name>,ou=groups,dc=example,dc=com` в поле DN группы и выберите роль Администратор (admin).

13. Нажмите **Сохранить**.

Внешняя авторизация будет настроена и включена. При входе внешнего пользователя в систему на LDAP-сервере будет выполняться поиск групп, в которых он состоит. Если найденные группы совпадают с группами, указанными в mappingе, то пользователю будет назначена соответствующая роль. Если пользователь не найден ни в одной из групп или LDAP-группы недоступны или некорректны, доступ к системе будет запрещен.

3.4.2 Подключение внешних пользователей через TACACS

Вы можете включить и настроить аутентификацию и авторизацию для внешних пользователей через TACACS. Пользователям может быть назначена одна из следующих ролей:

- **Администратор:** обладает полными правами на все действия в системе;
- **Супервизор:** имеет ограниченные права, может только просматривать информацию в системе, например, проводить аудит;
- **Оператор:** имеет ограниченные права, может редактировать файлы конфигурации балансировщика, к которым ему предоставлен доступ, и просматривать статистику зон, к которым ему предоставлен доступ. Доступы предоставляет администратор. Остальные настройки Angie ADC оператору не доступны и не отображаются в веб-интерфейсе.

Роли для внешних пользователей определяются на основе соответствия атрибутам этих пользователей в TACACS+.

Для подключения внешних пользователей к системе вам необходимо:

1. включить внешнюю аутентификацию через TACACS и задать настройки сервера;
2. включить внешнюю авторизацию через TACACS+ и задать настройки сервера;
3. настроить соответствие атрибутов TACACS+ ролям пользователей в Angie ADC.

На стороне TACACS-сервера также должны быть настроены пользователи и группы.

Пример настройки:

```
group = admin {
    service = shell {
        set priv-lvl = 15
    }
}

user = admin {
    password = clear admin
    member = admin
}
```

Предупреждение

Необходимо настроить и включить оба блока: аутентификацию и авторизацию. Если один из блоков не настроен или выключен, внешние пользователи не смогут получить доступ к системе.

3.4.2.1 Шаги

1. Откройте веб-интерфейс Angie ADC. В панели навигации перейдите в раздел Система → Аутентификация и авторизация.

Откроется страница с настройками аутентификации и авторизации для внешних пользователей.

adc-psl-2

Аутентификация и авторизация

Аутентификация и авторизация

Внешняя аутентификация

RADIUS

TACACS

LDAP

Сервер

Порт

Секрет

FQDN или IPv4/IPv6

49

Введите новый секрет (если нужно обновить)

Сохранить

Внешняя авторизация

RADIUS

TACACS

LDAP

Подключение TACACS-сервера

Сервер

Порт

FQDN или IPv4/IPv6

49

Секрет

Введите новый секрет (если меняете)

Подключаться с секретом

Настройка TACACS-авторизации

Ключи ролей в TACACS (по приоритету)

Параметры AUTHOR-запроса

priv-lvl
role

service=shell
protocol=ip

Мэппинг ролей

Правила мэппинга TACACS

Правила мэппинга TACACS не добавлены.

+

Добавить правило

Сохранить

2. Включите переключатель Включить внешнюю аутентификацию и выберите провайдера аутентификации TACACS.

Примечание

Одновременно поддерживается использование только одного провайдера внешней аутентификации. При добавлении нового провайдера аутентификации настройки предыдущего провайдера сбрасываются.

3. Укажите следующие параметры:

- **Адрес:** IP-адрес или доменное имя TACACS-сервера.
- **Порт:** порт подключения к TACACS-серверу (по умолчанию 49).
- **Секрет:** общий секрет для шифрования паролей.

4. Нажмите Сохранить.

Внешняя аутентификация с использованием провайдера TACACS будет включена.

5. В блоке **Внешняя авторизация** переведите переключатель **Включить внешнюю авторизацию** в положение "включено" и выберите провайдера авторизации **TACACS**.
6. Задайте параметры подключения к серверу:

- Адрес: IP-адрес или доменное имя TACACS-сервера.
 - Порт: порт подключения к TACACS-серверу (по умолчанию 49).
 - Секрет: общий секрет для шифрования паролей.
7. Укажите в порядке приоритета атрибуты ролей пользователя в TACACS+, например `priv-lvl`, затем `role`. По этим атрибутам Angie ADC будет определять роль для пользователя.
 8. Укажите, какой сервис и протокол будут использоваться в запросе на авторизацию (AUTHOR). Параметры зависят от выбранного сервера. Например, для TACACS+ рекомендуется указать `service=shell` и `protocol=ip`.
 9. Задайте соответствие значений атрибутов TACACS+ ролям в Angie ADC, например `priv-lvl=15` → Администратор или `role=adc-admin` → Администратор.
 10. Нажмите Сохранить.

Внешняя авторизация будет настроена и включена. При входе внешнего пользователя в систему на TACACS-сервер будет отправлен запрос на авторизацию. Если атрибуты и значения в ответе TACACS совпадают с указанными в маппинге, то пользователю будет назначена соответствующая роль в Angie ADC. Если полученные данные не совпадают с указанными в маппинге или TACACS-сервер недоступен, доступ к системе будет запрещен.

3.4.3 Подключение внешних пользователей через RADIUS

Вы можете включить и настроить аутентификацию и авторизацию для внешних пользователей через RADIUS. Пользователям может быть назначена одна из следующих ролей:

- **Администратор:** обладает полными правами на все действия в системе;
- **Супервизор:** имеет ограниченные права, может только просматривать информацию в системе, например, проводить аудит;
- **Оператор:** имеет ограниченные права, может редактировать файлы конфигурации балансировщика, к которым ему предоставлен доступ, и просматривать статистику зон, к которым ему предоставлен доступ. Доступы предоставляет администратор. Остальные настройки Angie ADC оператору не доступны и не отображаются в веб-интерфейсе.

Роли для внешних пользователей определяются на основе соответствия атрибутам этих пользователей в RADIUS.

Примечание

RADIUS-авторизация доступна только при аутентификации через RADIUS.

Для подключения внешних пользователей к системе вам необходимо:

1. включить внешнюю аутентификацию через RADIUS и задать настройки сервера;
2. включить внешнюю авторизацию через RADIUS и задать настройки сервера;
3. настроить соответствие атрибутов RADIUS ролям пользователей в Angie ADC.

На стороне RADIUS-сервера также должны быть настроены пользователи и их атрибуты.

Пример настройки:

```
user1 Cleartext-Password := "P@ssword"
  Filter-Id := "adc-admin",
  Service-Type := Administrative-User,
  Cisco-AVPair := "shell:priv-lvl=15",
  Reply-Message := "Welcome, Admin",
  Class := "AdminGroup",
```

```
Tunnel-Type := VLAN,
Tunnel-Medium-Type := IEEE-802,
Tunnel-Private-Group-ID := "100"

user2 Cleartext-Password := "P@ssword"
Filter-Id := "adc-readonly",
Service-Type := NAS-Prompt-User,
Cisco-AVPair := "shell:priv-lvl=7",
Reply-Message := "Welcome, Supervisor",
Class := "SupervisorGroup",
Tunnel-Type := VLAN,
Tunnel-Medium-Type := IEEE-802,
Tunnel-Private-Group-ID := "200"
```

⚠ Предупреждение

Необходимо настроить и включить оба блока: аутентификацию и авторизацию. Если один из блоков не настроен или выключен, внешние пользователи не смогут получить доступ к системе.

3.4.3.1 Шаги

1. Откройте веб-интерфейс Angie ADC. В панели навигации перейдите в раздел Система → Аутентификация и авторизация.

Откроется страница с настройками аутентификации и авторизации для внешних пользователей.

adc-psi-2 > Аутентификация и авторизация

Аутентификация и авторизация

☒ Внешняя аутентификация
 RADIUS TACACS LDAP

Сервер
 FQDN или IPv4/IPv6

Порт
 1812

Протокол
 UDP

Метод аутентификации
 PAP

NAS-идентификатор
 adc-backend

Секрет
 Введите новый секрет (если нужно обновить)

Сохранить

☒ Внешняя авторизация
 RADIUS TACACS LDAP

RADIUS-авторизация доступна только при аутентификации через RADIUS.

Настройка RADIUS-авторизации
 Атрибут роли в RADIUS
 filter-id

Маллинг ролей
 Правила малпинга RADIUS
 Правила малпинга RADIUS не добавлены.
 + Добавить правило

Сохранить

2. Включите переключатель Включить внешнюю аутентификацию и выберите провайдера аутентификации RADIUS.

ℹ Примечание

Одновременно поддерживается использование только одного провайдера внешней аутентификации. При добавлении нового провайдера аутентификации настройки предыдущего провайдера сбрасываются.

3. Укажите следующие параметры:

- **Сервер:** IP-адрес или доменное имя RADIUS-сервера.
- **Порт:** порт подключения к RADIUS-серверу (по умолчанию 1812).
- **Протокол:** протокол передачи данных: UDP или TCP (по умолчанию: UDP).
- **NAS-идентификатор:** идентификатор NAS (Network Access Server), передаваемый серверу RADIUS.
- **Метод аутентификации:** поддерживается только PAP.
- **Секрет:** общий секрет для шифрования паролей.

4. Нажмите **Сохранить**.

Внешняя аутентификация с использованием RADIUS будет включена.

5. В блоке **Внешняя авторизация** переведите переключатель **Включить внешнюю авторизацию** в положение "включено" и выберите провайдера авторизации RADIUS.
6. Укажите атрибут роли пользователя в RADIUS, например **Filter-Id**, **vsa:9:1** или другой используемый атрибут. По этому атрибуту Angie ADC будет определять роль для пользователя.

Допускается использование следующих атрибутов ролей:

- **Filter-Id** — роль пользователя.
 - **Class** — группа, к которой относится пользователь.
 - **Service-Type** — тип сервиса, по которому определяется роль пользователя (Administrative, Login и др.)
 - **Tunnel-Private-Group-ID** — VLAN ID (используется для VLAN-авторизации).
 - **VSA** — Vendor-Specific в формате **vsa:<vendorID>:<vendorType>** (например **vsa:9:1**).
7. Задайте соответствие значений атрибутов RADIUS ролям в Angie ADC, например для **Filter-Id** укажите **adc-admin** → **Администратор**, а при использовании **vsa:9:1** укажите **shell:priv-lvl=15** → **Администратор**.

8. Нажмите **Сохранить**.

Внешняя авторизация через RADIUS будет настроена и включена. При входе внешнего пользователя в систему через RADIUS будут получены атрибуты его роли. Если атрибуты и значения совпадают с указанными в маппинге, то пользователю будет назначена соответствующая роль в Angie ADC. Если данные не совпадают с указанными в маппинге или RADIUS-сервер недоступен, доступ к системе будет запрещен.

ГЛАВА 4

Балансировщик нагрузки

Балансировка

Балансировщик нагрузки позволяет распределять клиентские запросы по серверам на основании выбранного алгоритма балансировки, данных статистики или обратной связи от сервера (см. *Мемо-ды балансировки*). В идеале клиентские запросы равномерно распределяются по всем серверам, что увеличивает совокупную производительность системы. Доступна балансировка на двух уровнях: L7 (протокол HTTP) и L4 (TCP и UDP).

Высокая доступность

Высокая доступность при балансировке обеспечивается за счет *проверок работоспособности апстрим-серверов* и *резервирования проксируемых серверов и групп*.

Примечание

Отказоустойчивость самого балансировщика нагрузки обеспечивается отдельно, например с помощью *пары высокой доступности*.

Дополнительная настройка

Помимо алгоритмов распределения запросов, в HTTP- и TCP/UDP-балансировке могут использоваться смежные модули:

- Модуль `queue` создает очередь запросов. Если балансировщик не смог выбрать апстрим-сервер для обработки, то запрос попадает в очередь ожидания. Также можно задать *приоритизацию запросов* в очереди. Приоритетные запросы будут обрабатываться в первую очередь, что позволит в условиях критической перегрузки сохранить обслуживание наиболее важных клиентов, отсекая второстепенные запросы и повышая устойчивость к DDoS-атакам.
- Модуль `keepalive` позволяет держать открытые неактивные соединения с апстрим-серверами, т. е. не закрывать соединения при завершении обработки запроса.
- Директива `bind_conn`, реализованная в модуле `keepalive`, позволяет привязать клиентское соединение к серверному соединению.
- Модуль `sticky` также позволяет привязывать клиентское соединение к апстрим-серверу, но на основе информации, содержащейся в самом запросе.
- Модуль `zone` обеспечивает доступ с использованием REST API. API позволяет получать набор метрик, в том числе относящихся к балансировке.

- Модуль `upstream_probe` позволяет обеспечить активные проверки апстрим-серверов.

Подробнее см. в справочнике HTTP-модуля *Upstream* и потокового модуля *Upstream*.

Хранение версий конфигурации

Все версии конфигурации балансировщика нагрузки сохраняются. Вы можете просмотреть историю изменений, а также *восстановить конфигурацию* из резервной копии.

Справочная информация

В разделе *Справочная информация* представлены сведения о встроенных и сторонних модулях, примеры их настройки, а также поддерживаемые ими директивы и переменные.

В этом разделе:

- *Настройка конфигурации*
- *Управление резервными конфигурациями*
- *Методы балансировки*
- *Делегирование сервиса оператору*
- *Предоставление доступов на работу с конфигурацией*
- *Проверки работоспособности серверов*
- *Резервирование проксируемых серверов*
- *Загрузка и раздача статических файлов*
- *Аутентификация клиентов*
- *Справочная информация*

4.1 Настройка конфигурации

4.1.1 Общая информация

Конфигурацию балансировщика нагрузки можно просмотреть и отредактировать в веб-интерфейсе Angie ADC (Управление трафиком → Балансировщик нагрузки).

The screenshot shows the Angie ADC web interface. At the top, a status bar indicates 'Работает' (Working) and 'Перезагрузка: 40 мин'. Below this, a sidebar on the left lists files: 'http.d', 'default.conf', 'stream.d', 'example.conf', 'angle.conf' (selected), 'fastcgi_params', 'fastcgi.conf', 'mime.types', 'prometheus_all.conf', 'scgi_params', and 'uwsgi_params'. The main area displays the configuration for 'angle.conf' with line numbers 1 through 45. The configuration includes settings for user, worker_processes, error_log, pid, events, http, log_format, access_log, sendfile, tcp_nopush, keepalive_timeout, gzip, and server. At the bottom, there are buttons for 'Управление доступами', 'Сбросить', and 'Сохранить'.

В левой части экрана отображается список файлов конфигурации, к которым есть *доступ* у пользователя. В правой части экрана отображается содержимое каждого файла.

По умолчанию конфигурация балансировщика содержит следующие файлы и папки:

- папка **http.d** — содержит конфигурации веб-сайтов и веб-приложений, работающих по протоколам HTTP/HTTPS:
 - файл **default.conf** — пример настройки виртуального хоста; этот файл не используется.
- папка **stream.d** — настройки проксирования трафика на уровне транспортного протокола (TCP и UDP):
 - файл **example.conf** — пример настройки потокового (TCP/UDP) проксирования; этот файл не используется.
- файл **angie.conf** — главный конфигурационный файл веб-сервера; запускает процессы, определяет пути к логам и подключает все остальные папки и файлы;
- файл **fastcgi.conf** — настройки для работы с PHP (через PHP-FPM) и другими FastCGI-приложениями;
- файл **fastcgi_params** — аналог fastcgi.conf, содержащий базовые переменные окружения для FastCGI;
- файл **mime.types** — карта соответствия расширений файлов и их типов данных (например .html → text/html, .jpg → image/jpeg);
- файл **prometheus_all.conf** — файл настройки экспорта метрик в формате Prometheus для мониторинга (например, в Grafana);
- файл **scgi_params** — параметры для работы по протоколу SCGI;
- файл **uwsgi_params** — параметры для интеграции с приложениями на Python.

Администратор может добавлять новые файлы в папки **http.d** и **stream.d**, нажав на значок добавления напротив соответствующей папки, и удалять любые существующие файлы, нажав на три точки в нижней части экрана соответствующего файла и выбрав пункт **Удалить файл**.

При сохранении изменений конфигурация обновляется и применяется сразу. Если при сохранении файла возникает конфликт (другой пользователь уже внес изменения в файл), то система подсвечивает те места, где правки конфликтуют, и предлагает выбрать, какую версию файла сохранить: пользовательскую (**Сохранить мою версию**) или серверную (**Принять серверную версию**).

4.1.2 Перезапуск балансировщика

В верхней части экрана отображается статус балансировщика:

- **Работает** — балансировщик запущен;
- **Не работает** — балансировщик не запустился после перезагрузки конфигурации.

Также отображается время, прошедшее с последней перезагрузки (**Перезагрузка**) и количество перезагрузок при наведении на время (**Поколение**).

Совет

Если у балансировщика отображается статус **Не работает**, необходимо проверить журнал ошибок балансировщика (см. ниже) и устранить ошибки.

Также можно попробовать перезапустить балансировщик вручную. Для этого откройте любой файл конфигурации балансировщика и сохраните его без внесения изменений.

4.1.3 Логи балансировщика

В веб-интерфейсе можно просмотреть журнал ошибок балансировщика нагрузки. Для этого перейдите в раздел **Управление трафиком** → **Балансировщик нагрузки** и в нижней части экрана щелкните многоточие (...) и выберите **Логи**.

Также см. *Журналы событий и экспорт* и *diagnostics*.

4.1.4 Принцип конфигурации

Для настройки балансировщика нагрузки используется декларативная конфигурация. Балансировщик нагрузки состоит из модулей, которые настраиваются директивами, заданными в конфигурации. Директивы могут быть простыми и блочными. Простая директива состоит из имени и параметров, разделенных пробелами, и оканчивается точкой с запятой ;. В блочной директиве вместо точки с запятой после имени и параметров следует набор дополнительных инструкций внутри фигурных скобок { и }. Блочные директивы называются контекстами.

Примеры:

- контекст *events* (общая обработка соединений);
- контекст *http* (конфигурация HTTP-балансировки);
- контекст *stream* (конфигурация TCP/UDP балансировки).

Контексты *events*, *http*, *stream* располагаются в контексте *main*. Другие директивы, размещенные вне контекстов *events*, *http*, *stream*, считаются также находящимися в контексте *main*.

В контекстах *http* и *stream* размещаются:

- блоки *server*, в которых настраивается проксирование трафика;
- блоки *upstream*, в которых настраиваются группы upstream-серверов для балансировки нагрузки.

В конфигурации в веб-интерфейсе Angie ADC уже предустановлены некоторые общие параметры. Вам необходимо дополнить конфигурацию с учетом ваших целей.

Подробнее о конфигурации и обработке соединений см. *Конфигурационные файлы* и *Соединения, сессии, запросы, логи*.

Предупреждение

Строки конфигурации, приведенные ниже, нельзя удалять или менять (они используются для обеспечения работы соответствующего функционала Angie ADC).

Контекст *main*:

```
error_log /var/log/angie-lb/healthmonitoring.log error;
pid      /run/angie-lb.pid;
```

Контекст *http*:

```
include /etc/angie-lb/prometheus_all.conf;
include /etc/angie-lb/internal/prometheus.conf;
```

4.1.5 Использование портов

Если вы планируете использовать в конфигурации другие порты помимо 80/443, вы можете открыть их с помощью команды *open* через CLI Angie ADC.

Предупреждение

При настройке конфигурации не занимайте следующие порты (используются внутренними сервисами Angie ADC):

TCP: 22, 199, 2022, 2222, 2601, 2602, 2603, 2604, 2605, 2606, 2615, 2616, 2617, 2619, 2623, 3050, 3051, 3053, 3054, 3060, 3111, 3122, 3123, 3301, 3302, 3380, 3391, 5044, 5355, 5432, 8010, 8080, 8443, 9090, 9100, 9120, 9600, 9633, 9898, 9900, 60080

- Порты 8080 и 8443 можно использовать, если у вас выделен отдельный *интерфейс управления*.

UDP: 53, 161, 323, 546, 3784, 3785, 4784, 5355, 7784

4.1.6 Настройка простого прокси-сервера

Angie ADC можно использовать в качестве прокси-сервера, который принимает запросы, перенаправляет их на проксируемые серверы, получает от них ответы и отправляет их клиенту.

В примере ниже настраивается базовый прокси-сервер, который будет обслуживать запросы изображений из локального каталога и отправлять все остальные запросы на проксируемый сервер.

1. Создайте сервер, который будет использоваться в качестве бэкенда для прокси (проксируемый сервер), добавив блок **server** в контекст **http**:

```
server {
    listen 8081;
    root /static/www;

    location / {
    }
}
```

Здесь сервер слушает на порту 8081 и сопоставляет URI запросов с файлами в каталоге `/static/www` (как настроить загрузку и раздачу статических файлов см. соотв. статью). Директива `root` находится в контексте **server** и будет использоваться, если директива `location`, выбранная для выполнения запроса, не содержит собственной директивы `root`.

Примечание

В общем случае конфигурационный файл может содержать несколько блоков **server**, отличающихся портами, на которых они слушают (*listen*), и именами сервера (*server_name*). Определив, какой **server** будет обрабатывать запрос, Angie ADC сравнивает URI запроса с параметрами *location*, заданными в блоке **server**, и принимает решение.

2. Добавьте конфигурацию прокси-сервера:

```
server {
    listen 80;
    location / {
        proxy_pass http://localhost:8081/;
    }

    location ~ /\.(gif|jpg|png)$ {
        root /static/images;
    }
}
```

В первом блоке `location` добавлена директива `proxy_pass` с указанием протокола, имени и порта проксируемого сервера. Во втором блоке `location` задано регулярное выражение

для всех URI, оканчивающихся на `.gif`, `.jpg` или `.png`. Соответствующие запросы будут обслуживаться из каталога `/static/images`.

Примечание

При выборе блока `location`, который будет обслуживать запрос, сначала проверяются директивы `location`, задающие префиксы, и запоминается `location` с самым длинным подходящим префиксом. Затем проверяются блоки `location`, заданные регулярными выражениями, в порядке их объявления в конфигурации и используется первое совпадение. Иначе берется `location`, сохраненный до этого.

3. Сохраните изменения в конфигурации, нажав на кнопку **Сохранить**.

Получившийся прокси-сервер будет обслуживать запросы к файлам `.gif`, `.jpg` или `.png` из каталога `/static/images`, а все остальные запросы будет проксировать на сервер `localhost:8081`.

4.1.7 Настройка балансировки нагрузки

Чтобы настроить балансировку нагрузки для HTTP-трафика, добавьте блок `upstream` в контекст `http` и укажите в нем бэкенд-серверы, которые будут использоваться для балансировки нагрузки. Затем настройте проксирование запросов на эту группу серверов в блоке `server` с помощью директивы `location`.

Пример:

```
upstream backend {
    zone backend 1m;
    server backend1.example.com;
    server backend2.example.com;
}

server {
    listen 80;
    server_name localhost;

    location /api {
        proxy_pass http://backend;
    }
}
```

В приведенном примере задана группа серверов `backend`. Все запросы на порт 80 с URI, начинающимися с `/api`, проксируются в эту группу. Метод балансировки нагрузки не указан явно, поэтому по умолчанию используется метод `round-robin` (запросы распределяются по серверам последовательно).

4.1.8 Раширенная настройка

Дополнительно в конфигурации вы можете настроить:

- *метод балансировки*;
- *проверки работоспособности*;
- *резервирование серверов и групп* для отказоустойчивости;
- смежные модули для HTTP-балансировки (`keepalive`, `sticky`, `zone`, `queue`) и stream-балансировки (`sticky`, `zone`).

Подробнее см. в справочнике HTTP-модуля *Upstream* и потокового модуля *Upstream*.

4.2 Управление резервными конфигурациями

В Angie ADC реализовано хранение всех версий конфигурационных файлов балансировщика нагрузки. При сохранении изменений в любом файле конфигурации балансировщика Angie ADC автоматически применяет новую конфигурацию и сохраняет предыдущую версию в репозитории. Можно просмотреть список сохраненных версий конфигурации и применить нужную версию. Также поддерживается *выбор внутреннего или внешнего репозитория* для хранения конфигураций.

4.2.1 Просмотр списка сохраненных конфигураций

Чтобы просмотреть список сохраненных конфигураций:

1. Откройте веб-интерфейс Angie ADC.
2. Выберите **Управление трафиком** → **Балансировщик нагрузки**.
3. В нижней части экрана щелкните многоточие (...) и выберите **История**.

Откроется таблица со списком сохраненных конфигураций. Для каждой конфигурации указано время ее создания, версия Angie ADC, статус (рабочая или нерабочая конфигурация), пользовательский комментарий, теги (**angie-lb** и **UUID**) и дата последнего изменения.

4.2.2 Изменение рабочего статуса хранимой конфигурации

Чтобы пометить конфигурацию как рабочую или нерабочую:

1. Откройте веб-интерфейс Angie ADC.
2. Выберите **Управление трафиком** → **Балансировщик нагрузки**.
3. В нижней части экрана щелкните многоточие (...) и выберите **История**.
Откроется таблица со списком сохраненных конфигураций.
4. Щелкните многоточие (...) напротив конфигурации, статус которой вы хотите изменить, и нажмите **Пометить как нерабочую** или **Пометить как рабочую**.
5. Если вы устанавливаете нерабочий статус, укажите причину поломки и нажмите **Сохранить**.
Статус конфигурации будет изменен.

4.2.3 Восстановление последней рабочей версии конфигурации

Чтобы применить последнюю рабочую версию конфигурации:

1. Откройте веб-интерфейс Angie ADC.
2. Выберите **Управление трафиком** → **Балансировщик нагрузки**.
3. В нижней части экрана щелкните многоточие (...) и выберите **Применить последнюю рабочую версию**.
Будет применена последняя рабочая конфигурация.

4.2.4 Восстановление конфигурации из резервной копии

Чтобы применить произвольную конфигурацию из резервной копии:

1. Откройте веб-интерфейс Angie ADC.
2. Выберите **Управление трафиком** → **Балансировщик нагрузки**.
3. В нижней части экрана щелкните многоточие (...) и выберите **История**.
Откроется таблица со списком сохраненных конфигураций.

- Щелкните многоточие (...) напротив конфигурации, которую вы хотите использовать, и нажмите кнопку **Применить**.

Конфигурация будет применена сразу.

4.2.5 Выбор репозитория для хранения конфигураций

В Angie ADC поддерживаются два типа репозитория для хранения *резервных копий конфигурации балансировщика нагрузки*, локальный (**filesystem**) и внешний (**S3**). Для надежного хранения данных рекомендуется использовать внешний репозиторий. Резервная копия в любом репозитории обновляется автоматически при внесении изменений и сохранении новой конфигурации балансировщика нагрузки.

Локальный репозиторий filesystem подключен по умолчанию. При использовании локального репозитория резервные копии хранятся в файловой системе внутри Angie ADC.

i Примечание

При подключении нового репозитория (локального или внешнего) резервная копия в локальном репозитории автоматически удаляется.

Внешний репозиторий подключается по протоколу S3. Мы рекомендуем использовать внешний S3-совместимый репозиторий для хранения резервных копий конфигурации. В отличие от локального репозитория, при отключении внешнего репозитория резервные копии в нем сохраняются.

Просмотр информации о текущем репозитории:

- Откройте веб-интерфейс Angie ADC. В панели навигации перейдите в раздел **Управление трафиком** → **Балансировщик нагрузки** → многоточие (...) → **История** → **Управление резервным копированием**.

Откроется страница с информацией о текущем репозитории.

Подключение локального репозитория:

- Откройте веб-интерфейс Angie ADC. В панели навигации перейдите в раздел **Управление трафиком** → **Балансировщик нагрузки** → многоточие (...) → **История** → **Управление резервным копированием**.

Откроется страница с информацией о текущем репозитории.

- Перейдите на вкладку **Filesystem**.
- Нажмите **Подключить**.

Будет подключен локальный репозиторий и создана резервная копия конфигурации балансировщика.

i Примечание

Если у вас до этого уже была сохранена резервная копия в **filesystem**, она будет удалена.

Подключение внешнего репозитория (S3):

- Откройте веб-интерфейс Angie ADC. В панели навигации перейдите в раздел **Управление трафиком** → **Балансировщик нагрузки** → многоточие (...) → **История** → **Управление резервным копированием**.

Откроется страница с информацией о текущем репозитории.

- Перейдите на вкладку **S3**.
- Заполните поля для подключения репозитория:

- Конечная точка
- Имя корзины
- Ключ доступа
- Секретный токен

4. Нажмите Подключить.

Если вы подключаете репозиторий, в котором уже была резервная копия конфигурации, то будет использована эта копия. Если подключается новый репозиторий, будет создана резервная копия текущей конфигурации балансировщика.

Примечание

Если у вас до этого уже была сохранена резервная копия в `filesystem`, она будет удалена.

4.3 Методы балансировки

Примечание

Для каждой upstream-группы можно задать только один метод балансировки.

Алгоритм	Описание	HTTP TCP/UDP
round-robin	Запросы распределяются по серверам последовательно (используется по умолчанию). Поддерживается учет веса серверов (<code>weight</code>), медленный старт (<code>slow_start</code>), backup-группы серверов.	
hash	Задаёт метод балансировки нагрузки для группы, при котором соответствие клиента серверу определяется при помощи хэшированного значения ключа. Поддерживается учет веса серверов (<code>weight</code>). Подробнее см. TCP/UDP , HTTP .	
ip_hash	Сервер выбирается с помощью хэш-функции на основе IP-адреса клиента, что обеспечивает "липкость сессии". Частный случай <code>hash</code> . В качестве ключа для хэширования используются первые три октета IPv4-адреса клиента или IPv6-адрес клиента целиком. В итоге запрос конкретного клиента всегда попадает на один и тот же сервер, если он работоспособен. Поддерживается учет веса серверов (<code>weight</code>). Подробнее см. HTTP .	
least_conn	Новый запрос направляется на сервер с минимальным количеством активных соединений (наименее загруженный). Поддерживается учет веса серверов (<code>weight</code>), медленный старт (<code>slow_start</code>), backup-группы серверов. Подробнее см. TCP/UDP , HTTP .	
random	Запросы распределяются по серверам случайным образом. При большом числе запросов они будут распределяться по upstream-серверам равномерно, что хорошо подходит, например, для кластера балансировщиков. Поддерживается учет веса серверов (<code>weight</code>) и опциональный параметр <code>two</code> (выбор из двух случайных серверов по <code>least_conn</code>). Подробнее см. TCP/UDP , HTTP .	
least_time	Балансировка нагрузки на основе наименьшего времени ответа. Запросы распределяются на те серверы, которые обрабатывают их быстрее. Уменьшает время ожидания для пользователей. Если серверы имеют разное оборудование или настройки, помогает уравновесить нагрузку. Поддерживаются backup-группы серверов. Подробнее см. TCP/UDP , HTTP .	
feedback	Балансировка нагрузки по обратной связи (по произвольному параметру из ответа на основной или проверочный запрос). Поддерживается учет веса серверов (<code>weight</code>), медленный старт (<code>slow_start</code>), backup-группы серверов. Подробнее см. TCP/UDP , HTTP .	
least_bandwidth	Балансировка на основе наименьшего потребления полосы пропускания. Режимы: <code>upstream</code> , <code>downstream</code> , <code>both</code> . Поддерживается учет веса серверов (<code>weight</code>), медленный старт (<code>slow_start</code>), backup-группы серверов. Подробнее см. TCP/UDP , HTTP .	
least_packages	Балансировка на основе наименьшего числа пакетов в единицу времени. Может использоваться в случаях, когда оборудование, расположенное за балансировщиком, не способно "переварить" слишком большой поток пакетов. Режимы: <code>upstream</code> , <code>downstream</code> , <code>both</code> . Поддерживается учет веса серверов (<code>weight</code>), медленный старт (<code>slow_start</code>), backup-группы серверов. Подробнее см. TCP/UDP .	

4.3.1 Примеры HTTP-балансировки нагрузки

Базовая настройка

Самая простая конфигурация для балансировки нагрузки с помощью Angie ADC может выглядеть следующим образом:

```
http {
    upstream myapp1 {
        server srv1.example.com;
        server srv2.example.com;
```



```

        server srv3.example.com;
    }
    server {

        listen 80;
        location / {

            proxy_pass http://myapp1;
        }
    }
}

```

В приведенном примере:

- Группа серверов `myapp1` содержит три сервера `srv1`, `srv2` и `srv3`.
- На этих серверах работают три экземпляра одного и того же приложения.
- Используется метод балансировки `round-robin` (по умолчанию).
- Все запросы на порт 80 перенаправляются через группу серверов `myapp1`.

`least_conn`

Балансировка нагрузки по наименее загруженному серверу.

Модуль балансировки нагрузки `least_conn` в модуле HTTP позволяет распределять нагрузку более равномерно, направляя новые запросы на серверы с наименьшим количеством активных соединений. Этот метод рекомендуется, если время обработки запросов может сильно варьироваться.

Пример конфигурации:

```

http {

    upstream backend {

        least_conn;

        server backend1.example.com;
        server backend2.example.com;
        server backend3.example.com;
    }

    server {

        listen 80;
        location / {

            proxy_pass http://backend;
        }
    }
}

```

В этом примере:

- Директива `least_conn` включает метод балансировки нагрузки по наименьшему числу соединений.
- Для балансировки нагрузки определены три проксируемых сервера.
- Входящие запросы к серверу проксируются в группу `backend`.

Преимущества метода:

- Динамическая балансировка: непрерывно отслеживает количество активных соединений для адаптации к текущей нагрузке на каждом сервере.
- Простота: работает без необходимости дополнительных параметров или сложных настроек.
- Совместимость: совместим с другими конфигурациями проксируемых серверов, включая веса, максимальное количество сбоев и время ожидания после сбоя.

ip_hash

Балансировка нагрузки с сохранением сессий.

Методы `round-robin` и `least-conn` не гарантируют, что запросы от одного клиента будут обрабатываться одним и тем же сервером. Если важно, чтобы клиент всегда направлялся на один сервер (например, при использовании сессионных данных), можно использовать метод `ip_hash`.

Пример конфигурации:

```
upstream myapp1 {
    ip_hash;
    server srv1.example.com;
    server srv2.example.com;
    server srv3.example.com;
}
```

Метод `ip_hash` гарантирует, что запросы от одного клиента будут попадать на один сервер, если этот сервер доступен.

weight

Взвешенная балансировка нагрузки.

С помощью директивы `weight` можно указать, какой сервер должен обрабатывать больше запросов. Этот способ рекомендуется, если серверы имеют разную производительность. По умолчанию вес сервера равен 1.

Пример:

```
upstream myapp1 {
    server srv1.example.com weight=3;
    server srv2.example.com;
    server srv3.example.com;
}
```

В этой конфигурации каждые пять запросов будут распределены следующим образом: три запроса попадут на `srv1`, один запрос — на `srv2` и еще один — на `srv3`.

Метод `weight` может использоваться в комбинации с методами `round-robin` и `least-conn`.

least_time

Балансировка нагрузки на основе наименьшего времени ответа.

Модуль балансировки нагрузки `least_time` задает для группы метод балансировки нагрузки, при котором вероятность передачи соединения активному серверу обратно пропорциональна среднему времени его ответа; чем оно меньше, тем больше соединений будет получать сервер.

Когда использовать `least_time`?

- Переменная нагрузка: запросы распределяются на серверы, которые обрабатывают их быстрее.
- Высокая доступность: уменьшается время ожидания для пользователей.

- Гетерогенные серверы: если серверы имеют разное оборудование или настройки, `least_time` помогает уравновесить нагрузку.

Пример конфигурации:

```
http {
    upstream backend {
        least_time header;
        server backend1.example.com;
        server backend2.example.com;
        server backend3.example.com;
    }
    server {
        listen 80;
        location / {
            proxy_pass http://backend;
        }
    }
}
```

В этом примере `least_time header`: балансировка основана на времени отклика до получения заголовка ответа от сервера.

feedback

Балансировка по произвольному параметру из ответа на основной или проверочный запрос.

Модуль балансировки нагрузки **feedback** задает в `upstream` механизм балансировки нагрузки по обратной связи. Он динамически корректирует решения при балансировке, умножая вес каждого проксируемого сервера на среднее значение обратной связи, которое меняется с течением времени в зависимости от значения переменной и подчиняется необязательному условию. По умолчанию параметр `factor` равен 90.

Пример конфигурации:

```
upstream backend {
    zone backend 1m;

    feedback $feedback_value factor=80 account=$condition_value;

    server backend1.example.com;
    server backend2.example.com;
}

map $upstream_http_custom_score $feedback_value {
    "high" 100;
    "medium" 75;
    "low" 50;
    default 10;
}

map $upstream_probe $condition_value {
    "high_priority" "1";
}
```

```
"low_priority" "0";
default "1";
}
```

Эта конфигурация категоризирует ответы серверов по уровням обратной связи на основе определенных оценок из полей заголовков ответа, а также добавляет условие на `$upstream_probe`, чтобы учитывать только ответы от активной проверки `high_priority` или ответы на обычные клиентские запросы.

4.3.2 Примеры TCP/UDP-балансировки нагрузки

Конфигурация для TCP

```
stream {

    upstream tcp_backend {

        server 192.168.1.10:3306 weight=2;
        server 192.168.1.11:3306;
        server 192.168.1.12:3306;
    }

    server {

        listen 3306;
        proxy_pass tcp_backend;
        proxy_timeout 10s;
        proxy_connect_timeout 5s;
    }
}
```

В этой конфигурации задана группа серверов, участвующих в балансировке (`tcp_backend`). Настроен вес для серверов, чтобы более мощный сервер получал больше запросов (`weight=2`), Сервер слушает входящие соединения на порту 3306 (используется для MySQL). Трафик перенаправляется к группе серверов `tcp_backend`. Настроены тайм-ауты:

- максимальное время ожидания ответа от сервера 10s;
- максимальное время для установления соединения с сервером 5s.

Конфигурация для UDP

Для приложений, работающих через протокол UDP, конфигурация аналогична TCP, но с настройками для UDP-трафика.

```
stream {

    upstream udp_backend {

        server 192.168.1.20:53;
        server 192.168.1.21:53;
    }

    server {

        listen 53 udp;
        proxy_pass udp_backend;

        proxy_responses 1;
    }
}
```

```

        proxy_timeout 10s;
    }
}

```

least_bandwidth

Балансировка на основе наименьшего потребления полосы пропускания.

Алгоритм периодически вычисляет среднее использование полосы пропускания для каждого апстрим-сервера, используя формулу скользящего среднего для сглаживания колебаний со временем. Когда начинается новая сессия, модуль балансировки нагрузки **least_bandwidth** выбирает проксируемый сервер с наименьшим средним использованием пропускной способности, скорректированным с учетом веса сервера, если он указан. Этот механизм гарантирует, что сессии направляются на серверы, которые в настоящее время используют меньше пропускной способности, что способствует равномерному распределению сетевой нагрузки.

Пример конфигурации:

```

stream {

    upstream backend {

        least_bandwidth both factor=80;
        server backend1.example.com:12345;
        server backend2.example.com:12345;
    }
    server {

        listen 12345;
        proxy_pass backend;
    }
}

```

В этом примере:

- В директиве **least_bandwidth** настроен учет как входящей, так и исходящей пропускной способности.
- Коэффициент сглаживания установлен равным 80, что влияет на скорость адаптации среднего значения к изменениям в использовании пропускной способности.
- Для балансировки нагрузки определены два проксируемых сервера.

Преимущества метода:

- Динамическая балансировка: непрерывно отслеживает использование пропускной способности для адаптации к текущей нагрузке на каждом сервере.
- Коэффициент сглаживания **factor** контролирует отзывчивость расчета скользящего среднего; более высокий коэффициент означает более медленную адаптацию к изменениям.
- Режимы: позволяет выбирать балансировку на основе входящей пропускной способности, исходящей пропускной способности или обоих вариантов, в зависимости от потребностей приложения.
- Совместим с другими конфигурациями проксируемых серверов, включая веса, максимальное количество сбоев и время ожидания после сбоя.

least_packets

Балансировка на основе наименьшего числа пакетов в единицу времени.

Средняя скорость пакетов для каждого проксируемого сервера периодически проверяется с использованием формулы скользящего среднего для сглаживания колебаний со временем. Когда инициируется новый сеанс, модуль балансировки нагрузки `least_packets` в потоковом модуле выбирает проксируемый сервер с наименьшей средней скоростью передачи пакетов, скорректированной с учетом веса сервера. Этот процесс гарантирует, что сеансы направляются на серверы, которые в настоящее время наименее загружены с точки зрения обработки пакетов, что способствует равномерному распределению сетевой нагрузки.

Пример конфигурации:

```
stream {
    upstream backend {
        least_packets both factor=80;
        server backend1.example.com:12345;
        server backend2.example.com:12345;
    }
    server {
        listen 12345;
        proxy_pass backend;
    }
}
```

В этом примере:

- В директиве `least_packets` настроен учет как входящих, так и исходящих пакетов.
- Коэффициент сглаживания установлен равным 80, что влияет на то, как быстро среднее значение адаптируется к изменениям скорости пакетов.
- Определены два проксируемых сервера для балансировки нагрузки.

Преимущества метода:

- Динамическая балансировка: непрерывно отслеживает скорость пакетов для адаптации к текущей нагрузке на каждом сервере.
- Параметр `factor` контролирует отзывчивость расчета скользящего среднего; более высокий коэффициент означает более медленную адаптацию к изменениям.
- Режимы: позволяет выбирать балансировку на основе числа входящих пакетов, исходящих пакетов или обоих значений.
- Совместим с другими конфигурациями проксируемых серверов, включая веса, максимальное количество сбоев и время ожидания после сбоя.

hash

Контекстное хэширование.

Поддерживается метод `hash` (например, по IP или произвольному значению). Если сервер из пула станет недоступным или будет добавлен новый сервер, минимальное количество клиентов перенаправится на другие серверы. Используется для "липких сессий", когда важно, чтобы запросы от одного клиента всегда обрабатывались одним и тем же сервером.

Пример:

```
upstream tcp_backend {
```

```
hash $remote_addr consistent;
server 192.168.1.10:3306;
server 192.168.1.11:3306;
}
```

Ограничение IP

Для защиты можно использовать ограничения IP через `allow` и `deny`.

Пример:

```
server {

    listen 3306;
    allow 192.168.1.0/24;
    deny all;
    proxy_pass tcp_backend;
}
```

В этом примере разрешена только локальная сеть, остальным доступ запрещен.

4.4 Предоставление доступов на работу с конфигурацией

angie-adc1 > Балансировщик > local-lb > Конфигурация > Управление доступами

Управление доступами

Добавить доступ

Тип ресурса: Файл конфигурации: Пользователь: Доступ:

Текущие доступы:

Ресурс	Пользователь	Доступ	Кто выдал	Дата
Файл: http://my.conf	user	Просмотр	admin	28.07.2026

Всем пользователям Angie ADC (локальным и внешним) назначаются роли в системе, определяющие их доступ к конфигурации балансировщика нагрузки.

- **Администратор** обладает полными правами на все действия в системе, включая редактирование всех файлов конфигурации балансировщика нагрузки Angie ADC.
- **Супервизор** имеет ограниченные права: может только просматривать информацию в системе, например проводить аудит, в том числе может просматривать все файлы конфигурации балансировщика нагрузки.
- **Оператор** имеет ограниченные права: может редактировать файлы конфигурации балансировщика, к которым ему предоставлен доступ, и просматривать статистику зон, к которым ему предоставлен доступ. Доступы предоставляет администратор.

4.4.1 Предоставление прав на просмотр и редактирование

Чтобы *оператор мог настраивать балансировку для сервера*, ему необходимо разрешить редактирование файла конфигурации, в котором описан этот сервер. Права на просмотр и редактирование файлов предоставляет администратор через веб-интерфейс Angie ADC.

Примечание

Рекомендуется давать операторам доступ только к файлам в папках `http.d` и `stream.d`.

1. В веб-интерфейсе Angie ADC перейдите в раздел **Управление трафиком** → **Балансировщик нагрузки** → **Управление доступами**.
2. В открывшемся окне заполните поля формы для добавления нового доступа:
 - Чтобы предоставить доступ к файлу, выберите типа ресурса **Файл** и в выпадающем списке **Файл конфигурации** выберите нужный файл. Затем в поле **Доступ** укажите уровень доступа: **Просмотр** или **Редактирование**.
 - Чтобы предоставить доступ к зоне, выберите типа ресурса **Зона** и введите имя зоны в поле **Зона**. Для типа ресурса **Зона** всегда выдается доступ на редактирование.

Примечание

Зону предварительно необходимо настроить.

3. В поле **Пользователь** укажите логин пользователя, для которого настраивается доступ.
4. Нажмите **Добавить**.

Доступ будет добавлен и отобразится в таблице доступов.

4.4.2 Просмотр выданных доступов

1. В веб-интерфейсе Angie ADC перейдите в раздел **Управление трафиком** → **Балансировщик нагрузки** → **Управление доступами**.
2. В открывшемся окне в таблице **Текущие доступы** отобразится список выданных доступов.
Для удобства поиска поддерживается фильтрация.

4.4.3 Редактирование выданных доступов

Редактирование записей не поддерживается.

Чтобы изменить запись, необходимо удалить старую запись и добавить новую с измененными параметрами.

4.4.4 Удаление выданных доступов

1. В веб-интерфейсе Angie ADC перейдите в раздел **Управление трафиком** → **Балансировщик нагрузки** → **Управление доступами**.
2. В открывшемся окне в таблице **Текущие записи** отобразится список выданных доступов.
3. Нажмите на значок удаления напротив нужной записи.

Доступ будет удален.

4.5 Проверки работоспособности серверов

Angie ADC включает пассивные и активные проверки работоспособности серверов (health probes). Если сервер начинает отвечать с ошибками, Angie ADC временно исключает его из пула доступных серверов.

4.5.1 Пассивные проверки

Пассивные проверки работают автоматически на основе реальных клиентских запросов.

Пример:

```
upstream myapp1 {

    server srv1.example.com max_fails=2 fail_timeout=10s;
    server srv2.example.com;
    server srv3.example.com;
}
```

В этой конфигурации сервер помечается как недоступный после двух неудачных запросов подряд (`max_fails=2`). Через десять секунд сервер снова проверяется на доступность (`fail_timeout=10s`). Если он восстановился, запросы снова начинают отправляться на этот сервер.

4.5.2 Активные проверки

Активные проверки позволяют оценить доступность сервера без участия клиентов. Поддерживаются активные проверки для HTTP и TCP / UDP. Проверки задаются с помощью директивы `upstream_probe`. Для одного апстрима можно определить несколько проверок. Можно гибко настраивать поведение проверок. Поддерживается отправка ICMP echo-запросов (ping). Подробнее см. справочные материалы по активным проверкам [HTTP](#) и [stream](#).

4.5.2.1 Пример для HTTP

```
upstream backend {
    zone backend 1m;

    server backend1.example.com;
    server backend2.example.com;
}

map $upstream_status $good {
    200      "1";
}

server {
    listen ...;

    location /backend {
        ...
        proxy_pass http://backend;

        upstream_probe backend_probe
            uri=/probe
            port=10004
            interval=5s
            test=$good
            essential
            persistent
            fails=3
            passes=3
            max_body=10m
            mode=idle;
    }
}
```

Детали работы:

- Изначально сервер не получает клиентские запросы, пока не пройдет *все* заданные для него проверки с параметром **essential** (пропуская помеченные как **persistent**, если конфигурация перезагружена и до этого сервер считался работающим). Если таких проверок нет, сервер считается работающим.
- Сервер считается неработающим и не получает клиентские запросы, если *какая-либо* заданная для него проверка достигает своего порога **fails** или сам сервер достигает порога *max_fails*.
- Чтобы неработающий сервер снова мог считаться работающим, *все* заданные для него проверки должны достичь своего порога **passes**; после этого учитывается порог *max_fails*.

4.5.2.2 Пример для TCP/UDP

```
upstream backend {
    zone backend 1m;

    server a.example.com;
    server b.example.com;
}

map $upstream_probe_response $good {
    ~200    "1";
    default  "";
}

server {
    listen ...;

    # ...
    proxy_pass backend;
    upstream_probe_timeout 1s;

    upstream_probe backend_probe
        port=12345
        interval=5s
        test=$good
        essential
        persistent
        fails=3
        passes=3
        max_response=512k
        mode=onfail
        "send=data:GET / HTTP/1.0\r\n\r\n";
}
```

i Примечание

Согласно спецификации RFC 2616 (HTTP/1.1) и RFC 9110 (HTTP Semantics), заголовки HTTP должны разделяться последовательностью CRLF (`\r\n`), а не просто `\n`.

Детали работы:

- Изначально сервер не получает клиентские запросы, пока не пройдет *все* заданные для него проверки с параметром **essential** (пропуская помеченные как **persistent**, если конфигурация перезагружена и до этого сервер считался работающим). Если таких проверок нет, сервер считается работающим.
- Сервер считается неработающим и не получает клиентские запросы, если *какая-либо* заданная для него проверка достигает своего порога **fails** или сам сервер достигает порога *max_fails*.

- Чтобы неработающий сервер снова мог считаться работающим, все заданные для него проверки должны достичь своего порога `passes`; после этого учитывается порог `max_fails`.
- Параметр `send` - отправляемые для проверки данные, а переменная `$upstream_probe_response` - ответ бекенда.

4.5.2.3 Пример ICMP echo-запроса для HTTP

```
http {
    upstream u {
        zone z 1k;

        server 127.0.0.1:8081;
        server 127.0.0.1:8082;
    }

    server {
        listen ...;

        location / {
            proxy_pass http://u/;

            upstream_probe simple1
                interval=3s
                ping;

            upstream_probe simple2
                interval=3s
                ping ping_timeout=3s;
        }
    }
}
```

Детали работы:

- Используется проверка методом ICMP Echo вместо HTTP-запроса.
- ICMP-проверка проверяет доступность IP-адреса 127.0.0.1. Порты `upstream` игнорируются.
- Сервер считается неработающим и не получает клиентские запросы, если не отвечает на запрос ICMP Echo в течение заданного таймута. Первая проверка использует таймаут ожидания по умолчанию (1 секунда). Вторая проверка задает таймаут ожидания ответа явно (3 секунды).
- Единственным критерием прохождения проверки является успешное получение ICMP-ответа в пределах таймута.

4.6 Резервирование проксируемых серверов и групп

Для повышения отказоустойчивости системы вы можете дополнительно резервировать проксируемые серверы, чтобы они включались в работу при выходе основных серверов из строя.

В Angie ADC доступно два механизма резервирования:

- *Запасные серверы* (spare-серверы).

Запасные серверы (spare-серверы) включаются в работу при выходе из строя основных серверов в рамках одной группы. Это позволяет плавно подхватывать трафик без переключения на другую группу серверов. Запасные серверы можно использовать для поддержания производительности системы, например при выводе отдельных серверов на техническое обслуживание.

- *Резервные группы серверов* (backup-группы).

Резервные группы (backup-группы) серверов включаются в работу при отказе всех серверов активной группы и поддерживают отказоустойчивость системы при выходе из строя групп серверов целиком. С помощью backup-групп можно реализовать многоуровневое резервирование большой инфраструктуры.

Можно использовать оба механизма одновременно для обеспечения максимальной надежности.

Примечание

Отказоустойчивость самого балансировщика нагрузки обеспечивается отдельно, например с помощью *пары высокой доступности*.

4.6.1 Запасные серверы (spare-серверы)

При настройке апстрим-групп часть серверов в группе можно выделить как запасные, назначив им *параметр* `spare` в блоке `server`. Также можно объявить серверы запасными динамически с помощью API-интерфейса.

Пример:

```
http {
    upstream u {
        zone z 1m;
        server 127.0.0.1:8081 max_fails=1 fail_timeout=2s weight=2;
        server 127.0.0.1:8082 spare;
        server 127.0.0.1:8083 spare;
    }
}
```

По умолчанию запасные (spare) серверы находятся в режиме ожидания и имеют статус `idle`. В случае выхода из строя одного или нескольких основных серверов группы они включаются в балансировку и переходят в статус `up`. При восстановлении основных серверов замещающие spare-серверы снова переводятся в режим ожидания. При этом поддерживаются sticky-сессии: запасные серверы, выведенные из активного пула, будут по-прежнему использоваться для таких сессий. Если на запасном сервере настроены активные проверки работоспособности, то сервер не будет включаться в работу, пока проверки не будут пройдены. Неработоспособные запасные серверы будут иметь такие же статусы, как неработоспособные основные.

Решение о вводе spare-сервера в балансировку определяется разницей между совокупным весом всех серверов, используемых в группе в качестве основных `main_weight`, и совокупным весом всех активных серверов `active_weight`. Если общий вес активных серверов становится меньше веса всех основных серверов — spare-серверы вводятся в работу, если больше — spare-серверы выводятся в режим ожидания. При этом всегда используется минимально необходимое количество spare-серверов. При введении дополнительного spare-сервера будет выбран сервер с максимальным весом, а при выведении из строя лишних spare-серверов сначала будет выбран сервер с минимальным весом.

В режиме отладки через REST API доступны значения счетчиков `main_weight` и `active_weight`. Для upstream-сервера отображается поле `spare` со значением `idle` (сервер не используется) или `running` (сервер активен). Поле `spare_num` отображает количество серверов, считающихся запасными.

4.6.2 Резервные группы серверов (backup-группы)

Резервные группы серверов (backup-группы), в отличие от spare-серверов, представляют собой отдельные группы серверов, выделенные только для резервирования. Upstream-сервер может входить только в одну резервную группу.

Если балансировщик не может найти ни один рабочий сервер в основной группе, то он переходит к серверам backup-группы. Можно добавить одну или несколько backup-групп серверов с разным

уровнем резервирования. Тогда балансировщик будет сначала переходить к backup-группе с наименьшим уровнем. Если групп много, можно указать для них произвольные уровни, например для удобства сортировки.

Уровень резервирования backup-группы можно задать с помощью директивы `backup=n`:

- `backup=1` (или просто `backup`) — группа резервных серверов первого уровня (в API отображается как `true`);
- `backup=2` — группа резервных серверов второго уровня (в API отображается как соответствующее число);
- `backup=3` — группа резервных серверов третьего уровня.

Пример:

```
upstream my_backend {
    server primary1.example.com;
    server primary2.example.com;

    server backend.example.com service=http resolve backup=prio;

    server backup1a.example.com backup;
    server backup1b.example.com backup;

    server backup2a.example.com backup=2;
    server backup2b.example.com backup=2;

    backup_switch permanent=2m;
}
```

Если upstream-сервер настраивался с помощью SRV-записи, то уровень backup-группы можно определить динамически через поле приоритета (`priority`). Для этого используется:

- Абсолютное значение приоритета (директива `backup=prio`).
В этом случае значение приоритета `priority=n` из SRV-записи будет использовано в качестве уровня резервной группы `backup=n`.
- Относительное значение приоритета `priority=n` из SRV-записи.
В этом случае директива `backup=n` задает базовый уровень отсчета (например `backup=3`), а относительные уровни резервирования отсчитываются от этого уровня. Сервер с наименьшим значением `priority` получит уровень `backup=3`, следующий за ним — `backup=4`, и так далее. Если уровень отсчета `backup=n` не задан, отсчет будет идти от нуля: сервер с наименьшим значением `priority` получит `backup=0` и станет основным, за ним — `backup=1`, резервный, и так далее.

По умолчанию балансировщик всегда начинает поиск подходящего сервера с основной группы, даже если в данный момент активна резервная. Чтобы этого избежать, можно использовать директиву `backup_switch permanent`, задав для нее значение, например `permanent=2m`. В этом случае балансировщик нагрузки будет пытаться вернуться на основную группу не чаще, чем раз в две минуты, что позволит избежать избыточных переключений группы.

Если указать параметр `permanent` без значения, то выбранная backup-группа будет сохраняться в качестве активной даже после того, как серверы из основной группы снова станут доступными. В случае отказа текущей backup-группы балансировщик будет переключаться на следующую по уровню backup-группу, и, только дойдя до максимального уровня, вернется на основную группу.

4.7 Загрузка и раздача статических файлов

Angie ADC поддерживает загрузку и раздачу статических файлов (изображений, статических HTML-страниц и пр.). Загрузка осуществляется по протоколу SCP (Secure Copy Protocol), после чего файл будет доступен по HTTP.

Чтобы добавить статический файл, выполните следующие действия:

1. На машине-источнике запустите загрузку файла (например страницу-заглушку `maintenance.html`):

```
scp -P 2222 maintenance.html admin@<имя_хоста>:/maintenance.html
```

2. В веб-интерфейсе Angie ADC перейдите к редактированию конфигурации балансировщика нагрузки (Управление трафиком → Балансировщик нагрузки) и в блоке `http` настройте доступ к каталогу с этим файлом:

```
http {
    server {
        listen 80;
        root /var/transfer;
    }
}
```

3. Если для расширения файла не определен MIME-тип или требуется указать его явно, добавьте соответствующую строку в блок `types` (например `application/pkix-crl crl;` для CRL).
4. Сохраните изменения, нажав кнопку **Сохранить**.

Файл `maintenance.html` будет доступен по адресу `http://<имя_хоста>/maintenance.html`.

Чтобы проверить, что файл корректно отдается клиентам, выполните запрос `curl`. Например, для хоста `adc.angie.local`:

```
$ curl adc.angie.local/maintenance.html -v
```

Если получен код 200 OK и указан правильный Content-Type, то файл был успешно загружен и доступен.

4.7.1 Раздача файлов из разных локальных каталогов

В зависимости от типа файлы могут раздаваться из разных каталогов, например HTML-страницы из `/static/www`, а изображения из `/static/images`. Для настройки используется директива `location`.

Чтобы настроить раздачу файлов из разных локальных каталогов, выполните следующие действия:

1. Создайте локальный каталог со всеми необходимыми подкаталогами.

Например:

```
mkdir -p static/www static/images
```

2. Перенесите в этот каталог статические файлы. Например, поместите файл `index.html` в `/static/www`, а изображение `image1.png` в `/static/images`.
3. Скопируйте весь каталог в файловое хранилище Angie ADC по SCP:

```
scp -P 2222 -r static admin@<имя_хоста>:/
```

4. В веб-интерфейсе Angie ADC перейдите к редактированию конфигурации балансировщика нагрузки (Управление трафиком → Балансировщик нагрузки) и в блоке `http` настройте доступ к каталогам со статическими файлами:

```
server {
    location / {
        root /static/www;
    }

    location /images/ {
        root /static;
    }
}
```

5. Сохраните изменения, нажав кнопку **Сохранить**.

Теперь запросы, URI которых начинаются на `/images/`, будут обрабатываться в `/static/images`. Другие запросы будут обрабатываться в `/static/www`. Если файл не найден, отобразится ошибка 404.

Например, на запрос `http://<имя_хоста>/images/image1.png` будет отдан файл `/static/images/image1.png`, а на запрос `http://<имя_хоста>/files/index.html` будет отдан файл `/static/www/files/index.html`.

4.8 Аутентификация клиентов

angie-adc1 > Аутентификация клиентов

Аутентификация клиентов

Настройка порталов для аутентификации клиентов.

Порталы

[+ Добавить](#)

Порталы ещё не настроены.

Добавьте портал, чтобы получить URL для проверки сессии и входа клиентов.

Новый портал

[Сбросить](#)

Имя		customer-a-ldap
Пути		
URL проверки сессии : /auth/verify?provider=customer-a-ldap		Копировать
URL входа : /auth/login?provider=customer-a-ldap		Копировать
Провайдер RADIUS TACACS LDAP		
Параметры подключения LDAP-сервера		
Сервер	Порт	DN сервисной учетной записи
FQDN или IPv4/IPv6	389	cn=svc-ldap,dc=example,dc=org
Режим защиты соединения Без TLS LDAPS StartTLS		
☒ Проверить TLS сертификат		
Поиск пользователя		
Базовый DN пользователя	Идентификатор логина	
dc=example,dc=org	uid	
Фильтр поиска (&(objectClass=person)(uid=%s))		
+ Добавить портал		

Angie ADC позволяет настроить аутентификацию клиентов при обращении к сервисам через балансировщик нагрузки. При доступе к защищенным ресурсам клиент будет попадать на страницу логина (портал аутентификации):

Вход

Провайдер: customer-a-ldap

Логин

Пароль

Войти

При успешной аутентификации клиент будет перенаправлен на запрашиваемую страницу. Для аутентификации используется внешний сервер (LDAP, TACACS+ или RADIUS). Настройки аутентификации клиентов распространяются на все узлы *кластера*.

4.8.1 Настройка аутентификации клиентов

1. В веб-интерфейсе Angie ADC перейдите в раздел **Управление трафиком** → **Аутентификация клиентов**.
2. В открывшемся окне нажмите **Добавить** и укажите имя нового портала для аутентификации. Имя должно быть уникальным и валидным для использования в URL: нижний регистр, начинается с буквы; допускаются буквы, цифры и дефисы, не допускаются пробелы и специальные символы.

В блоке Пути отобразятся URL проверки сессии и URL входа. Эти пути необходимо в дальнейшем указать в конфигурации балансировщика нагрузки (см. пункт 5).

3. Выберите тип провайдера, LDAP, TACACS+ или RADIUS, и задайте параметры подключения к серверу:

- Для LDAP:
 - Сервер: IP-адрес или доменное имя LDAP-сервера.
 - Порт: порт подключения к LDAP-серверу (по умолчанию 389).
 - Режим защиты соединения (по умолчанию Без TLS, также доступны варианты LDAPS и StartTLS).
 - DN сервисной учетной записи: логин сервисной учетной записи, от имени которой система подключается к LDAP.
 - Пароль сервисной учетной записи: пароль сервисной учетной записи для подключения к LDAP-серверу.
 - Базовый DN пользователя: задает ветку на LDAP-сервере, в которой система будет искать пользователя, например `ou=users,dc=example,dc=com`.
 - Идентификатор логина: поле в LDAP, которое содержит логин, например `uid`, `sAMAccountName` или `userPrincipalName`.
 - Фильтр поиска: LDAP-фильтр для поиска пользователя, например `(&(objectClass=user)(sAMAccountName=%s))`.

Примечание

Фильтр LDAP определяет, в каком формате пользователь должен вводить логин при входе в систему:

- если используется фильтр `(&(objectClass=user)(sAMAccountName=%s))`, то логин будет без доменного имени, например `ivanov`;

– если используется фильтр `(&(objectClass=user)(userPrincipalName=%s))`, то логин будет полным: `ivanov@domain.com`.

- Для RADIUS:
 - Сервер: IP-адрес или доменное имя RADIUS-сервера.
 - Порт: порт подключения к RADIUS-серверу (по умолчанию 1812).
 - Протокол: протокол передачи данных: UDP или TCP (по умолчанию: UDP).
 - NAS-идентификатор: идентификатор NAS (Network Access Server), передаваемый серверу RADIUS.
 - Метод аутентификации: поддерживается только PAP.
 - Секрет: общий секрет для шифрования паролей.
 - Для TACACS+:
 - Адрес: IP-адрес или доменное имя TACACS-сервера.
 - Порт: порт подключения к TACACS-серверу (по умолчанию 49).
 - Секрет: общий секрет для шифрования паролей.
4. Нажмите **Добавить портал**.
 5. Перейдите в раздел **Управление трафиком** → **Балансировщик нагрузки** и укажите пути портала аутентификации для нужного location, а также страницу ошибки в случае непрохождения аутентификации.

Например:

```
include /etc/angie-lb/internal/adc-ui-portal-locations.inc;

location /app/ {
    auth_request /internal-verify;
    error_page 401 = @auth_login;
    proxy_pass http://<backend-address>;
    proxy_set_header Host                $host;
    proxy_set_header X-Forwarded-Proto <http или https>;
}

location = /internal-verify {
    internal;
    proxy_set_header X-Angie-Auth-Request 1;
    proxy_pass http://unix:/run/angie-adc-auth-dataplane/auth-dataplane.sock:/
    →auth/verify?provider=<portal-name>;
}

location @auth_login {
    return 302 /auth/portal/login?provider=<portal-name>&redirect_url=$request_
    →uri;
}
```

4.8.2 Изменение параметров портала аутентификации

1. В веб-интерфейсе Angie ADC перейдите в раздел **Управление трафиком** → **Аутентификация клиентов**.
2. В открывшемся окне в списке порталов выберите портал, параметры которого вы хотите изменить.

- Внесите изменения и нажмите **Сохранить** портал.

Важно

Если портал использовался для аутентификации в конфигурации балансировщика нагрузки, не забудьте отредактировать конфигурацию соответствующим образом.

4.8.3 Удаление портала аутентификации

- В веб-интерфейсе Angie ADC перейдите в раздел **Управление трафиком** → **Аутентификация клиентов**.
- В открывшемся окне в списке порталов выберите портал, который вы хотите удалить, и нажмите **Удалить**.

Портал будет удален.

Важно

Если портал использовался для аутентификации в конфигурации балансировщика нагрузки, не забудьте удалить его из конфигурации, иначе пользователи будут по-прежнему перенаправляться на страницу аутентификации, но не смогут ее пройти, т.к. портал удален.

4.9 Делегирование сервиса оператору

Angie ADC поддерживает передачу управления выбранными сервисами оператору. Для этого администратор создает отдельные файлы конфигурации, в которых будет настраиваться балансировка сервиса, и выдает права на доступ к этим файлам оператору. Администратор также подключает эти файлы в основной конфигурационный файл **angie.conf** с помощью директивы **include**. Оператор, занимающийся обслуживанием сервиса, редактирует конфигурацию балансировщика нагрузки в выделенных для него файлах. При этом он не видит другие файлы конфигурации, т.к. не имеет к ним доступа. При сохранении любого файла, включенного по **include**, вся конфигурация балансировщика валидируется и применяется. Также можно настроить доступ оператора к данным мониторинга сервиса.

Важно

Включаемые файлы должны содержать синтаксически верные директивы и блоки, иначе итоговая конфигурация не пройдет валидацию и не будет применена.

4.9.1 Настройка со стороны администратора

Ниже приведен пример делегирования оператору управления сервисом **example.com**.

- В веб-интерфейсе Angie ADC перейдите в раздел **Управление трафиком** → **Балансировщик нагрузки**.
- Создайте файлы **example_upstream.conf** и **example_server.conf** в папке **http.d**.
- В конфигурации **angie.conf** создайте блок, управляющий сервисом **example.com**, например:

```
http {
    upstream example_upstream {
        zone example_zone 1m;
        include "http.d/example_upstream.conf";
    }
}
```

```
server {
    listen      VIP:80;
    server_name example.com;
    status_zone example_zone;

    location / {
        proxy_pass "http://example_upstream";
        include "http.d/example_server.conf";
    }
}
```

Здесь:

- Создана группа серверов **example_upstream** для балансировки нагрузки.
 - Выделена *зона* разделяемой памяти **example_zone**. В ней хранится текущее состояние серверов, счетчики запросов и сессии.
 - С помощью директивы *include* подключен внешний файл **example_upstream.conf**, в котором оператор сможет указать IP-адреса или домены бэкенд-серверов.
 - Описаны правила обработки запросов.
 - Метрики для вывода статистики (*status_zone*) будут собираться в уже заданную зону разделяемой памяти **example_zone**.
 - Все входящие запросы будут перенаправляться на бэкенд-серверы, описанные в файле **example_upstream.conf**.
 - С помощью директивы *include* подключен файл **example_server.conf** с дополнительными настройками для проксирования, которые также может настраивать оператор.
4. Выдайте оператору *права* на редактирование файлов **http.d/example_upstream.conf** и **http.d/example_server.conf**.

Совет

Рекомендуется также передать оператору внутренние инструкции и стандарты компании по наполнению этих конфигурационных файлов.

5. Выдайте оператору *права доступа* к зоне разделяемой памяти **example_zone** для просмотра статистики сервиса на странице мониторинга.

Теперь в разделе **Управление трафиком** → **Балансировщик нагрузки** оператору доступны файлы **http.d/example_upstream.conf** и **http.d/example_server.conf**.

Изменения, внесенные оператором в файл, автоматически валидируются системой и и применяются после нажатия кнопки **Сохранить** без прерывания трафика. Если конфигурация не проходит валидацию, то она не применяется.

4.9.2 Пример работы оператора

1. В веб-интерфейсе Angie ADC перейдите в раздел **Управление трафиком** → **Балансировщик нагрузки**.

Откроется экран с файлами, доступ к которым у вас есть. Вы можете просматривать и редактировать эти файлы.

Важно

Отредактированные файлы должны содержать синтаксически верные директивы и блоки, иначе итоговая конфигурация балансировщика не пройдет валидацию и не будет применена.

- В файле `example_upstream.conf` укажите адреса бэкендов и алгоритм балансировки. Например:

```
least_conn;
server 192.0.2.10:8080;
server 192.0.2.11:8080;
```

- Нажмите **Сохранить**. Изменения в конфигурации будут применены сразу.
- Если необходимо, в файле `example_server.conf` добавьте дополнительные настройки для проксирования, например заголовок запроса:

```
proxy_set_header X-Custom-Header "value";
```

- Нажмите **Сохранить**. Изменения в конфигурации будут применены сразу.

4.10 Справочная информация

Справочные материалы по настройке балансировщика Angie ADC.

4.10.1 Общие сведения

В этих статьях рассказывается о различных аспектах обработки запросов и взаимодействия с другими серверами.

- *Конфигурационные файлы*
- *Соединения, сессии, запросы, логи*
- *Режим отладки*

4.10.2 Основной модуль

Core Управление служебными файлами, процессами и другими модулями Angie ADC.

4.10.3 HTTP-модули

HTTP Обработка HTTP-запросов и ответов, управление HTTP-сервером, соединениями и статическими файлами.

4.10.4 Потокные модули

Stream Обработка HTTP-запросов и ответов, управление HTTP-сервером, соединениями и статическими файлами.

4.10.5 Почтовые модули

Mail Функциональность почтового прокси-сервера.

4.10.6 Сторонние модули

<i>GeoIP2</i>	Добавляет поиск по геоданным в базах MaxMind GeoIP2.
<i>NJS:</i>	Позволяют использовать njs, подмножество языка JavaScript, в конфигурации Angie ADC соответственно в контекстах http и stream .
• <i>HTTP JS</i>	
• <i>Stream JS</i>	

4.10.7 Инструкции

- *Настройка ACME*

4.10.8 Указатель директив

Указатель директив

4.10.8.1 Конфигурационные файлы

Angie ADC использует текстовый конфигурационный файл. По умолчанию этот файл называется **angie.conf** и обычно находится в директории **/etc/angie**.

Конфигурационный файл обычно состоит из следующих контекстов:

- *events* — общая обработка соединений;
- *http* — HTTP-трафик;
- *stream* — TCP- и UDP-трафик.

Директивы, размещенные вне этих контекстов, считаются находящимися в контексте **main**:

```
user angie; # директива в контексте 'main'

events {
    # конфигурация обработки соединений
}

http {
    # Конфигурация трафика HTTP, для всех вложенных виртуальных серверов

    server {
        # конфигурация виртуального HTTP сервера 1
        location /one {
            # конфигурация обработки HTTP запросов с URI, начинающимися с '/one'
        }
        location /two {
            # конфигурация обработки HTTP запросов с URI, начинающимися с '/two'
        }
    }
}
```

```
server {
    # конфигурация виртуального HTTP сервера 2
}

stream {
    # Конфигурация трафика TCP/UDP, для всех вложенных виртуальных серверов
    server {
        # конфигурация виртуального TCP сервера 1
    }
}
```

Для упрощения управления конфигурацией рекомендуется использовать директиву *include* в основном файле `angie.conf`, чтобы ссылаться на содержимое файлов, специфичных для функций:

```
include /etc/angie/http.d/*.conf;
include /etc/angie/stream.d/*.conf;
```

Наследование

В общем случае дочерний контекст (тот, который содержится в другом контексте, который считается родительским) унаследует настройки директив, определенных на уровне родителя. Некоторые директивы могут появляться в нескольких контекстах; в таких случаях вы можете переопределить настройки, унаследованные от родителя, включив директиву в дочерний контекст.

Синтаксис

Единицы измерения

Размеры можно указывать в следующих единицах:

Без суффикса	Байты
k, K	Килобайты
m, M	Мегабайты
g, G	Гигабайты

Например: 1024, 8k, 1m.

Суффикс `g` принимают только `client_max_body_size`, `directio`, `directio_alignment`, `worker_rlimit_core`, `proxy_cache_max_range_offset`, а также параметры `max_size` и `min_free` директивы `proxy_cache_path`. Директивы кэширования с префиксами `fastcgi_`, `scgi_` и `uwsgi_` ведут себя так же. Для всех остальных размеров предельный суффикс — `m`, включая размеры буферов и зон разделяемой памяти: `proxy_buffer_size 1g`; — ошибка конфигурации. То же ограничение действует и для размеров файлов: у `proxy_max_temp_file_size` значение по умолчанию — 1024m, но `1g` эта директива не принимает.

Интервалы времени можно указывать в миллисекундах, секундах, минутах, часах, днях и так далее, с использованием следующих суффиксов:

ms	Миллисекунды
s	Секунды
m	Минуты
h	Часы
d	Дни
w	Недели
M	Месяцы (принято считать равными 30 дням)
y	Годы (принято считать равными 365 дням)

Несколько единиц могут быть объединены в одном значении, указывая их в порядке от наиболее значимого к наименее значимому, при необходимости разделяя пробелами. Например, "1h 30m" обозначает тот же промежуток времени, что и "90m" или "5400s". Значение без суффикса интерпретируется как секунды. Рекомендуется всегда указывать суффикс.

Некоторые интервалы времени могут быть указаны только с разрешением в секундах.

Директивы

Каждая директива состоит из имени и набора параметров. Если какая-либо часть директивы должна содержать пробелы, она должна быть заключена в кавычки или экранирована:

```
add_header X-MyHeader "foo bar";
add_header X-MyHeader foo\ bar;
```

Если именованный параметр требует пробелов и вы используете кавычки, его имя также должно быть заключено в кавычки:

```
server example.com "sid=server 1";
```

Настройка хэшей

Для эффективной обработки статических наборов данных, таких как имена серверов, значения директивы *map*, MIME-типы и имена заголовков запросов, Angie ADC использует хэш-таблицы. При запуске и каждом переопределении конфигурации Angie ADC определяет оптимальный размер для этих хэш-таблиц, чтобы размер корзины, которая хранит ключи с одинаковыми хэш-значениями, не превышал заданный параметр (*hash bucket size*). Размер таблицы измеряется в корзинах и корректируется до тех пор, пока не превысит параметр *hash max size*. Большинство хэш-таблиц имеют соответствующие директивы для настройки этих параметров, такие как *server_names_hash_max_size* и *server_names_hash_bucket_size* для имен серверов.

Параметр *hash bucket size* выравнивается по кратности размера линии кэша процессора. Такое выравнивание улучшает эффективность поиска ключей на современных процессорах, уменьшая количество обращений к памяти. Если *hash bucket size* равен размеру одной линии кэша, максимальное количество обращений к памяти во время поиска ключа будет два: одно для вычисления адреса корзины и второе для поиска внутри корзины. Поэтому, если Angie ADC сообщает, что следует увеличить либо *hash max size*, либо *hash bucket size*, начните с увеличения *hash max size*.

Перезагрузка конфигурации

Перезагрузка конфигурационного файла происходит автоматически при сохранении изменений в конфигурации балансировщика нагрузки. Подробнее см. [Настройка конфигурации](#).

4.10.8.2 Соединения, сессии, запросы, логи

Механизмы обработки соединений

Angie поддерживает различные методы обработки соединений. Доступность конкретного метода зависит от используемой платформы. На платформах, поддерживающих несколько методов, Angie

обычно автоматически выбирает наиболее эффективный метод. Однако, при необходимости, метод обработки соединений можно явно выбрать с помощью директивы *use*.

Доступны следующие методы обработки соединений:

Метод	Описание
<code>select</code>	Стандартный метод. Соответствующий модуль собирается автоматически на платформах, не имеющих более эффективных методов.
<code>poll</code>	Стандартный метод. Соответствующий модуль собирается автоматически на платформах, не имеющих более эффективных методов.
<code>kqueue</code>	Эффективный метод, доступный на FreeBSD 4.1+, OpenBSD 2.9+, NetBSD 2.0 и macOS.
<code>epoll</code>	Эффективный метод, доступный на Linux 2.6+.
<code>/dev/poll</code>	Эффективный метод, доступный на Solaris 7 11/99+, HP/UX 11.22+ (eventport), IRIX 6.5.15+ и Tru64 UNIX 5.1A+.
<code>eventport</code>	Метод <code>event ports</code> доступен на Solaris 10+. (Из-за известных проблем рекомендуется использовать метод <code>/dev/poll</code> .)

Обработка HTTP-запросов

Каждый HTTP-запрос проходит через ряд фаз, на каждой из которых выполняется определенный тип обработки.

<code>Post-read</code>	Начальная фаза. Модуль <i>RealIP</i> вызывается на этой фазе.
<code>Server-rewrite</code>	Фаза, на которой обрабатываются директивы из модуля <i>Rewrite</i> , определенные в блоке <code>server</code> (но вне блока <code>location</code>).
<code>Find-config</code>	Специальная фаза, на которой выбирается <i>location</i> на основе URI запроса.
<code>Rewrite</code>	Похожа на фазу <code>Server-rewrite</code> , но применяется к правилам <i>rewrite</i> , определенным в блоке <code>location</code> , выбранном на предыдущей фазе.
<code>Post-rewrite</code>	Специальная фаза, на которой запрос перенаправляется в новое место, как на фазе <code>Find-config</code> , если его URI был изменен во время фазы <code>Rewrite</code> .
<code>Preaccess</code>	На этой фазе стандартные модули Angie, такие как <i>Limit Req</i> , регистрируют свои обработчики.
<code>Access</code>	Фаза, на которой проверяется право клиента на выполнение запроса, обычно с помощью стандартных модулей Angie, таких как <i>Auth Basic</i> .
<code>Post-access</code>	Специальная фаза, на которой обрабатывается директива <i>satisfy any</i> .
<code>Precontent</code>	На этой фазе стандартные модули, такие как директивы <i>try_files</i> и <i>mirror</i> , регистрируют свои обработчики.
<code>Content</code>	Фаза, на которой обычно генерируется ответ. На этом этапе несколько стандартных модулей Angie регистрируют свои обработчики, включая <i>Index</i> ; также сюда относятся директивы <i>proxy_pass</i> , <i>fastcgi_pass</i> , <i>uwsgi_pass</i> , <i>scgi_pass</i> и <i>grpc_pass</i> . Обработчики вызываются последовательно, пока один из них не произведет вывод.
<code>Log</code>	Финальная фаза, на которой выполняется логирование запросов. В настоящее время только модуль <i>Log</i> регистрирует свой обработчик на этом этапе для ведения журнала доступа.

Обработка TCP/UDP-сессий

TCP/UDP-сессия от клиента проходит через ряд фаз, на каждой из которых выполняется определенный тип обработки:

Post-accept	Начальная фаза после принятия соединения от клиента. На этой фазе вызывается модуль <i>RealIP</i> .
Pre-access	Предварительная фаза для проверки доступа. Модули <i>Set</i> вызываются на этой фазе.
Access	Фаза для ограничения доступа клиента перед фактической обработкой данных. На этом этапе вызывается модуль <i>Access</i> .
SSL	Фаза, на которой происходит терминация TLS/SSL. На этой фазе вызывается модуль <i>SSL</i> .
Preread	Фаза для чтения начальных байт данных в <i>preread buffer</i> , чтобы позволить таким модулям, как <i>SSL Preread</i> , проанализировать данные до их обработки.
Content	Обязательная фаза, на которой данные фактически обрабатываются, обычно с участием модуля <i>Return</i> , который отправляет ответ клиенту. Директива <i>proxy_pass</i> также относится сюда.
Log	Финальная фаза, на которой фиксируется результат обработки сессии клиента. На этой фазе вызывается модуль <i>Log</i> .

Обработка запросов

Выбор виртуального сервера

Изначально соединение создается в контексте сервера по умолчанию. Имя сервера может быть определено на следующих этапах обработки запроса, каждый из которых участвует в выборе конфигурации сервера:

- Во время SSL-рукопожатия, заранее, в соответствии с SNI.
- После обработки строки запроса.
- После обработки поля заголовка *Host*.

Если имя сервера не определено после обработки строки запроса или поля заголовка *Host*, Angie использует пустое имя в качестве имени сервера.

На каждом из этих этапов могут применяться различные конфигурации сервера. Поэтому некоторые директивы следует указывать с осторожностью:

- В случае директивы *ssl_protocols* список протоколов устанавливается библиотекой OpenSSL до применения конфигурации сервера в соответствии с именем, запрошенным через SNI. В результате протоколы следует указывать только для сервера по умолчанию.
- Директивы *client_header_buffer_size* и *merge_slashes* применяются до чтения строки запроса. Поэтому эти директивы используют либо конфигурацию сервера по умолчанию, либо конфигурацию сервера, выбранную через SNI.
- В случае директив *ignore_invalid_headers*, *large_client_header_buffers* и *underscores_in_headers*, которые участвуют в обработке полей заголовков запроса, конфигурация сервера дополнительно зависит от того, была ли она обновлена в соответствии со строкой запроса или полем заголовка *Host*.
- Ответ с ошибкой обрабатывается с использованием директивы *error_page* на сервере, который в данный момент обрабатывает запрос.

Виртуальные серверы на основе имен

Сначала Angie ADC определяет, какой сервер должен обрабатывать запрос. Рассмотрим простую конфигурацию, где все три виртуальных сервера слушают на порту 80:

```
server {
    listen 80;
    server_name example.org www.example.org;
```

```
# ...
}

server {

    listen 80;
    server_name example.net www.example.net;
    # ...
}

server {

    listen 80;
    server_name example.com www.example.com;
    # ...
}
```

В этой конфигурации Angie ADC определяет, какой сервер должен обработать запрос, основываясь исключительно на поле заголовка **Host**. Если значение этого заголовка не совпадает ни с одним из имен серверов или если запрос не содержит этого заголовка, Angie ADC направит запрос к серверу по умолчанию для этого порта. В приведенной выше конфигурации сервером по умолчанию является первый — что является стандартным поведением Angie ADC. Также можно явно указать, какой сервер должен быть сервером по умолчанию, используя параметр `default_server` в директиве `listen`:

```
server {

    listen 80 default_server;
    server_name example.net www.example.net;
    # ...
}
```

Примечание

Обратите внимание, что сервер по умолчанию является свойством слушающего сокета, а не имени сервера.

Международные имена

Международные доменные имена (IDNs) должны указываться в директиве `server_name` с использованием представления ASCII (Punycode):

```
server {

    listen 80;
    server_name xn--e1afmkfd.xn--80akhbyknj4f; # пример.испытание
    # ...
}
```

Запрет запросов с неопределенными именами серверов

Если запросы без заголовка **Host** нежелательны, можно определить сервер, который просто отклоняет такие запросы:

```
server {
```

```
listen 80;
server_name "";
return 444;
}
```

В этой конфигурации имя сервера задано пустой строкой, что соответствует запросам без заголовка `Host`. Затем возвращается специальный нестандартный код 444, который закрывает соединение.

Сочетание виртуальных серверов, основанных на именах и IP-адресах

Рассмотрим более сложную конфигурацию, где некоторые виртуальные серверы слушают на разных адресах:

```
server {

    listen 192.168.1.1:80;
    server_name example.org www.example.org;
    # ...
}

server {

    listen 192.168.1.1:80;
    server_name example.net www.example.net;
    # ...
}

server {

    listen 192.168.1.2:80;
    server_name example.com www.example.com;
    # ...
}
```

В этой конфигурации Angie ADC сначала проверяет IP-адрес и порт запроса по директивам `listen` блоков `server`. Затем он проверяет поле заголовка `Host` запроса по записям `server_name` блоков `server`, которые совпали с IP-адресом и портом. Если имя сервера не найдено, запрос будет обработан сервером по умолчанию. Например, запрос к `www.example.com`, полученный на порту 192.168.1.1:80, будет обработан сервером по умолчанию для этого порта — т.е. первым сервером — поскольку `www.example.com` не определен для этого порта.

Как упоминалось ранее, сервер по умолчанию является свойством слушающего сокета, и можно определить разные серверы по умолчанию для разных портов:

```
server {

    listen 192.168.1.1:80;
    server_name example.org www.example.org;
    # ...
}

server {

    listen 192.168.1.1:80 default_server;
    server_name example.net www.example.net;
    # ...
}
```

```
server {

    listen 192.168.1.2:80 default_server;
    server_name example.com www.example.com;
    # ...
}
```

Выбор локаций

Рассмотрим простую конфигурацию PHP-сайта:

```
server {

    listen 80;
    server_name example.org www.example.org;
    root /data/www;

    location / {

        index index.html index.php;
    }

    location ~* \.(gif|jpg|png)$ {

        expires 30d;
    }

    location ~ \.php$ {

        fastcgi_pass localhost:9000;
        fastcgi_param SCRIPT_FILENAME
        $document_root$fastcgi_script_name;
        include fastcgi_params;
    }
}
```

Angie сначала ищет наиболее конкретный префикс `location`, заданный буквальными строками, независимо от их порядка в списке. В приведенной выше конфигурации единственный префикс локации — это `location /`, который соответствует любому запросу и будет использоваться в качестве последнего средства. Затем Angie проверяет локации, определенные с помощью регулярных выражений, в порядке их появления в конфигурационном файле. Первое совпадающее выражение завершает поиск, и Angie использует эту `location`. Если ни одно регулярное выражение не совпадает с запросом, Angie использует наиболее конкретную префиксную `location`, найденную ранее.

Примечание

Локации всех типов проверяют только URI часть строки запроса, исключая аргументы. Это связано с тем, что аргументы в строке запроса могут быть указаны разными способами, например:

- `/index.php?user=john&page=1`
- `/index.php?page=1&user=john`

Кроме того, строка запроса может содержать любое количество параметров:

- `/index.php?page=1&something+else&user=john`

Теперь давайте рассмотрим, как запросы будут обрабатываться в приведенной выше конфигурации:

- Запрос `/logo.gif` сначала соответствует префиксу `location /`, а затем регулярному выражению `.(gif|jpg|png)$`. Поэтому он обрабатывается последней локацией. Используя директиву `root /data/www`, запрос сопоставляется с файлом `/data/www/logo.gif`, и файл отправляется клиенту.
- Запрос `/index.php` также изначально соответствует префиксу `location /`, а затем регулярному выражению `.(php)$`. Следовательно, он обрабатывается последней локацией, и запрос передается серверу FastCGI, слушающему на `localhost:9000`. Директива `fastcgi_param` устанавливает параметр FastCGI `SCRIPT_FILENAME` в `/data/www/index.php`, и сервер FastCGI выполняет файл. Переменная `$document_root` устанавливается в значение директивы `root`, а переменная `$fastcgi_script_name` устанавливается в URI запроса, т.е. `/index.php`.
- Запрос `/about.html` соответствует только префиксу `location /`, поэтому он обрабатывается в этой локации. Используя директиву `root /data/www`, запрос сопоставляется с файлом `/data/www/about.html`, и файл отправляется клиенту.

Обработка запроса `/` более сложна. Он соответствует только префиксу `location /`, поэтому он обрабатывается в этой локации. Директива `index` затем проверяет наличие индексных файлов в соответствии с ее параметрами и директивой `root /data/www`. Если файл `/data/www/index.html` не существует, но файл `/data/www/index.php` существует, директива выполняет внутреннюю переадресацию на `/index.php`, и Angie снова ищет локации, как если бы запрос был отправлен клиентом. Как упоминалось ранее, переадресованный запрос в конечном итоге будет обработан сервером FastCGI.

Проксирование и балансировка нагрузки

Одним из распространенных способов использования Angie ADC является настройка в качестве прокси-сервера. В этой роли Angie ADC принимает запросы, перенаправляет их на проксируемые серверы, получает ответы от этих серверов и отправляет ответы обратно клиентам.

Простой прокси-сервер:

```
server {
    location / {
        proxy_pass http://backend:8080;
    }
}
```

Директива `proxy_pass` заставит Angie ADC передавать запросы клиентов на сервер `backend:8080` (прокси-сервер). Существует множество дополнительных *директив*, которые можно использовать для дальнейшей настройки прокси-соединения.

Проксирование FastCGI

Angie ADC может использоваться для маршрутизации запросов на FastCGI-серверы, которые выполняют приложения, построенные с использованием различных фреймворков и языков программирования, таких как PHP.

Простейшая конфигурация Angie ADC для работы с FastCGI-сервером включает использование директивы `fastcgi_pass` вместо директивы `proxy_pass`, а также директив `fastcgi_param` для установки параметров, передаваемых FastCGI-серверу. Предположим, что FastCGI-сервер доступен по адресу `localhost:9000`. В PHP параметр `SCRIPT_FILENAME` используется для определения имени скрипта, а параметр `QUERY_STRING` используется для передачи параметров запроса. Результирующая конфигурация будет следующей:

```
server {
```

```
location / {

    fastcgi_pass localhost:9000;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    fastcgi_param QUERY_STRING $query_string;
}

location ~ \.(gif|jpg|png)$ {

    root /data/images;
}
}
```

Эта конфигурация настраивает сервер, который направляет все запросы, кроме запросов к статическим изображениям, на проксируемый сервер, работающий на `localhost:9000` через протокол FastCGI.

Проксирование WebSocket

Для обновления соединения с HTTP/1.1 до WebSocket используется механизм **протокольного переключения**, доступный в HTTP/1.1.

Однако есть тонкость: поскольку заголовок **Upgrade** является **заголовком перехода**, он не передается от клиента к проксируемому серверу. При использовании прямого проксирования клиенты могут использовать метод **CONNECT** для обхода этой проблемы. Этот подход не работает при обратном проксировании, так как клиенты не осведомлены о каких-либо прокси-серверах, и требуется специальная обработка на прокси-сервере.

Angie ADC реализует специальный режим работы, который позволяет установить туннель между клиентом и проксируемым сервером, если проксируемый сервер возвращает ответ с кодом 101 (Switching Protocols), а клиент запрашивает переключение протокола через заголовок **Upgrade** в запросе.

Как уже упоминалось, заголовки перехода, включая **Upgrade** и **Connection**, не передаются от клиента к проксируемому серверу. Поэтому, чтобы проксируемый сервер был осведомлен о намерении клиента переключиться на протокол WebSocket, эти заголовки должны быть переданы явно:

```
location /chat/ {

    proxy_pass http://backend;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
}
```

Более сложный пример демонстрирует, как значение поля заголовка **Connection** в запросе к проксируемому серверу зависит от наличия поля **Upgrade** в заголовке запроса клиента:

```
http {

    map $http_upgrade $connection_upgrade {

        default upgrade;
        '' close;
    }

    server {

        ...
    }
}
```

```
location /chat/ {

    proxy_pass http://backend;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection $connection_upgrade;
}
}
```

По умолчанию соединение будет закрыто, если проксируемый сервер не передает никаких данных в течение 60 секунд. Этот тайм-аут можно увеличить с помощью директивы `proxy_read_timeout`. В качестве альтернативы: на проксируемом сервере можно настроить периодическую отправку кадров WebSocket ping для сброса тайм-аута и проверки того, активно ли соединение.

Балансировка нагрузки

Балансировка нагрузки между несколькими экземплярами приложения — это широко используемая техника для оптимизации использования ресурсов, максимизации пропускной способности, уменьшения задержек и обеспечения отказоустойчивых конфигураций.

Angie ADC можно использовать в качестве высокоэффективного HTTP-балансировщика нагрузки для распределения трафика между несколькими серверами приложений, тем самым улучшая производительность, масштабируемость и надежность веб-приложений.

Самая простая конфигурация для балансировки нагрузки с помощью Angie ADC может выглядеть следующим образом:

```
http {

    upstream myapp1 {

        server srv1.example.com;
        server srv2.example.com;
        server srv3.example.com;
    }

    server {

        listen 80;

        location / {

            proxy_pass http://myapp1;
        }
    }
}
```

В приведенном примере три экземпляра одного и того же приложения работают на серверах с `srv1` по `srv3`. Когда метод балансировки нагрузки не настроен явно, по умолчанию используется круговой метод (round-robin). Реализация обратного прокси в Angie также поддерживает встроенные (или пассивные) проверки состояния серверов. Эти проверки настраиваются с помощью директив `max_fails` и `fail_timeout` внутри блока `server` в контексте `upstream`.

Логирование

Syslog

Директивы `error_log` и `access_log` поддерживают логирование в `syslog`. Для настройки логирования в `syslog` используются следующие параметры:

<code>server=address</code>	Указывает адрес сервера <code>syslog</code> . Адрес может быть доменным именем или IP-адресом с необязательным портом, либо путем к UNIX-доменному сокету, указанным после префикса <code>"unix:"</code> . Если порт не указан, используется UDP-порт 514. Если доменное имя разрешается в несколько IP-адресов, используется первый разрешенный адрес.
<code>facility=string</code>	Устанавливает уровень для сообщений <code>syslog</code> , как определено в RFC 3164 . Возможные уровни включают: <code>"kern"</code> , <code>"user"</code> , <code>"mail"</code> , <code>"daemon"</code> , <code>"auth"</code> , <code>"intern"</code> , <code>"lpr"</code> , <code>"news"</code> , <code>"uucp"</code> , <code>"clock"</code> , <code>"authpriv"</code> , <code>"ftp"</code> , <code>"ntp"</code> , <code>"audit"</code> , <code>"alert"</code> , <code>"cron"</code> , <code>"local0".."local7"</code> . По умолчанию используется <code>"local7"</code> .
<code>severity=string</code>	Определяет уровень серьезности сообщений <code>syslog</code> для <code>access_log</code> , как указано в RFC 3164 . Возможные значения те же, что и для второго параметра (уровень) директивы <code>error_log</code> . По умолчанию используется <code>"info"</code> . Серьезность сообщений об ошибках определяется Angie, поэтому этот параметр игнорируется в директиве <code>error_log</code> .
<code>tag=string</code>	Устанавливает тег для сообщений <code>syslog</code> . По умолчанию используется тег <code>"angie"</code> .
<code>nohostname</code>	Отключает добавление поля <code>hostname</code> в заголовок сообщения <code>syslog</code> .

Пример конфигурации `syslog`:

```
error_log syslog:server=192.168.1.1 debug;

access_log syslog:server=[2001:db8::1]:12345,facility=local7,tag=angie,severity=info,
↪combined;
```

4.10.8.3 Основной модуль

Модуль предоставляет основную функциональность и директивы конфигурации, необходимые для базовой работы сервера, а также решает важные задачи, такие как управление рабочими процессами, настройка событийно-ориентированных моделей и обработка входящих соединений и запросов. Он включает ключевые директивы для настройки основного процесса, ведения журналов ошибок и контроля поведения сервера на низком уровне.

Пример конфигурации

```
user www www;
worker_processes 2;

error_log /var/log/error.log info;

events {
    use kqueue; worker_connections 2048;
}
```


Директивы

accept_mutex

<i>Синтаксис</i>	<code>accept_mutex on off;</code>
По умолчанию	<code>accept_mutex off;</code>
<i>Контекст</i>	events

Когда `accept_mutex` включен, рабочие процессы будут принимать новые соединения поочередно. Иначе уведомления о новых соединениях получают все рабочие процессы, что может привести к неэффективному использованию системных ресурсов, если количество новых соединений невелико.

Примечание

Нет необходимости включать `accept_mutex` на системах, которые поддерживают флаг `EPOLLEXCLUSIVE`, или при использовании директивы `reuseport`.

accept_mutex_delay

<i>Синтаксис</i>	<code>accept_mutex_delay время;</code>
По умолчанию	<code>accept_mutex_delay 500ms;</code>
<i>Контекст</i>	events

Если `accept_mutex` включен, эта директива указывает максимальное время, в течение которого рабочий процесс ждет, чтобы продолжить принимать новые соединения, если другой рабочий процесс уже обрабатывает новые соединения.

daemon

<i>Синтаксис</i>	<code>daemon on off;</code>
По умолчанию	<code>daemon on;</code>
<i>Контекст</i>	main

Определяет, будет ли Angie запускаться в режиме демона. Используется в основном для разработки.

debug_connection

<i>Синтаксис</i>	<code>debug_connection адрес CIDR unix;;</code>
По умолчанию	—
<i>Контекст</i>	events

Включает отладочный лог для отдельных клиентских соединений. Для остальных соединений используется уровень лога, заданный директивой `error_log`. Указывать соединения можно по IPv4- или IPv6-адресу, сети или имени хоста. Используйте параметр `unix:`, чтобы включить отладочный лог для соединений через UNIX-сокеты.

```
events {

    debug_connection 127.0.0.1;
    debug_connection localhost;
    debug_connection 192.0.2.0/24;
    debug_connection ::1;
    debug_connection 2001:0db8::/32;
    debug_connection unix;;
    # ...
}
```

Примечание

Чтобы эта директива работала, в сборке Angie должен быть включен отладочный лог.

debug_points

<i>Синтаксис</i>	<code>debug_points abort stop;</code>
По умолчанию	—
<i>Контекст</i>	main

Эта директива используется для отладки.

В случае обнаружения внутренней ошибки, например, утечки сокетов в момент перезапуска рабочих процессов, включение `debug_points` либо создаст файл дампа памяти (`abort`), либо остановит процесс (`stop`) для анализа с помощью отладчика.

env

<i>Синтаксис</i>	<code>env переменная [=значение];</code>
По умолчанию	<code>env TZ;</code>
<i>Контекст</i>	main

По умолчанию Angie удаляет все переменные окружения, унаследованные от родительского процесса, кроме переменной `TZ`. Эта директива позволяет сохранить часть унаследованных переменных, поменять их значения или создать новые переменные окружения.

Эти переменные затем доступны рабочими процессами.

Обратите внимание, что управление системными библиотеками таким образом может быть не всегда эффективным, поскольку библиотеки часто проверяют переменные только во время инициализации, которая происходит до срабатывания этой директивы.

Пример:

```
env MALLOC_OPTIONS;
env PERL5LIB=/data/site/modules;
env OPENSSL_ALLOW_PROXY_CERTS=1;
```

Примечание

Переменная окружения **ANGIE** используется внутри Angie и не должна задаваться напрямую пользователем.

error_log

<i>Синтаксис</i>	<code>error_log файл [уровень];</code>
По умолчанию	<code>error_log logs/error.log error;</code>
<i>Контекст</i>	main, http, mail, stream, server, location

Настраивает логирование, позволяя указывать несколько логов на одном уровне конфигурации. Если файл лога не указан явно на уровне конфигурации **main**, будет использоваться файл по умолчанию.

Первый параметр указывает файл для хранения лога. Специальное значение **stderr** задает стандартный поток ошибок. Для настройки логирования в *syslog* используйте префикс **"syslog:"**. Для логирования в циклический буфер памяти используйте префикс **"memory:"**, за которым следует размер буфера; обычно он используется для отладки.

Второй параметр задает уровень логирования одним из следующих значений: **debug**, **info**, **notice**, **warn**, **error**, **crit**, **alert** или **emerg**. Уровни перечислены в порядке возрастания серьезности. При задании уровня будут логироваться сообщения равного и более высокого уровня:

Настройка	Уровни записи
debug	debug, info, notice, warn, error, crit, alert, emerg
info	info, notice, warn, error, crit, alert, emerg
notice	notice, warn, error, crit, alert, emerg
warn	warn, error, crit, alert, emerg
error	error, crit, alert, emerg
crit	crit, alert, emerg
alert	alert, emerg
emerg	emerg

Если этот параметр не задан, по умолчанию используется уровень логирования **error**.

Примечание

Чтобы работал уровень логирования **debug**, должен быть включен режим отладки.

Каждая запись в журнале ошибок имеет следующий формат:

```
временная_метка [уровень] PID#TID: *номер_соединения сообщение
```

Где:

- **временная_метка** — дата и время события;
- **уровень** — уровень логирования события;
- **PID#TID** — идентификаторы процесса и потока;
- ***номер_соединения** — порядковый номер соединения, общий для всех запросов этого соединения (если применимо);
- **сообщение** — текст сообщения об ошибке или событии.

events

<i>Синтаксис</i>	<code>events { ... };</code>
По умолчанию	—
<i>Контекст</i>	main

Предоставляет контекст конфигурационного файла для директив, влияющих на обработку соединений.

include

<i>Синтаксис</i>	<code>include файл маска;</code>
По умолчанию	—
<i>Контекст</i>	любой

Включает в конфигурацию другой файл или файлы, подходящие под заданную *маску*. Включаемые файлы должны содержать синтаксически верные директивы и блоки.

Пример:

```
include mime.types;
include vhosts/*.conf;
```

lock_file

<i>Синтаксис</i>	<code>lock_file файл;</code>
По умолчанию	<code>lock_file logs/angie.lock;</code>
<i>Контекст</i>	main

Angie использует механизм блокировок для реализации *accept_mutex* и сериализации доступа к разделяемой памяти. На большинстве систем блокировки управляются с помощью атомарных операций, что делает эту директиву ненужной. Однако на некоторых системах используется альтернативный механизм *lock file*. Эта директива устанавливает префикс для имен файлов блокировок.

master_process

<i>Синтаксис</i>	<code>master_process on off;</code>
По умолчанию	<code>master_process on;</code>
<i>Контекст</i>	main

Определяет, будут ли запускаться рабочие процессы. Эта директива предназначена для разработчиков Angie.

multi_accept

<i>Синтаксис</i>	multi_accept on off;
По умолчанию	multi_accept off;
<i>Контекст</i>	events

on	Рабочий процесс будет принимать сразу все новые соединения.
off	Рабочий процесс будет принимать только одно новое соединение за раз.

Примечание

Директива игнорируется при использовании метода обработки соединений *kqueue*, так как он сам задает число новых соединений, ожидающих приема.

pcre_jit

<i>Синтаксис</i>	pcre_jit on off;
По умолчанию	pcre_jit off;
<i>Контекст</i>	main

Разрешает или запрещает использование JIT-компиляции (PCRE JIT) для регулярных выражений, известных на момент парсинга конфигурации.

Использование PCRE JIT заметно ускоряет обработку регулярных выражений.

Примечание

Для работы JIT необходима библиотека PCRE версии 8.20 или выше, собранная с параметром сборки `--enable-jit`. Если Angie собирается с библиотекой PCRE (`--with-pcre=`), поддержка JIT включается с помощью параметра `--with-pcre-jit`.

pid

<i>Синтаксис</i>	pid <i>файл</i> off;
По умолчанию	pid logs/angie.pid;
<i>Контекст</i>	main

Указывает *файл*, где будет храниться идентификатор главного процесса Angie. Файл создается атомарно, что обеспечивает корректность его содержимого. Настройка `off` отключает создание этого файла.

Примечание

Если значение *file* изменяется при переконфигурации, но указывает на симлинк предыдущего PID-файла, файл не будет создан заново.

ssl_engine

<i>Синтаксис</i>	<code>ssl_engine устройство;</code>
По умолчанию	—
<i>Контекст</i>	main

Задаёт название аппаратного SSL-акселератора.

ssl_object_cache_inheritable

<i>Синтаксис</i>	<code>ssl_object_cache_inheritable on off;</code>
Значение по умолчанию	<code>ssl_object_cache_inheritable on;</code>
<i>Контекст</i>	main

Если включено, SSL-объекты (SSL-сертификаты, секретные ключи, доверенные сертификаты СА, списки отзыва сертификатов) наследуются при перезагрузке конфигурации.

SSL-объекты, загруженные из файлов, наследуются, если время их изменения и индекс файла не изменились с момента предыдущей загрузки конфигурации. Секретные ключи, указанные как `engine:name:id`, никогда не наследуются, тогда как ключи, указанные как `data:value`, всегда наследуются.

SSL-объекты, загруженные из переменных, не могут быть унаследованы.

Пример:

```
ssl_object_cache_inheritable on;

http {
    server {
        ssl_certificate     example.com.crt;
        ssl_certificate_key example.com.key;
    }
}
```

thread_pool

<i>Синтаксис</i>	<code>thread_pool имя threads=число [max_queue=число];</code>
По умолчанию	<code>thread_pool default threads=32 max_queue=65536;</code>
<i>Контекст</i>	main

Задаёт *имя* и параметры пула потоков, используемого для многопоточной обработки операций чтения и отправки файлов *без блокирования* рабочих процессов.

Параметр **threads** задаёт число потоков в пуле.

Если все потоки в пуле заняты выполнением заданий, новые задания ждут в очереди. Параметр **max_queue** ограничивает число заданий, ожидающих своего выполнения в очереди. По умолчанию в очереди может находиться до 65536 заданий. Если очередь переполнена, новые задания завершаются с ошибкой.

timer_resolution

<i>Синтаксис</i>	<code>timer_resolution интервал;</code>
По умолчанию	—
<i>Контекст</i>	main

Уменьшает разрешение таймеров времени в рабочих процессах, за счет чего уменьшается число системных вызовов `gettimeofday()`. По умолчанию `gettimeofday()` вызывается при каждом получении событий из ядра. При уменьшении разрешения функция вызывается только раз за указанный интервал.

Пример:

```
timer_resolution 100ms;
```

Внутренняя реализация интервала зависит от используемого метода:

- Фильтр `EVFILT_TIMER`, если используется *kqueue*.
- Функция `timer_create()`, если используется *eventport*.
- Функция `setitimer()`, в противном случае.

use

<i>Синтаксис</i>	<code>use метод;</code>
По умолчанию	—
<i>Контекст</i>	events

Задаёт *метод*, используемый для *обработки соединений*. Обычно нет необходимости задавать его явно, поскольку по умолчанию Angie выбирает наиболее эффективный метод.

user

<i>Синтаксис</i>	<code>user пользователь [группа];</code>
По умолчанию	<code>user <параметр сборки --user> <параметр сборки --group>;</code>
<i>Контекст</i>	main

Задаёт пользователя и группу для рабочих процессов. Если задан только пользователь, для группы также задается указанное имя пользователя.

worker_aio_requests

<i>Синтаксис</i>	<code>worker_aio_requests число;</code>
По умолчанию	<code>worker_aio_requests 32;</code>
<i>Контекст</i>	events

При использовании *aio* с методом обработки соединений *epoll* задает максимум операций асинхронного ввода-вывода, ожидающих обработки, для одного рабочего процесса.

worker_connections

<i>Синтаксис</i>	<code>worker_connections число;</code>
По умолчанию	<code>worker_connections 512;</code>
<i>Контекст</i>	events

Задаёт максимум соединений, которые одновременно может открыть рабочий процесс.

Обратите внимание, что это число включает все соединения, такие как соединения с проксируемыми серверами, а не только клиентские. Кроме того, фактическое количество одновременных соединений не может превышать системный лимит на открытые файлы, который можно настроить с помощью `worker_rlimit_nofile`.

worker_cpu_affinity

<i>Синтаксис</i>	<code>worker_cpu_affinity маска_CPU ...;</code> <code>worker_cpu_affinity auto [маска_CPU];</code>
По умолчанию	—
<i>Контекст</i>	main

Привязывает рабочие процессы к группам процессоров. Каждая группа процессоров задается битовой маской разрешенных процессоров. Для каждого рабочего процесса должна быть задана отдельная группа. По умолчанию рабочие процессы не привязаны к конкретным процессорам.

Пример:

```
worker_processes 4;
worker_cpu_affinity 0001 0010 0100 1000;
```

Эта конфигурация привязывает каждый рабочий процесс к отдельному процессору.

В качестве альтернативы:

```
worker_processes 2;
worker_cpu_affinity 0101 1010;
```

Это привязывает первый рабочий процесс к CPU0 и CPU2, а второй рабочий процесс к CPU1 и CPU3. Такая настройка подходит для гипертренинга.

Специальное значение `auto` позволяет автоматически привязывать рабочие процессы к доступным процессорам:

```
worker_processes auto;
worker_cpu_affinity auto;
```

С помощью необязательной маски можно ограничить процессоры, доступные для автоматической привязки:

```
worker_cpu_affinity auto 01010101;
```

Примечание

Директива доступна только на FreeBSD и Linux.

worker_priority

<i>Синтаксис</i>	<code>worker_priority число;</code>
По умолчанию	<code>worker_priority 0;</code>
<i>Контекст</i>	main

Задаёт приоритет планирования рабочих процессов подобно тому, как это делается командой `nice`: отрицательное *число* означает более высокий приоритет. Диапазон возможных значений — от -20 до 20.

Пример:

```
worker_priority -10;
```

worker_processes

<i>Синтаксис</i>	<code>worker_processes число auto;</code>
По умолчанию	<code>worker_processes 1;</code>
<i>Контекст</i>	main

Задаёт число рабочих процессов.

Оптимальное значение зависит от разных факторов, включая число ядер процессора, количество жестких дисков и характер нагрузки. Если вы не уверены, рекомендуется начать с числа доступных ядер процессора. Значение `auto` пытается автоматически определить оптимальное количество.

worker_rlimit_core

<i>Синтаксис</i>	<code>worker_rlimit_core размер;</code>
По умолчанию	—
<i>Контекст</i>	main

Меняет ограничение на наибольший размер дампа памяти (`RLIMIT_CORE`) для рабочих процессов. Используется для увеличения ограничения без перезапуска главного процесса.

worker_rlimit_nofile

<i>Синтаксис</i>	<code>worker_rlimit_nofile число;</code>
По умолчанию	—
<i>Контекст</i>	main

Меняет ограничение на максимальное число открытых файлов (`RLIMIT_NOFILE`) для рабочих процессов. Используется для увеличения ограничения без перезапуска главного процесса.

worker_shutdown_timeout

<i>Синтаксис</i>	worker_shutdown_timeout время;
По умолчанию	—
<i>Контекст</i>	main

Задаёт таймаут в секундах для постепенного завершения рабочих процессов. По истечении указанного времени Angie попытается закрыть все открытые сейчас соединения, чтобы ускорить завершение работы.

Постепенное завершение работы инициируется отправкой сигнала QUIT главному процессу, который приказывает рабочим процессам прекратить принимать новые подключения, позволяя завершить уже существующие. Рабочие процессы продолжают обрабатывать активные запросы до их завершения, после чего корректно завершают свою работу. Если соединения остаются открытыми дольше worker_shutdown_timeout, Angie принудительно закроет эти соединения для завершения работы. Также клиентские постоянные соединения закрываются только в случае, если они неактивны не менее времени, заданного в *lingering_timeout*.

working_directory

<i>Синтаксис</i>	working_directory каталог;
По умолчанию	—
<i>Контекст</i>	main

Задаёт каталог, который будет текущим для рабочего процесса. Основное применение — запись дампа памяти, поэтому рабочий процесс должен иметь права на запись в этот каталог.

4.10.8.4 HTTP-модули

<i>HTTP</i>	Основная функциональность для обработки HTTP-запросов и ответов, управления HTTP
<i>Access</i>	Контроль доступа на основе IP-адресов и диапазонов CIDR.
<i>ACME</i>	Автоматическое получение и обновление SSL-сертификатов по протоколу ACME для HTTP
<i>Addition</i>	Вставка заданного фрагмента до или после тела ответа.
<i>API</i>	RESTful HTTP-интерфейс для получения базовой информации о веб-сервере и его статике
<i>Auth Basic</i>	Базовая HTTP-аутентификация для контроля доступа по имени пользователя и паролю.
<i>Auth Request</i>	Авторизация с помощью подзапроса к внешнему HTTP-сервису.
<i>AutoIndex</i>	Автоматический листинг директорий без индексного файла.
<i>Browser</i> (устарел)	Определение браузера на основе заголовка User-Agent .
<i>Charset</i>	Настройка и преобразование кодировки ответа.
<i>DAV</i>	Управление файлами на сервере по протоколу WebDAV.
<i>DoH</i>	Сервер DNS over HTTPS (RFC 8484), проксирующий DNS-запросы к группе DNS-серверов
<i>Empty GIF</i>	Отдача однопиксельного прозрачного GIF.
<i>FastCGI</i>	Проксирование запроса к FastCGI-серверу.
<i>FLV</i>	Псевдо-стриминг файлов в формате Flash Video (FLV).
<i>Geo</i>	Преобразование IP-адресов в заданные значения переменных.
<i>gRPC</i>	Проксирование запроса к gRPC-серверу.
<i>GunZIP</i>	Распаковка сжатых GZip-ответов для их модификации и в случаях, когда клиент не подд
<i>GZip</i>	Сжатие ответов методом GZip для экономии трафика.
<i>GZip Static</i>	Отдача статических файлов, предварительно сжатых методом GZip.
<i>Headers</i>	Изменение полей заголовка ответа.
<i>HTTP2</i>	Обработка запросов по протоколу HTTP/2.
<i>HTTP3</i>	Обработка запросов по протоколу HTTP/3.

Таблица 1 – продолжение с предыдущей с

<i>Index</i>	Настройка индексных файлов, обслуживающих запросы с косой чертой в конце (/).
<i>Limit Conn</i>	Ограничение числа одновременных запросов (активных соединений) для защиты от пере
<i>Limit Req</i>	Ограничение частоты запросов для защиты от перегрузки и подбора паролей.
<i>Log</i>	Настройка журнала запросов для отслеживания обращений к ресурсам с целью мониторин
<i>Map</i>	Преобразование переменных на основе predetermined пар "ключ-значение".
<i>Memcached</i>	Получение ответов от Memcached-сервера.
<i>Metric</i>	Пользовательские числовые метрики в API статистики.
<i>Mirror</i>	Зеркалирование запросов на другие серверы.
<i>MP4</i>	Псевдо-стриминг файлов в формате MP4.
<i>Prometheus</i>	Метрики сервера в формате, совместимом с Prometheus, для мониторинга и сбора статис
<i>Proxy</i>	Реверсивное проксирование запросов к другим HTTP-серверам.
<i>Random Index</i>	Случайный выбор индексного файла для запросов, оканчивающихся косой чертой (/).
<i>RealIP</i>	Определение адреса и порта клиента при работе за другим прокси-сервером.
<i>Referer</i>	Валидация значений заголовка Referer .
<i>Rewrite</i>	Модификация URI запроса, перенаправления, установка переменных и выбор configura
<i>SCGI</i>	Проксирование запроса к SCGI-серверу.
<i>Secure Link</i>	Создание защищенных ссылок с возможностью ограничения срока доступа.
<i>Slice</i>	Разделение запроса на множество подзапросов к отдельным фрагментам для лучшего кэ
<i>Split Clients</i>	Создание переменных для A/B-тестирования, канареечных релизов, шардинга и других с
<i>SSI</i>	Обработка команд SSI (Server Side Includes) в ответах.
<i>SSL</i>	Настройка SSL/TLS для обработки запросов по протоколу HTTPS.
<i>Stub Status</i> (устарел)	Глобальные счетчики соединений и запросов в текстовом формате.
<i>Sub</i>	Поиск и замена фрагментов в теле ответа.
<i>Tunnel</i>	Обработка запросов HTTP/1.1 CONNECT и установление сквозных туннелей.
<i>Upstream</i>	Настройка групп проксируемых серверов для балансировки нагрузки.
<i>Upstream Probe</i>	Настройка активных проверок работоспособности для групп проксируемых серверов.
<i>UserID</i>	Выдача и обработка cookie с уникальным идентификатором клиента для отслеживания с
<i>uWSGI</i>	Проксирование запроса к uWSGI-серверу.

HTTP-модуль

Базовый HTTP-модуль реализует основную функциональность HTTP-сервера: это определение серверных блоков, настройка локаций для маршрутизации запросов, передача статических файлов и контроль доступа, настройка перенаправлений, поддержка соединений keep-alive и управление заголовками запросов и ответов.

Остальные модули этого раздела расширяют эту функциональность, позволяя гибко настраивать и оптимизировать работу HTTP-сервера под различные сценарии и требования.

Директивы

absolute_redirect

Синтаксис	absolute_redirect on off;
По умолчанию	absolute_redirect on;
Контекст	http, server, location

Если запрещено, то перенаправления, выдаваемые Angie, будут относительными.

См. также директивы *server_name_in_redirect* и *port_in_redirect*.

aio

Синтаксис	<code>aio on off threads [=nul];</code>
По умолчанию	<code>aio off;</code>
Контекст	<code>http, server, location</code>

Разрешает или запрещает использование файлового асинхронного ввода-вывода (AIO) во FreeBSD и Linux:

```
location /video/ {
    aio      on;
    output_buffers 1 64k;
}
```

Во FreeBSD AIO можно использовать, начиная с FreeBSD 4.3. До FreeBSD 11.0 AIO можно либо собрать в ядре статически:

```
options VFS_AIO
```

либо загрузить динамически через загружаемый модуль ядра:

```
kldload aio
```

В Linux AIO можно использовать только начиная с версии ядра 2.6.22. Кроме того, необходимо также дополнительно включить *directio*, иначе чтение будет блокирующимся:

```
location /video/ {
    aio      on;
    directio 512;
    output_buffers 1 128k;
}
```

В Linux *directio* можно использовать только для чтения блоков, выравненных на границу 512 байт (или 4K для XFS). Невыравненный конец файла будет читаться блокированно. То же относится к запросам с указанием диапазона запрашиваемых байт (byte-range requests) и к запросам FLV не с начала файла: чтение невыравненных начала и конца ответа будет блокирующимся.

При одновременном включении AIO и *sendfile* в Linux для файлов, размер которых больше либо равен указанному в директиве *directio*, будет использоваться AIO, а для файлов меньшего размера или при выключенном *directio* — *sendfile*:

```
location /video/ {
    sendfile on;
    aio      on;
    directio 8m;
}
```

Кроме того, читать и отправлять файлы можно в многопоточном режиме, не блокируя при этом рабочий процесс:

```
location /video/ {
    sendfile on;
    aio      threads;
}
```

Операции чтения или отправки файлов будут обрабатываться потоками из указанного пула. Если пул потоков не задан явно, используется пул с именем `default`. Имя пула может быть задано при помощи переменных:

```
aio threads=pool$disk;
```

В настоящий момент многопоточность совместима только с методами `epoll`, `kqueue` и `eventport`. Отправка файлов в многопоточном режиме поддерживается только на Linux.

См. также директиву `sendfile`.

aio_write

Синтаксис	<code>aio_write on off;</code>
По умолчанию	<code>aio_write off;</code>
Контекст	<code>http, server, location</code>

При включенном `aio` разрешает его использование для записи файлов. В настоящий момент это работает только при использовании `aio threads` и ограничено записью временных файлов с данными, полученными от проксируемых серверов.

alias

Синтаксис	<code>alias <i>путь</i>;</code>
По умолчанию	—
Контекст	<code>location</code>

Задаёт замену для указанного `location`'а. Например, при такой конфигурации:

```
location /i/ {
    alias /data/w3/images/;
}
```

на запрос `/i/top.gif` будет отдан файл `/data/w3/images/top.gif`.

В значении параметра `путь` можно использовать переменные, кроме `$document_root` и `$realpath_root`.

Если `alias` используется внутри `location`'а, заданного регулярным выражением, то регулярное выражение должно содержать группы захвата, а сам `alias` — ссылки на эти группы, например:

```
location ~ ^/users/(.+\. (?:(gif|jpe?g|png)))$ {
    alias /data/w3/images/$1;
}
```

Если `location` и последняя часть значения директивы совпадают:

```
location /images/ {
    alias /data/w3/images/;
}
```

то лучше воспользоваться директивой `root`:

```
location /images/ {
    root /data/w3;
}
```

⚠ Предупреждение

При использовании `alias` совпавший префикс локации удаляется из URI по его длине, причем сам префикс повторно не проверяется. Завершающие косые черты у имени локации и у значения `alias` должны быть согласованы. Если в значении `alias` нет завершающей косой черты, которая есть у локации, остаток дописывается вплотную (например, `/data/w3/imagestop.gif`) и отдается не тот файл; в обратном случае лишь удваивается косая черта, что обычно безвредно. Для локации, совпадающей с последней частью пути, предпочтительнее директива `root`, которая добавляет URI целиком и не подвержена этой проблеме.

auth_delay

Синтаксис	<code>auth_delay время;</code>
По умолчанию	<code>auth_delay 0s;</code>
Контекст	<code>http, server, location</code>

Задерживает обработку неавторизованных запросов с кодом ответа 401 для предотвращения атак по времени в случае ограничения доступа по паролю или по результату подзапроса.

auto_redirect

Синтаксис	<code>auto_redirect [on off default];</code>
По умолчанию	<code>auto_redirect default;</code>
Контекст	<code>http, server, location</code>

Контролирует поведение перенаправления, когда префикс `location` заканчивается косой чертой:

```
location /prefix/ {
    auto_redirect on;
}
```

Здесь запрос к `/prefix` будет перенаправлен на `/prefix/`.

Значение `on` включает перенаправление в явном виде; `off` отключает его. Если указано `default`, перенаправление включается, только если `location` обрабатывает запросы через `api`, `proxy_pass`, `fastcgi_pass`, `uwsgi_pass`, `scgi_pass`, `memcached_pass`, или `grpc_pass`.

chunked_transfer_encoding

Синтаксис	<code>chunked_transfer_encoding on off;</code>
По умолчанию	<code>chunked_transfer_encoding on;</code>
Контекст	<code>http, server, location</code>

Позволяет запретить формат передачи данных частями (chunked transfer encoding) в HTTP/1.1. Это может понадобиться при использовании программ, не поддерживающих chunked encoding, несмотря на требования стандарта.

client

Синтаксис	<code>client { ... }</code>
По умолчанию	—
Контекст	<code>http</code>

Создает специальный контекст `client` для обработки внутренних HTTP-запросов, которые Angie выполняет самостоятельно без участия внешних клиентов.

Контекст `client` изолирует служебный трафик различных модулей Angie от пользовательского трафика, позволяя дополнительно управлять им. Внутри этого контекста можно определять только именованные `location` (с префиксом `@`); они недоступны для внешних HTTP-запросов и могут вызываться только программно через внутренние механизмы сервера.

Контекст `client` используется для:

- отправки запросов к центру сертификации в модуле ACME через предопределенный `location @acme`, который можно дополнительно настроить с помощью директив модуля Proху;
- проверки состояния проксируемых серверов через `upstream_probe`;
- режима sticky learn с `remote_action` в потоковом модуле Upstream.

Поддержка нескольких блоков `client` позволяет группировать общие настройки для нескольких блоков `location` внутри каждого блока, что помогает избежать дублирования конфигурации.

При этом директивы, указанные в каждом блоке `client`, наследуются только явно объявленными внутри него блоками `location`. В частности, поэтому они не влияют на конфигурацию других модулей, которые неявно используют блок `client` для исходящих запросов (например, *ACME*).

Пример использования нескольких блоков `client` с наследованием настроек:

```
client {

    proxy_set_header Host docker.example.com;
    proxy_set_header Authorization "Basic QWxhZGRpbjpvGVuIHNlc2FtZQ==";

    location @docker_events {

    }

    location @docker_containers {

    }
}

client {

    proxy_method GET;
    proxy_set_header Host backend.example.com;
    proxy_set_header X-Real-IP $remote_addr;

    location @health_check {

        proxy_pass http://upstream-server/health;
    }
}
```

Примечание

Здесь допускаются те же директивы, что и в обычных `location`, однако реально работают только обработчики контента (например, *autoindex*) и переменных (например, *map*), а также директивы, которые сами порождают запросы, например *upstream_probe*.

Директивы, работающие на других *стадиях обработки запроса* (например, *Limit Req*, *auth_request*, *try_files* и т. д.), здесь не работают.

client_body_buffer_size

Синтаксис	<code>client_body_buffer_size</code> размер;
По умолчанию	<code>client_body_buffer_size</code> 8k 16k;
Контекст	http, server, location

Задаёт размер буфера для чтения тела запроса клиента. Если тело запроса больше заданного буфера, то все тело запроса или только его часть записывается во временный файл. По умолчанию размер одного буфера равен двум размерам страницы. На *x86*, других 32-битных платформах и *x86-64* это 8К. На других 64-битных платформах это обычно 16К.

client_body_in_file_only

Синтаксис	<code>client_body_in_file_only</code> on clean off;
По умолчанию	<code>client_body_in_file_only</code> off;
Контекст	http, server, location

Определяет, сохранять ли все тело запроса клиента в файл. Директиву можно использовать для отладки и при использовании переменной *\$request_body_file*.

on	временные файлы по окончании обработки запроса не удаляются
clean	разрешает удалять временные файлы, оставшиеся по окончании обработки запроса

client_body_in_single_buffer

Синтаксис	<code>client_body_in_single_buffer</code> on off;
По умолчанию	<code>client_body_in_single_buffer</code> off;
Контекст	http, server, location

Определяет, сохранять ли все тело запроса клиента в одном буфере. Директива рекомендуется при использовании переменной *\$request_body* для уменьшения требуемого числа операций копирования.

client_body_temp_path

Синтаксис	<code>client_body_temp_path путь [уровень1 [уровень2 [уровень3]]];</code>
По умолчанию	<code>client_body_temp_path client_body_temp;</code>
Контекст	http, server, location

Задаёт каталог для хранения временных файлов с телами запросов клиентов. В каталоге может использоваться иерархия подкаталогов до трех уровней. Например, при такой конфигурации

```
client_body_temp_path /spool/angie/client_temp 1 2;
```

путь к временному файлу будет следующего вида:

```
/spool/angie/client_temp/7/45/00000123457
```

client_body_timeout

Синтаксис	<code>client_body_timeout время;</code>
По умолчанию	<code>client_body_timeout 60s;</code>
Контекст	http, server, location

Задаёт таймаут при чтении тела запроса клиента. Таймаут устанавливается не на всю передачу тела запроса, а только между двумя последовательными операциями чтения. Если по истечении этого времени клиент ничего не передаст, обработка запроса прекращается с ошибкой 408 (Request Time-out).

client_header_buffer_size

Синтаксис	<code>client_header_buffer_size размер;</code>
По умолчанию	<code>client_header_buffer_size 1k;</code>
Контекст	http, server

Задаёт размер буфера для чтения заголовка запроса клиента. Для большинства запросов достаточно буфера размером в 1К байт. Однако если в запросе есть длинные cookies, или же запрос пришел от WAP-клиента, то он может не поместиться в 1К. Поэтому, если строка запроса или поле заголовка запроса не помещаются полностью в этот буфер, то выделяются буферы большего размера, задаваемые директивой *large_client_header_buffers*.

Если директива указана на уровне *server*, то может использоваться значение из сервера по умолчанию. Подробнее см. в разделе Выбор виртуального сервера.

client_header_timeout

Синтаксис	<code>client_header_timeout время;</code>
По умолчанию	<code>client_header_timeout 60s;</code>
Контекст	http, server

Задаёт таймаут при чтении заголовка запроса клиента. Если по истечении этого времени клиент не передаст полностью заголовок, обработка запроса прекращается с ошибкой 408 (Request Time-out).

client_max_body_size

Синтаксис	<code>client_max_body_size</code> размер;
По умолчанию	<code>client_max_body_size 1m</code> ;
Контекст	http, server, location

Задаёт максимально допустимый размер тела запроса клиента. Если размер больше заданного, то клиенту возвращается ошибка 413 (Request Entity Too Large). Следует иметь в виду, что браузеры не умеют корректно показывать эту ошибку.

0	отключает проверку размера тела запроса клиента
---	---

connection_pool_size

Синтаксис	<code>connection_pool_size</code> размер;
По умолчанию	<code>connection_pool_size 256 512</code> ;
Контекст	http, server, location

Позволяет производить точную настройку выделения памяти под конкретные соединения. Эта директива не оказывает существенного влияния на производительность, и ее не следует использовать. По умолчанию:

256 (байт)	на 32-битных платформах
512 (байт)	на 64-битных платформах

default_type

Синтаксис	<code>default_type</code> mime- <i>mun</i> ;
По умолчанию	<code>default_type</code> text/plain;
Контекст	http, server, location

Задаёт MIME-тип ответов по умолчанию. Соответствие расширений имен файлов MIME-типу ответов задается с помощью директивы *types*.

directio

Синтаксис	<code>directio</code> размер off;
По умолчанию	<code>directio</code> off;
Контекст	http, server, location

Разрешает использовать флаги `O_DIRECT` (FreeBSD, Linux), `F_NOCACHE` (macOS) или функцию `directio()` (Solaris) при чтении файлов, размер которых больше либо равен указанному. Директива автоматически запрещает использование *sendfile* для данного запроса. Рекомендуется использовать для больших файлов:

```
directio 4m;
```

или при использовании *aio* в Linux.

directio_alignment

Синтаксис	<code>directio_alignment размер;</code>
По умолчанию	<code>directio_alignment 512;</code>
Контекст	http, server, location

Устанавливает выравнивание для *directio*. В большинстве случаев достаточно 512-байтового выравнивания, однако при использовании XFS под Linux его нужно увеличить до 4К.

disable_symlinks

Синтаксис	<code>disable_symlinks off;</code> <code>disable_symlinks on if_not_owner [from=часть];</code>
По умолчанию	<code>disable_symlinks off;</code>
Контекст	http, server, location

Определяет, как следует поступать с символическими ссылками при открытии файлов:

off	Символические ссылки в пути допускаются и не проверяются. Это стандартное поведение.
on	Если любой компонент пути является символической ссылкой, доступ к файлу запрещается.
if_not_owner	Доступ к файлу запрещается, если любой компонент пути является символической ссылкой, а ссылка и объект, на который она ссылается, имеют разных владельцев.
from=часть	При проверке символических ссылок (параметры on и if_not_owner) обычно проверяются все компоненты пути. Можно не проверять символические ссылки в начальной части пути, указав дополнительно параметр from=часть . В этом случае символические ссылки проверяются лишь начиная с компонента пути, который следует за заданной начальной частью. Если значение не является начальной частью проверяемого пути, путь проверяется целиком, как если бы этот параметр не был указан вовсе. Если значение целиком совпадает с именем файла, символические ссылки не проверяются. В значении параметра можно использовать переменные.

Пример:

```
disable_symlinks on from=$document_root;
```

Эта директива доступна только на системах, в которых есть интерфейсы `openat()` и `fstatat()`. К таким системам относятся современные версии FreeBSD, Linux и Solaris.

Предупреждение

Параметры **on** и **if_not_owner** требуют дополнительных затрат на обработку.

На системах, не поддерживающих операцию открытия каталогов только для поиска, для использования этих параметров требуется, чтобы рабочие процессы имели право читать все проверяемые каталоги.

Примечание

Модули AutoIndex, Random Index и DAV в настоящий момент игнорируют эту директиву.

early_hints

Синтаксис	early_hints <i>строка</i> ...;
По умолчанию	—
Контекст	http, server, location

Задаёт условия, при которых ответ с кодом "103 Early Hints" будет передан клиенту. Такой ответ может быть возвращён проксируемыми и gRPC-бэкендами. Если значение хотя бы одного из строковых параметров непустое и не равно 0, то ответ будет передан:

```
map $http_sec_fetch_mode $early_hints {
    navigate $http2$http3;
}

server {
    ...
    location / {
        early_hints $early_hints;
        proxy_pass http://example.com;
    }
}
```

В значении параметров можно использовать переменные.

error_log_user_tag

Синтаксис	error_log_user_tag <i>значение</i> ;
По умолчанию	—
Контекст	http, server, location, limit_except

Добавляет тег, зависящий от запроса, в записи *error_log*. *Значение* может содержать переменные. Директива может задаваться несколько раз для добавления нескольких тегов. Теги используются в фильтрах **filter=tag:** директивы *error_log*.

error_page

Синтаксис	error_page <i>код</i> ... [= [<i>ответ</i>]] <i>uri</i> ;
По умолчанию	—
Контекст	http, server, location, if в location

Задаёт URI, который будет показываться для указанных ошибок. В значении *uri* можно использовать переменные.

Пример:

```
error_page 404          /404.html;
error_page 500 502 503 504 /50x.html;
```

При этом делается внутреннее перенаправление на указанный *uri*, а метод запроса клиента меняется на "GET" (для всех методов, отличных от "GET" и "HEAD").

Кроме того, можно поменять код ответа на другой, используя синтаксис вида **=ответ**, например:

```
error_page 404 =200 /empty.gif;
```

Если ошибочный ответ обрабатывается проксированным сервером или FastCGI/uwsgi/SCGI/gRPC-сервером, и этот сервер может вернуть разные коды ответов, например, 200, 302, 401 или 404, то можно выдавать возвращаемый им код:

```
error_page 404 = /404.php;
```

Если при внутреннем перенаправлении не нужно менять URI и метод, то можно передать обработку ошибки в именованный **location**:

```
location / {
    error_page 404 = @fallback;
}

location @fallback {
    proxy_pass http://backend;
}
```

Примечание

Если при обработке *uri* происходит ошибка, клиенту возвращается ответ с кодом последней случившейся ошибки.

Также существует возможность использовать перенаправления URL для обработки ошибок:

```
error_page 403          http://example.com/forbidden.html;
error_page 404 =301 http://example.com/notfound.html;
```

В этом случае по умолчанию клиенту возвращается код ответа 302. Его можно изменить только на один из кодов ответа, относящихся к перенаправлениям (301, 302, 303, 307 и 308).

etag

Синтаксис	etag on off;
По умолчанию	etag on;
Контекст	http, server, location

Разрешает или запрещает автоматическую генерацию поля **Etag** заголовка ответа для статических ресурсов.

http

Синтаксис	<code>http { ... }</code>
По умолчанию	—
Контекст	<code>main</code>

Предоставляет контекст конфигурационного файла, в котором указываются директивы HTTP-сервера.

if_modified_since

Синтаксис	<code>if_modified_since off exact before;</code>
По умолчанию	<code>if_modified_since exact;</code>
Контекст	<code>http, server, location</code>

Определяет, как сравнивать время модификации ответа с временем в поле `If-Modified-Since` заголовка запроса:

<code>off</code>	ответ всегда считается изменившимся
<code>exact</code>	точное совпадение
<code>before</code>	время модификации ответа меньше или равно времени, заданному в поле <code>If-Modified-Since</code> заголовка запроса

ignore_invalid_headers

Синтаксис	<code>ignore_invalid_headers on off;</code>
По умолчанию	<code>ignore_invalid_headers on;</code>
Контекст	<code>http, server</code>

Если включено, Angie игнорирует поля заголовка с недопустимыми именами. Допустимыми считаются имена, состоящие из английских букв, цифр, дефисов и возможно знаков подчеркивания (последнее контролируется директивой `underscores_in_headers`).

Если директива указана на уровне `server`, то может использоваться значение из сервера по умолчанию.

internal

Синтаксис	<code>internal;</code>
По умолчанию	—
Контекст	<code>location</code>

Указывает, что `location` может использоваться только для внутренних запросов. Для внешних запросов будет возвращаться ошибка 404 (Not Found) в контексте данного `location`, что позволяет перенаправить такие запросы с помощью `error_page`. Внутренними запросами являются:

- запросы, перенаправленные директивами `error_page`, `index`, `random_index` и `try_files`;

- запросы, перенаправленные с помощью поля **X-Accel-Redirect** заголовка ответа вышестоящего сервера;
- подзапросы, формируемые командой **include virtual** модуля SSI, директивами модуля Addition, а также директивами *auth_request* и *mirror*;
- запросы, измененные директивой *rewrite*;
- запросы, перенаправленные директивой *goto*.

Пример:

```
error_page 404 /404.html;

location = /404.html {
    internal;
}
```

Благодаря тому, что ошибка 404 возвращается в контексте **location** с директивой **internal**, можно перенаправлять внешние запросы в другое место. Это позволяет использовать один префикс как для внешнего, так и для внутреннего запроса, но с разной обработкой, например:

```
location /path {
    internal;
    error_page 404 =@external;

    proxy_pass https://internal;
}

location @external {
    proxy_pass https://external;
}
```

Здесь внешний запрос **GET /path** будет проксирован на **https://external/path**, а такой же внутренний запрос будет проксирован на **https://internal/path**.

i Примечание

Для предотвращения заикливания, которое может возникнуть при использовании некорректных конфигураций, количество внутренних перенаправлений ограничено десятью. По достижении этого ограничения будет возвращена ошибка 500 (Internal Server Error). В таком случае в лог-файле ошибок можно увидеть сообщение **rewrite or internal redirection cycle**.

keepalive_disable

Синтаксис	keepalive_disable none <i>браузер</i> ...;
По умолчанию	keepalive_disable msie6;
Контекст	http, server, location

Запрещает соединения keep-alive с некорректно ведущими себя браузерами. Параметры *браузер* указывают, на какие браузеры это распространяется.

none	разрешает соединения keep-alive со всеми браузерами
msie6	запрещает соединения keep-alive со старыми версиями MSIE после получения запроса POST
safari	запрещает соединения keep-alive с Safari и подобными им браузерами на macOS и подобных ей ОС

keepalive_requests

Синтаксис	keepalive_requests <i>число</i> ;
По умолчанию	keepalive_requests 1000;
Контекст	http, server, location

Задаёт максимальное число запросов, которые можно сделать по одному keep-alive соединению. После того, как сделано максимальное число запросов, соединение закрывается.

Периодическое закрытие соединений необходимо для освобождения памяти, выделенной под конкретные соединения. Поэтому использование слишком большого максимального числа запросов может приводить к чрезмерному потреблению памяти и не рекомендуется.

keepalive_time

Синтаксис	keepalive_time <i>время</i> ;
По умолчанию	keepalive_time 1h;
Контекст	http, server, location

Ограничивает максимальное время, в течение которого могут обрабатываться запросы в рамках соединения keep-alive. По достижении заданного времени соединение закрывается после обработки очередного запроса.

keepalive_timeout

Синтаксис	keepalive_timeout <i>таймаут</i> [<i>заголовок_таймаута</i>];
По умолчанию	keepalive_timeout 75s;
Контекст	http, server, location

<i>таймаут</i>	задаёт время, в течение которого keep-alive соединение с клиентом не будет закрыто со стороны сервера
0	запрещает keep-alive соединения с клиентами

Второй, *необязательный*, параметр задаёт значение в поле **Keep-Alive: timeout=время** заголовка ответа. Два параметра могут отличаться друг от друга.

Поле **Keep-Alive: timeout=время** заголовка понимают Mozilla и Konqueror. MSIE сам закрывает keep-alive соединение примерно через 60 секунд.

Примечание

Эта директива применяется к соединениям с клиентами. Таймаут для неактивных соединений с проксируемыми серверами группы задается отдельно директивой `keepalive_timeout` в блоке `upstream`.

`large_client_header_buffers`

Синтаксис	<code>large_client_header_buffers</code> <i>количество размер</i> ;
По умолчанию	<code>large_client_header_buffers 4 8k</code> ;
Контекст	<code>http, server</code>

Задаёт максимальное число и размер буферов для чтения большого заголовка запроса клиента. Строка запроса не должна превышать размера одного буфера, иначе клиенту возвращается ошибка 414 (Request-URI Too Large). Поле заголовка запроса также не должно превышать размера одного буфера, иначе клиенту возвращается ошибка 400 (Bad Request). Буферы выделяются только по мере необходимости. По умолчанию размер одного буфера равен 8К байт. Если по окончании обработки запроса соединение переходит в состояние `keep-alive`, эти буферы освобождаются.

Если директива указана на уровне `server`, то может использоваться значение из сервера по умолчанию.

`limit_except`

Синтаксис	<code>limit_except</code> <i>метод1</i> [<i>метод2...</i>] { ... };
По умолчанию	—
Контекст	<code>location</code>

Ограничивает HTTP-методы, доступные внутри `location`. Параметр *метод* может быть одним из GET, HEAD, POST, PUT, DELETE, MKCOL, COPY, MOVE, OPTIONS, PROPFIND, PROPPATCH, LOCK, UNLOCK или PATCH. Если разрешен метод GET, то метод HEAD также будет разрешен. Доступ к остальным методам может быть ограничен при помощи директив модулей Access и Auth Basic.

```
limit_except GET {
    allow 192.168.1.0/32;
    deny all;
}
```

Примечание

Ограничение в примере действует для всех методов, **кроме** GET и HEAD.

`limit_rate`

Синтаксис	<code>limit_rate</code> <i>скорость</i> ;
По умолчанию	<code>limit_rate 0</code> ;
Контекст	<code>http, server, location, if в location</code>

Ограничивает скорость передачи ответа клиенту. Скорость задается в байтах в секунду. Значение 0 отключает ограничение скорости. Ограничение устанавливается на запрос, поэтому, если клиент одновременно откроет два соединения, суммарная скорость будет вдвое выше заданного ограничения.

В значении параметра можно использовать переменные. Это может быть полезно в случаях, когда скорость нужно ограничивать в зависимости от какого-либо условия:

```
map $slow $rate {
    1      4k;
    2      8k;
}

limit_rate $rate;
```

Ограничение скорости можно также задать в переменной `$limit_rate`, однако использовать данный метод не рекомендуется:

```
server {

    if ($slow) {
        set $limit_rate 4k;
    }

}
```

Кроме того, ограничение скорости может быть задано в поле **X-Accel-Limit-Rate** заголовка ответа проксированного сервера. Эту возможность можно запретить с помощью директив `proxy_ignore_headers`, `fastcgi_ignore_headers`, `uwsgi_ignore_headers` и `scgi_ignore_headers`.

limit_rate_after

Синтаксис	limit_rate_after размер;
По умолчанию	limit_rate_after 0;
Контекст	http, server, location, if в location

Задаёт начальный объём данных, после передачи которого начинает ограничиваться скорость передачи ответа клиенту. В значении параметра можно использовать переменные.

Пример:

```
location /flv/ {
    flv;
    limit_rate_after 500k;
    limit_rate      50k;
}
```

lingering_close

Синтаксис	lingering_close on always off;
По умолчанию	lingering_close on;
Контекст	http, server, location

Управляет закрытием соединений с клиентами.

on	Angie будет ждать и обрабатывать дополнительные данные, поступающие от клиента, перед полным закрытием соединения, но только если эвристика указывает на то, что клиент может еще послать данные.
always	Angie всегда будет ждать и обрабатывать дополнительные данные, поступающие от клиента.
off	Angie не будет ждать поступления дополнительных данных и сразу же закроет соединение. Это поведение нарушает протокол и поэтому не должно использоваться без необходимости.

Для управления закрытием HTTP/2-соединений директива должна быть задана на уровне *server*.

lingering_time

Синтаксис	lingering_time время;
По умолчанию	lingering_time 30s;
Контекст	http, server, location

Если действует *lingering_close*, эта директива задает максимальное время, в течение которого Angie будет обрабатывать (читать и игнорировать) дополнительные данные, поступающие от клиента. По прошествии этого времени соединение будет закрыто, даже если будут еще данные.

lingering_timeout

Синтаксис	lingering_timeout время;
По умолчанию	lingering_timeout 5s;
Контекст	http, server, location

Если действует *lingering_close*, эта директива задает максимальное время ожидания поступления дополнительных данных от клиента. Если в течение этого времени данные не были получены, соединение закрывается. В противном случае данные читаются и игнорируются, и Angie снова ждет поступления данных. Цикл "ждать-читать-игнорировать" повторяется, но не дольше чем задано директивой *lingering_time*.

При постепенном завершении клиентские постоянные соединения закрываются только в случае, если они неактивны не менее времени, заданного в *lingering_timeout*.

Примечание

В nginx аналогичная директива называется *keepalive_min_timeout*.

listen

Синтаксис	<pre>listen <i>адрес[:порт]</i> [default_server] [ssl] [http2 quic] [proxy_protocol] [setfib=<i>число</i>] [fastopen=<i>число</i>] [backlog=<i>число</i>] [rcvbuf=<i>размер</i>] [sndbuf=<i>размер</i>] [accept_filter=<i>филтър</i>] [deferred] [bind] [ipv6only=on off] [reuseport] [multipath] [so_keepalive=on off][keepidle]:[keepintvl]:[keepcnt]]; listen <i>порт</i> [default_server] [ssl] [http2 quic] [proxy_protocol] [setfib=<i>число</i>] [fastopen=<i>число</i>] [backlog=<i>число</i>] [rcvbuf=<i>размер</i>] [sndbuf=<i>размер</i>] [accept_filter=<i>филтър</i>] [deferred] [bind] [ipv6only=on off] [reuseport] [multipath] [so_keepalive=on off][keepidle]:[keepintvl]:[keepcnt]]; listen unix:<i>путь</i> [default_server] [ssl] [http2 quic] [proxy_protocol] [backlog=<i>число</i>] [rcvbuf=<i>размер</i>] [sndbuf=<i>размер</i>] [accept_filter=<i>филтър</i>] [deferred] [bind] [so_keepalive=on off][keepidle]:[keepintvl]:[keepcnt]];</pre>
По умолчанию	<code>listen *:80 *:8000;</code>
Контекст	server

Задаёт *адрес* и *порт* для прослушивающего сокета или путь к UNIX-доменному сокету, на котором сервер будет принимать запросы. *Адрес* также может быть именем хоста, например:

```
listen 127.0.0.1:8000;
listen 127.0.0.1;
listen 8000;
listen *:8000;
listen localhost:8000;
```

IPv6-адреса указываются в квадратных скобках:

```
listen [::]:8000;
listen [::1];
```

Диапазоны портов задаются при помощи указания первого и последнего порта через дефис:

```
listen 127.0.0.1:8000-8009;
listen 8000-8009;
```

UNIX-доменные сокеты задаются с префиксом `unix::`

```
listen unix:/var/run/angie.sock;
```

Можно указать *адрес* и *порт* вместе, только *адрес* или только *порт*. Если какие-либо параметры опущены, применяются следующие правила:

- Если указан только *адрес*, используется порт 80.
- Если указан только *порт*, Angie будет слушать на всех доступных интерфейсах IPv4 (и IPv6, если включен). Сервером по умолчанию для пары `*:порт` становится первый блок `server`, указавший эту пару.
- Если директива не указана вовсе, Angie использует `*:80`, если запущен с правами суперпользователя, или `*:8000` в противном случае.

<code>default_server</code>	сервер, в котором указан этот параметр, будет сервером по умолчанию для указанной пары <i>адрес:порт</i> (вместе они образуют <i>слушающий сокет</i>). Если же директив с параметром <code>default_server</code> нет, сервером по умолчанию для слушающего сокета будет первый в конфигурации сервер, обслуживающий этот сокет. Такой сервер по умолчанию действует только для соединений, отнесенных к этой паре; см. как Angie сопоставляет соединение с парой.
<code>ssl</code>	указывает на то, что все соединения, принимаемые на данном слушающем сокете, должны работать в режиме SSL. Это позволяет задать компактную конфигурацию для сервера, работающего сразу в двух режимах — HTTP и HTTPS.
<code>http2</code>	позволяет принимать на слушающем сокете HTTP/2-соединения. Обычно, чтобы это работало, следует также указать параметр <code>ssl</code> , однако Angie можно также настроить и на прием HTTP/2-соединений без SSL. Устарело Используйте взамен директиву <code>http2</code>.
<code>quic</code>	позволяет принимать на этом порту QUIC-соединения. Для использования этой опции в Angie должен быть включен и настроен модуль HTTP/3. Когда <code>quic</code> включен, также можно указать <code>reuseport</code> , чтобы использовать несколько рабочих процессов.
<code>proxy_protocol</code>	указывает на то, что все соединения, принимаемые на данном слушающем сокете, должны использовать протокол PROXY.

В директиве `listen` можно также указать несколько дополнительных параметров, специфичных для связанных с сокетами системных вызовов. Следующие параметры можно задать в любой директиве `listen`, но только один раз для каждого слушающего сокета:

setfib= <i>число</i>	задает таблицу маршрутизации, FIB (параметр SO_SETFIB) для слушающего сокета. В настоящий момент это работает только на FreeBSD.
fastopen= <i>число</i>	включает "TCP Fast Open" для слушающего сокета и ограничивает максимальную длину очереди соединений, которые еще не завершили процесс трехстороннего рукопожатия.
⚠ Предупреждение Не включайте "TCP Fast Open", не убедившись, что сервер может адекватно обрабатывать многократное получение одного и того же SYN-пакета с данными.	
backlog= <i>число</i>	задает параметр backlog в вызове listen(), который ограничивает максимальный размер очереди ожидающих приема соединений. По умолчанию backlog устанавливается равным -1 для FreeBSD, DragonFly BSD и macOS, и 511 для других платформ.
rcvbuf= <i>размер</i> sndbuf= <i>размер</i>	задает размер буфера приема (параметр SO_RCVBUF) для слушающего сокета. задает размер буфера передачи (параметр SO_SNDBUF) для слушающего сокета.
accept_filter= <i>фи</i>	задает название accept-фильтра (параметр SO_ACCEPTFILTER) для слушающего сокета, который включается для фильтрации входящих соединений перед передачей их в accept(). Работает только на FreeBSD и NetBSD 5.0+. Можно использовать два фильтра: dataready и httpready.
deferred	указывает использовать отложенный accept() (параметр TCP_DEFER_ACCEPT сокета) на Linux.
bind	указывает, что для данного слушающего сокета нужно делать bind() отдельно. Это нужно потому, что если описаны несколько директив listen с одинаковым портом, но разными адресами, и одна из директив listen слушает на всех адресах для данного <i>порта</i> (*:порт), то Angie сделает bind() только на *:порт. Необходимо заметить, что в этом случае для определения адреса, на который пришло соединение, делается системный вызов getsockname(). Если же используются параметры setfib, fastopen, backlog, rcvbuf, sndbuf, accept_filter, deferred, ipv6only, reuseport, multipath или so_keepalive, то для данной пары <i>адрес:порт</i> всегда делается отдельный вызов bind().
ipv6only=on off	определяет (через параметр сокета IPV6_V6ONLY), будет ли слушающий на wildcard-адресе [:] IPv6-сокеты принимать только IPv6-соединения, или же одновременно IPv6- и IPv4-соединения. По умолчанию параметр включен. Установить его можно только один раз на старте.
reuseport	указывает, что нужно создавать отдельный слушающий сокет для каждого рабочего процесса (через параметр сокета SO_REUSEPORT для Linux 3.9+ и DragonFly BSD или SO_REUSEPORT_LB для FreeBSD 12+), позволяя ядру распределять входящие соединения между рабочими процессами. В настоящий момент это работает только на Linux 3.9+, DragonFly BSD и FreeBSD 12+.
⚠ Предупреждение Ненадлежащее использование параметра reuseport может быть небезопасно.	
multipath	включает прием соединений по протоколу Multipath TCP (MPTCP), поддерживаемому в ядре Linux с версии 5.6. Параметр несовместим с quick.

so_keepalive=on | off | [keepidle]:[keepintvl]:[keepcnt]

Конфигурирует для слушающего сокета поведение "TCP keepalive".

<code>''</code>	если параметр опущен, для сокета будут действовать настройки операционной системы
<code>on</code>	для сокета включается параметр <code>SO_KEEPALIVE</code>
<code>off</code>	для сокета параметр <code>SO_KEEPALIVE</code> выключается

Некоторые операционные системы поддерживают настройку параметров "TCP keepalive" на уровне сокета посредством параметров `TCP_KEEPIIDLE`, `TCP_KEEPIINTVL` и `TCP_KEEPCNT`. На таких системах (в настоящее время это Linux, NetBSD, DragonFly, FreeBSD и macOS) их можно сконфигурировать с помощью параметров `keepidle`, `keepintvl` и `keepcnt`. Один или два параметра могут быть опущены, в таком случае для соответствующего параметра сокета будут действовать стандартные системные настройки. Например,

```
so_keepalive=30m::10
```

установит таймаут бездействия (`TCP_KEEPIIDLE`) в 30 минут, для интервала проб (`TCP_KEEPIINTVL`) будет действовать стандартная системная настройка, а счетчик проб (`TCP_KEEPCNT`) будет равен 10.

Дополнительные параметры привязывают слушающий сокет к сетевому интерфейсу и задают для этого сокета алгоритм управления перегрузкой TCP:

<code>interface=имя</code>	привязывает слушающий сокет к указанному сетевому интерфейсу (параметр сокета <code>SO_BINDTODEVICE</code>)
<code>congestion=алгор</code>	задает алгоритм управления перегрузкой TCP (параметр сокета <code>TCP_CONGESTION</code>) для слушающего сокета; принимаемые соединения наследуют этот алгоритм. Параметр несовместим с <code>quic</code> и с UNIX-сокетами

При использовании любого из этих параметров для данной пары *адрес:порт* всегда выполняется отдельный вызов `bind()`.

Имя алгоритма управления перегрузкой не проверяется на этапе конфигурации: если алгоритм неизвестен или недоступен, при настройке слушающего сокета в лог однократно записывается сообщение уровня `alert`, а для сокета действует системный алгоритм по умолчанию.

Пример:

```
listen 127.0.0.1 default_server accept_filter=dataready backlog=1024;
```

location

Синтаксис	<code>location ([= ~ ~* ^~] uri @имя) + { ... }</code>
По умолчанию	—
Контекст	<code>server</code> , <code>location</code>

Устанавливает конфигурацию в зависимости от того, соответствует ли URI запроса какому-либо из выражений сопоставления.

Для сопоставления используется URI запроса в нормализованном виде (значение `$uri`), после декодирования текста, заданного в виде `%XX`, преобразования относительных элементов пути "." и ".." в реальные и возможной замены двух и более подряд идущих косых черт на одну.

Задать `location` можно префиксной строкой или регулярным выражением.

Регулярные выражения задаются с модификатором:

<code>~*</code>	Для поиска совпадения без учета регистра символов
<code>~</code>	С учетом регистра

Чтобы найти `location`, соответствующий запросу, вначале проверяются `location`'ы, заданные префиксными строками (префиксные `location`'ы). Среди них ищется `location` с совпадающим префиксом максимальной длины и запоминается.

Примечание

Для операционных систем, нечувствительных к регистру символов, таких как macOS, сравнение с префиксными строками производится без учета регистра. Однако сравнение ограничено только однобайтными `locale`'ями.

Затем проверяются регулярные выражения в порядке их следования в конфигурационном файле. Проверка регулярных выражений прекращается после первого же совпадения, и используется соответствующая конфигурация. Если совпадение с регулярным выражением не найдено, то используется конфигурация запомненного ранее префиксного `location`'а.

За некоторыми исключениями, о которых говорится ниже, блоки `location` могут быть вложенными.

Регулярные выражения могут создавать группы захвата, которые затем можно использовать в других директивах.

Если у совпавшего префиксного `location`'а указан модификатор `~`, то регулярные выражения не проверяются.

Кроме того, с помощью модификатора `=` можно задать точное совпадение URI и `location`. При точном совпадении поиск сразу же прекращается. Например, если запрос `/` случается часто, можно ускорить обработку таких запросов, указав `location =/`, так как поиск прекратится после первого же сравнения. Такой `location` не может иметь вложенные `location`, так как он задает полное совпадение.

После того как подходящий `location` выбран, запрос сопоставляется с файлом с помощью директивы `root` или `alias`, а `try_files` может проверять наличие файлов внутри него; общее описание см. в *Выбор локаций*.

Пример:

```
location =/ {
    #конфигурация А
}

location / {
    #конфигурация Б
}

location /documents/ {
    #конфигурация В
}

location ~~/images/ {
    #конфигурация Г
}

location ~*\.(gif|jpg|jpeg)$ {
    #конфигурация Д
}
```

- Для запроса `/` будет выбрана конфигурация А,
- для запроса `/index.html` — конфигурация Б,
- для запроса `/documents/document.html` — конфигурация В,

- для запроса `/images/1.gif` — конфигурация Г,
- а для запроса `/documents/1.jpg` — конфигурация Д.

Примечание

Если префиксный `location` задан с косой чертой в конце и включена директива `auto_redirect`, происходит следующее: На запрос с URI без косой черты в конце, в остальном совпадающий с префиксом, будет возвращено постоянное перенаправление с кодом 301, указывающее на URI запроса с добавленной в конце косой чертой.

Если задать `location` с точным совпадением URI, перенаправление не используется:

```
location /user/ {
    proxy_pass http://user.example.com;
}

location =/user {
    proxy_pass http://login.example.com;
}
```

Префикс `@` задает *именованный location*. Такой `location` не используется при обычной обработке запросов, а предназначен только для перенаправления в него запросов. Такие `location` не могут быть вложенными и не могут содержать вложенные `location`.

Комбинированные location

Для удобства несколько `location` с одинаковой конфигурацией можно записать компактно, перечислив в одном `location` сразу несколько выражений сопоставления и задав для них единую конфигурацию. Такие `location` называются *комбинированными*.

Если, например, предположить, что в предыдущем примере конфигурации А, Г и Д совпадают, то их можно записать с помощью комбинированного `location`:

```
location =/
    ~/images/
    ~*\.(gif|jpg|jpeg)$ {
    #общая конфигурация
}
```

Именованный `location` также может быть частью комбинированного:

```
location =/
    @named_combined {
    #...
}
```

Предупреждение

В комбинированных `location` между модификатором выражения сопоставления и самим выражением не может стоять пробел. Правильно: `location ~*/match(ing|es|er)$ ....`

Примечание

Сейчас комбинированные `location` не могут **непосредственно** содержать директивы `proxy_pass`, в которых задан URI, а также `api` и `alias`. При этом такие директивы можно

использовать в других `location`, вложенных в комбинированный.

`log_not_found`

Синтаксис	<code>log_not_found on off;</code>
По умолчанию	<code>log_not_found on;</code>
Контекст	<code>http, server, location</code>

Разрешает или запрещает записывать в `error_log` ошибки о том, что файл не найден.

`log_subrequest`

Синтаксис	<code>log_subrequest on off;</code>
По умолчанию	<code>log_subrequest off;</code>
Контекст	<code>http, server, location</code>

Разрешает или запрещает записывать в `access_log` подзапросы.

`max_headers`

Синтаксис	<code>max_headers <i>число</i>;</code>
Значение по умолчанию	<code>max_headers 1000;</code>
Контекст	<code>http, server</code>

Устанавливает максимальное количество полей заголовков запроса клиента. При превышении предела клиенту возвращается ошибка 400 (Bad Request).

Если эта директива задана на уровне `server`, может использоваться значение с сервера по умолчанию. Дополнительные сведения см. в разделе *Выбор виртуального сервера*.

`max_ranges`

Синтаксис	<code>max_ranges <i>число</i>;</code>
По умолчанию	—
Контекст	<code>http, server, location</code>

Ограничивает максимальное допустимое число диапазонов в запросах с указанием диапазона запрашиваемых байт (byte-range requests). Запросы, превышающие указанное ограничение, обрабатываются как если бы они не содержали указания диапазонов. По умолчанию число диапазонов не ограничено.

0	полностью запрещает поддержку диапазонов
---	--

merge_slashes

Синтаксис	<code>merge_slashes on off;</code>
По умолчанию	<code>merge_slashes on;</code>
Контекст	<code>http, server</code>

Разрешает или запрещает преобразование URI путем замены двух и более подряд идущих косых черт ("/") на одну.

Необходимо иметь в виду, что это преобразование необходимо для корректной проверки префиксных строк и регулярных выражений. Если его не делать, то запрос `//scripts/one.php` не попадет в

```
location /scripts/ { }
```

и может быть обслужен как статический файл. Поэтому он преобразуется к виду `/scripts/one.php`.

Запрет преобразования может понадобиться, если в URI используются имена, закодированные методом `base64`, в котором задействован символ `"/"`. Однако из соображений безопасности лучше избегать отключения преобразования.

Если директива указана на уровне `server`, то может использоваться значение из сервера по умолчанию.

msie_padding

Синтаксис	<code>msie_padding on off;</code>
По умолчанию	<code>msie_padding on;</code>
Контекст	<code>http, server, location</code>

Разрешает или запрещает добавлять в ответы для MSIE со статусом больше 400 комментариев для увеличения размера ответа до 512 байт.

msie_refresh

Синтаксис	<code>msie_refresh on off;</code>
По умолчанию	<code>msie_refresh off;</code>
Контекст	<code>http, server, location</code>

Разрешает или запрещает выдавать для MSIE клиентов refresh'ы вместо перенаправлений.

open_file_cache

Синтаксис	<code>open_file_cache off;</code> <code>open_file_cache max=N [inactive=время];</code>
По умолчанию	<code>open_file_cache off;</code>
Контекст	<code>http, server, location</code>

Задаёт кэш, в котором могут храниться:

- дескрипторы открытых файлов, информация об их размерах и времени модификации;
- информация о существовании каталогов;
- информация об ошибках поиска файла — "нет файла", "нет прав на чтение" и тому подобное.

Кэширование ошибок нужно разрешить отдельно директивой `open_file_cache_errors`.

max	задает максимальное число элементов в кэше; при переполнении кэша удаляются наименее востребованные элементы (LRU)
inactive	задает время, после которого элемент кэша удаляется, если к нему не было обращений в течение этого времени. По умолчанию: 60 секунд.
off	запрещает кэш.

Пример:

```
open_file_cache          max=1000 inactive=20s;
open_file_cache_valid    30s;
open_file_cache_min_uses 2;
open_file_cache_errors   on;
```

`open_file_cache_errors`

Синтаксис	<code>open_file_cache_errors on off;</code>
По умолчанию	<code>open_file_cache_errors off;</code>
Контекст	http, server, location

Разрешает или запрещает кэширование ошибок поиска файлов в `open_file_cache`.

`open_file_cache_events`

Синтаксис	<code>open_file_cache_events on off;</code>
По умолчанию	<code>open_file_cache_events off;</code>
Контекст	http, server, location

Разрешает использование событий ядра для проверки актуальности элементов `open_file_cache`. Директива работает только с методом kqueue. Обратите внимание, что только NetBSD 2.0+ и FreeBSD 6.0+ поддерживают события для произвольных типов файловых систем; остальные операционные системы поддерживают события только для основных файловых систем, таких как UFS или FFS.

`open_file_cache_min_uses`

Синтаксис	<code>open_file_cache_min_uses число;</code>
По умолчанию	<code>open_file_cache_min_uses 1;</code>
Контекст	http, server, location

Задает минимальное число обращений к файлу в течение времени, заданного параметром `inactive` директивы `open_file_cache`, необходимых для того, чтобы дескриптор файла оставался открытым в кэше.

open_file_cache_valid

Синтаксис	<code>open_file_cache_valid</code> время;
По умолчанию	<code>open_file_cache_valid 60s</code> ;
Контекст	http, server, location

Определяет время, через которое следует проверять актуальность информации об элементе в *open_file_cache*.

output_buffers

Синтаксис	<code>output_buffers</code> <i>число</i> <i>размер</i> ;
По умолчанию	<code>output_buffers 2 32k</code> ;
Контекст	http, server, location

Задаёт *число* и *размер* буферов, используемых при чтении ответа с диска.

port_in_redirect

Синтаксис	<code>port_in_redirect on off</code> ;
По умолчанию	<code>port_in_redirect on</code> ;
Контекст	http, server, location

Разрешает или запрещает указывать порт в абсолютных перенаправлениях, выдаваемых Angie.

Использование в перенаправлениях основного имени сервера управляется директивой *server_name_in_redirect*.

postpone_output

Синтаксис	<code>postpone_output</code> размер;
По умолчанию	<code>postpone_output 1460</code> ;
Контекст	http, server, location

Если это возможно, то отправка данных клиенту будет отложена пока Angie не накопит по крайней мере указанное количество байт для отправки.

0	запрещает отложенную отправку данных
---	--------------------------------------

read_ahead

Синтаксис	<code>read_ahead</code> размер;
По умолчанию	<code>read_ahead 0</code> ;
Контекст	http, server, location

Задаёт ядру размер предчтения при работе с файлами.

На Linux используется системный вызов `posix_fadvise(0, 0, 0, POSIX_FADV_SEQUENTIAL)`, поэтому параметр размер там игнорируется.

На FreeBSD используется системный вызов `fcntl(O_READAHEAD, размер)`, появившийся во FreeBSD 9.0-CURRENT.

recursive_error_pages

Синтаксис	<code>recursive_error_pages on off;</code>
По умолчанию	<code>recursive_error_pages off;</code>
Контекст	http, server, location

Разрешает или запрещает делать несколько перенаправлений через директиву `error_page`. Число таких перенаправлений ограничено.

request_pool_size

Синтаксис	<code>request_pool_size размер;</code>
По умолчанию	<code>request_pool_size 4k;</code>
Контекст	http, server

Позволяет производить точную настройку выделения памяти под конкретные запросы. Эта директива не оказывает существенного влияния на производительность, и ее не следует использовать.

reset_timeout_connection

Синтаксис	<code>reset_timeout_connection on off;</code>
По умолчанию	<code>reset_timeout_connection off;</code>
Контекст	http, server, location

Разрешает или запрещает сброс соединений по таймауту, а также при закрытии соединений с помощью нестандартного кода 444. Сброс делается следующим образом. Перед закрытием сокета для него задается параметр `SO_LINGER` с таймаутом 0. После этого при закрытии сокета клиенту отсылается `TCP RST`, а вся память, связанная с этим сокетом, освобождается. Это позволяет избежать длительного нахождения уже закрытого сокета в состоянии `FIN_WAIT1` с заполненными буферами.

Примечание

соединения keep-alive по истечении таймаута закрываются обычным образом.

resolver

Синтаксис	<code>resolver [адрес ...] [conf [=файл]] [valid=время] [ipv4=on off] [ipv6=on off] [status_zone=зона];</code> <code>resolver off;</code>
По умолчанию	<code>resolver conf;</code>
Контекст	http, server, location, upstream

Задаёт серверы DNS, используемые для преобразования имен вышестоящих серверов в адреса, например:

```
resolver 127.0.0.53 [::1]:5353;
```

Адрес может быть указан в виде доменного имени или IP-адреса, и необязательного порта. Если порт не указан, используется порт 53. Серверы DNS опрашиваются циклически.

Необходимо указать хотя бы один адрес DNS-сервера или параметр `conf`, кроме формы `resolver off`;

Примечание

Рекомендуется использовать локальный доверенный резолвер, например 127.0.0.53 (systemd-resolved), а не публичный (например, 8.8.8.8). Публичные резолверы раскрывают DNS-запросы третьим сторонам и повышают риск атак с подменой кэша.

Примечание

Значение директивы наследуется вложенными блоками и может быть переопределено в них при необходимости. В пределах одного блока директиву допустимо указывать только один раз; повторное указание является ошибкой конфигурации.

Включает разрешение DNS-имен для вышестоящих серверов, задаваемых через переменные или настроенных с параметром `resolve`, а также для работы ACME-клиента и прикрепления OSCP-ответов.

Помимо явно заданных адресов серверов директива распознает:

<code>off</code>	явно отключает разрешение DNS-имен
<code>conf [=<i>файл</i>]</code>	считывает адреса DNS-серверов из указанного файла или из <code>/etc/resolv.conf</code> , если файл не указан

Файл может содержать строки `nameserver` в том же формате, что использует системный резолвер:

```
nameserver 127.0.1.4
nameserver 127.0.1.5
```

Адреса, считанные из файла, добавляются к адресам, заданным явно в директиве, и все они затем опрашиваются совместно циклическим образом. Если файл отсутствует, недоступен для чтения или не содержит адресов, а явно ни один сервер не задан, используется сервер по умолчанию 127.0.0.1:53.

В процессе работы Angie перечитывает файл при его изменении, обновляя набор DNS-серверов.

По умолчанию Angie кэширует каждый ответ на время, заданное значением TTL в DNS-ответе. Необязательный параметр `valid` позволяет это переопределить:

<code>valid</code>	необязательный параметр, позволяет переопределить срок кэширования ответа
--------------------	---

```
resolver 127.0.0.53 [::1]:5353 valid=30s;
```

По умолчанию Angie будет искать как IPv4-, так и IPv6-адреса при преобразовании имен в адреса.

ipv4=off	запрещает поиск IPv4-адресов
ipv6=off	запрещает поиск IPv6-адресов

status_zone	необязательный параметр; включает сбор метрик запросов и ответов DNS-сервера в указанной зоне, отображая их в <code>/status/resolvers/<зона></code> , вкладке «DNS Resolvers» и в выводе Prometheus. Без него эти метрики не собираются, и предупреждение не выводится
-------------	--

Совет

Для предотвращения DNS-спуфинга рекомендуется использовать DNS-серверы в защищенной доверенной локальной сети.

resolver_timeout

Синтаксис	<code>resolver_timeout</code> время;
По умолчанию	<code>resolver_timeout 30s</code> ;
Контекст	http, server, location, upstream

Задаёт таймаут для преобразования имени в адрес, например:

```
resolver_timeout 5s;
```

root

Синтаксис	<code>root</code> <i>путь</i> ;
По умолчанию	<code>root html</code> ;
Контекст	http, server, location, if в location

Задаёт корневой каталог для запросов. Например, при такой конфигурации

```
location /i/ {
    root /data/w3;
}
```

в ответ на запрос `/i/top.gif` будет отдан файл `/data/w3/i/top.gif`.

В значении параметра путь можно использовать переменные, кроме `$document_root` и `$realpath_root`.

Путь к файлу формируется путем простого добавления URI к значению директивы `root`. Если же URI необходимо поменять, следует воспользоваться директивой *alias*. Завершающая косая черта в значении параметра *путь* отбрасывается, поскольку URI добавляется вместе со своей начальной косой чертой.

satisfy

Синтаксис	satisfy all any;
По умолчанию	satisfy all;
Контекст	http, server, location

Разрешает доступ, если его разрешают все (**all**) или хотя бы один (**any**) из модулей Access, Auth Basic или Auth Request.

```
location / {
    satisfy any;

    allow 192.168.1.0/32;
    deny all;

    auth_basic "closed site";
    auth_basic_user_file conf/htpasswd;
}
```

send_lowat

Синтаксис	send_lowat размер;
По умолчанию	send_lowat 0;
Контекст	http, server, location

При установке этой директивы в ненулевое значение Angie будет пытаться минимизировать число операций отправки на клиентских сокетах либо при помощи флага **NOTE_LOWAT** метода `kqueue`, либо при помощи параметра сокета **SO_SNDLOWAT**. В обоих случаях будет использован указанный размер.

send_timeout

Синтаксис	send_timeout время;
По умолчанию	send_timeout 60s;
Контекст	http, server, location

Задаёт таймаут при передаче ответа клиенту. Таймаут устанавливается не на всю передачу ответа, а только между двумя операциями записи. Если по истечении этого времени клиент ничего не примет, соединение будет закрыто.

sendfile

Синтаксис	sendfile on off;
По умолчанию	sendfile off;
Контекст	http, server, location, if в location

Разрешает или запрещает использовать **sendfile()**.

Возможно использование *aio* для подгрузки данных для **sendfile()**:

```
location /video/ {
    sendfile      on;
    tcp_nopush    on;
    aio           on;
}
```

В такой конфигурации функция `sendfile()` вызывается с флагом `SF_NODISKIO`, в результате чего она не блокируется на диске, а сообщает об отсутствии данных в памяти. После этого Angie иницирует асинхронную подгрузку данных, читая один байт. При этом ядро FreeBSD подгружает в память первые 128K байт файла, однако при последующих чтениях файл подгружается частями только по 16K. Изменить это можно с помощью директивы `read_ahead`.

sendfile_max_chunk

Синтаксис	<code>sendfile_max_chunk</code> размер;
По умолчанию	<code>sendfile_max_chunk 2m</code> ;
Контекст	http, server, location

Ограничивает объем данных, который может передан за один вызов `sendfile()`. Без этого ограничения одно быстрое соединение может целиком захватить рабочий процесс.

server

Синтаксис	<code>server { ... }</code>
По умолчанию	—
Контекст	http

Задаёт конфигурацию для виртуального сервера. Четкого разделения виртуальных серверов на IP-based (на основании IP-адреса) и name-based (на основании поля "Host" заголовка запроса) нет. Вместо этого директивами `listen` описываются все адреса и порты, на которых нужно принимать соединения для этого сервера, а в директиве `server_name` указываются все имена серверов.

Подробнее: Как обрабатываются запросы

server_name

Синтаксис	<code>server_name имя ...;</code>
По умолчанию	<code>server_name "";</code>
Контекст	server

Задаёт имена виртуального сервера, например:

```
server {
    server_name example.com www.example.com;
}
```

Первое имя становится основным именем сервера.

В именах серверов можно использовать звездочку ("`*`") для замены первой или последней части имени:

```
server {
    server_name example.com *.example.com www.example.*;
}
```

Такие имена называются именами с маской.

Два первых вышеприведенных имени можно объединить в одно:

```
server {
    server_name .example.com;
}
```

В качестве имени сервера можно также использовать регулярное выражение, указав перед ним тильду ("~"):

```
server {
    server_name ~^www\d+\.example\.com$ www.example.com;
}
```

Регулярное выражение может содержать группы захвата, которые могут затем использоваться в других директивах:

```
server {
    server_name ~^(www\.)?(.+)$;

    location / {
        root /sites/$2;
    }
}

server {
    server_name _;

    location / {
        root /sites/default;
    }
}
```

Именованные группы захвата в регулярном выражении создают переменные, которые могут затем использоваться в других директивах:

```
server {
    server_name ~^(www\.)?(<domain>.+)$;

    location / {
        root /sites/$domain;
    }
}

server {
    server_name _;

    location / {
        root /sites/default;
    }
}
```

Примечание

Если параметр директивы задан как `$hostname`, используется имя хоста веб-сервера.

Можно также указать пустое имя сервера (`""`):

```
server {
    server_name www.example.com "";
}
```

При поиске виртуального сервера по имени, которому соответствует несколько указанных вариантов (например, одновременно подходят и имя с маской, и регулярное выражение), будет выбран первый подходящий вариант в следующем порядке приоритета:

- точное имя;
- самое длинное имя с маской в начале, например `*.example.com`;
- самое длинное имя с маской в конце, например `mail.*`;
- первое подходящее регулярное выражение (в порядке следования в конфигурации), в том числе пустое имя.

Предупреждение

Чтобы использовать `server_name` с TLS, необходима терминация TLS-соединения. Эта директива сопоставляется с `Host` в HTTP-запросе, поэтому рукопожатие должно быть закончено, а соединение расшифровано.

`server_name_in_redirect`

Синтаксис	<code>server_name_in_redirect on off;</code>
По умолчанию	<code>server_name_in_redirect off;</code>
Контекст	<code>http, server, location</code>

Разрешает или запрещает использовать в абсолютных перенаправлениях, выдаваемых Angie, основное имя сервера, задаваемое директивой `server_name`.

<code>on</code>	используется основное имя сервера, задаваемое директивой <code>server_name</code>
<code>off</code>	используется имя, указанное в поле "Host" заголовка запроса. Если же этого поля нет, то используется IP-адрес сервера.

Использование в перенаправлениях порта управляется директивой `port_in_redirect`.

`server_names_hash_bucket_size`

Синтаксис	<code>server_names_hash_bucket_size размер;</code>
По умолчанию	<code>server_names_hash_bucket_size 32 64 128;</code>
Контекст	<code>http</code>

Задаёт размер корзины в хэш-таблицах имен серверов. Значение по умолчанию зависит от размера строки кэша процессора. Подробнее настройка хэш-таблиц обсуждается отдельно.

server_names_hash_max_size

Синтаксис	<code>server_names_hash_max_size</code> размер;
По умолчанию	<code>server_names_hash_max_size 512;</code>
Контекст	http

Задаёт максимальный размер хэш-таблиц имен серверов. Подробнее настройка хэш-таблиц обсуждается отдельно.

server_tokens

Синтаксис	<code>server_tokens on off build строка;</code>
По умолчанию	<code>server_tokens on;</code>
Контекст	http, server, location

Разрешает или запрещает указывать версию Angie на страницах ошибок и в поле заголовка ответа **Server**.

Если указан параметр `build`, то вместе с версией будет также указано имя сборки, заданное соответствующим параметром скрипта.

Директива может быть задана *строкой*, которая может также содержать переменные. Тогда на страницах ошибок и в поле заголовка ответа **Server** вместо имени сервера, версии и имени сборки будет указываться значение этой строки с подставленными переменными. Пустая *строка* запрещает выдачу поля **Server**.

status_zone

Синтаксис	<code>status_zone off зона ключ zone=зона[:число];</code>
По умолчанию	—
Контекст	server, location, if в location

Выделяет зону разделяемой памяти для сбора метрик `/status/http/location_zones/<зона>` и `/status/http/server_zones/<зона>`.

Несколько контекстов **server** могут совместно использовать одну и ту же зону для сбора данных; особое значение `off` выключает сбор данных во вложенных блоках **location**.

Синтаксис с одним значением *зоны* объединяет все метрики для текущего контекста в одну зону разделяемой памяти:

```
server {
    listen 80;
    server_name *.example.com;

    status_zone single;
    # ...
}
```

Альтернативный синтаксис позволяет задавать следующие параметры:

<i>ключ</i>	Строка с переменными, значение которой определяет группировку запросов в зоне. Все запросы, дающие одинаковые значения после подстановки, объединяются в одну группу. Если подстановка возвращает пустое значение, метрики не обновляются.
<i>зона</i>	Имя зоны разделяемой памяти.
<i>число</i> (необязательный)	Максимальное количество отдельных групп для сбора метрик. Если новые значения <i>ключа</i> превышают этот лимит, они объединяются в группу <i>zone</i> . Значение по умолчанию — 1.

В следующем примере все запросы с одинаковым значением `$host` группируются в `host_zone`. Метрики собираются отдельно для каждого уникального значения `$host` до тех пор, пока количество групп метрик не достигнет 10. После этого любые новые значения `$host` будут добавляться в группу `host_zone`:

```
server {
    listen 80;
    server_name *.example.com;

    status_zone $host zone=host_zone:10;

    location / {
        proxy_pass http://example.com;
    }
}
```

Результирующие метрики разделяются по отдельным хостам в выводе API.

Примечание

Эти метрики собираются, только если задан `status_zone`. Без него сервер или location не отображается в `/status/http/server_zones/<зона>`, `/status/http/location_zones/<зона>`, виджете «HTTP Zones» и в выводе Prometheus, и предупреждение при этом не выводится. См. пример конфигурации.

subrequest_output_buffer_size

Синтаксис	<code>subrequest_output_buffer_size</code> размер;
По умолчанию	<code>subrequest_output_buffer_size 4k 8k;</code>
Контекст	<code>http</code> , <code>server</code> , <code>location</code>

Задаёт размер буфера, используемого для хранения тела ответа подзапроса. По умолчанию размер одного буфера равен размеру страницы памяти. В зависимости от платформы это или 4К, или 8К, однако его можно сделать меньше.

Примечание

Директива применима только для подзапросов, тело ответа которых сохраняется в памяти. Например, подобные подзапросы создаются при помощи SSI.

tcp_nodelay

Синтаксис	<code>tcp_nodelay on off;</code>
По умолчанию	<code>tcp_nodelay on;</code>
Контекст	<code>http, server, location</code>

Разрешает или запрещает использование параметра `TCP_NODELAY`. Параметр включается при переходе соединения в состояние `keep-alive`. Также, он включается на `SSL`-соединениях, при небуферизованном проксировании и при проксировании `WebSocket`.

tcp_nopush

Синтаксис	<code>tcp_nopush on off;</code>
По умолчанию	<code>tcp_nopush off;</code>
Контекст	<code>http, server, location</code>

Разрешает или запрещает использование параметра сокета `TCP_NOPUSH` во `FreeBSD` или `TCP_CORK` в `Linux`. Параметр включается только при использовании *sendfile*. Включение параметра позволяет:

- передавать заголовок ответа и начало файла в одном пакете в `Linux` и во `FreeBSD 4.*`;
- передавать файл полными пакетами.

time_format

Синтаксис	<code>time_format \$переменная формат;</code>
По умолчанию	—
Контекст	<code>http</code>

Определяет *\$переменную*, содержащую текущее локальное время, отформатированное в соответствии со строкой *формата*, которая поддерживает спецификаторы в стиле `strftime`, а также `%L` для миллисекунд. Директива полезна, когда возможностей `$time_local` и `$time_iso8601` недостаточно — например, если требуется точность до миллисекунд:

```
time_format $time_ms "%Y-%m-%dT%H:%M:%S.%LZ";
log_format tf '$time_ms $remote_addr "$request" $status';
```

При таком формате переменная `$time_ms` принимает значение вида `2024-01-15T14:34:56.789+03:00`.

Попытка переопределить переменную, не допускающую переопределения, например `$time_local`, или повторно объявить `time_format` с тем же именем приводит к ошибке загрузки конфигурации вида: `"the duplicate "... variable"`.

Строка *формата* поддерживает следующие спецификаторы; все они вычисляются в том же часовом поясе, что и `$time_local`:

%Y	год, 4 цифры
%y	год, 2 цифры
%m	месяц, 01-12
%d	день месяца, 01-31
%e	день месяца, дополненный пробелом, 1-31
%H	час, 00-23
%I	час, 01-12
%M	минута, 00-59
%S	секунда, 00-59
%L	миллисекунды, 000-999
%p	обозначение AM/PM в верхнем регистре, AM / PM
%P	то же самое в нижнем регистре, am / pm
%a	сокращенное название дня недели, Sun-Sat
%A	полное название дня недели, Sunday-Saturday
%b, %h	сокращенное название месяца, Jan-Dec
%B	полное название месяца, January-December
%z	смещение часового пояса, +0300
%Z	смещение часового пояса в формате ISO 8601, +03:00
%n	перевод строки
%t	табуляция
%%	символ %

Примечание

Нераспознанный спецификатор копируется в вывод без изменений, включая ведущий символ %; одиночный символ % в самом конце строки *формата* отбрасывается.

try_files

Синтаксис	<code>try_files <i>файл</i> ... <i>uri</i>;</code> <code>try_files <i>файл</i> ... =<i>код</i>;</code>
По умолчанию	—
Контекст	server, location

Проверяет существование файлов в заданном порядке и использует для обработки запроса первый найденный файл, причем обработка делается в контексте этого же *location*'а. Путь к файлу строится из параметра *файл* в соответствии с директивами *root* и *alias*. С помощью косой черты в конце имени можно проверить существование каталога, например, *\$uri/*. В случае, если ни один файл не найден, производится внутреннее перенаправление на *uri*, заданный последним параметром.

В параметрах *файл* и *uri* можно использовать переменные.

Например:

```
location /images/ {
    try_files $uri /images/default.gif;
}

location = /images/default.gif {
    expires 30s;
}
```

Последний параметр может быть URI для внутреннего перенаправления, ссылкой на именованный location (например, @drupal) или кодом ответа в форме =код (например, =404):


```
location / {
    try_files $uri $uri/index.html $uri.html =404;
}
```

Следует помнить, что чрезмерное использование директивы `try_files` увеличивает число системных вызовов, что может негативно сказаться на производительности.

Так, не следует использовать `try_files` для формирования поведения, фактически дублирующего поведение по умолчанию, например:

```
location /bad_pattern {

    # try_files $uri $uri/ =404; # не рекомендуется!
}
```

Также не следует использовать `try_files` исключительно для перенаправления при отсутствии файла. Дело в том, что директива `try_files` имеет две особенности:

- Во-первых, она проверяет существование каждого файла, что увеличивает нагрузку на систему.
- Во-вторых, любые ошибки открытия файла (например, `too many open files`, ошибки прав доступа) также считаются отсутствием файла и вызывают переход к запасному обработчику, что может привести к подмене ошибок 5xx успешными ответами и некорректному кэшированию.

Так, на практике можно встретить следующую проблемную конструкцию:

```
location / {
    try_files $uri $uri/ @drupal; # не рекомендуется!
}
```

Ее проблема в том, что единственная цель здесь — перенаправление. Использование `try_files` приводит к перечисленным выше недостаткам, но не дает никаких преимуществ, поскольку проверка существования файлов не нужна. Правильное решение — использовать директиву `error_page`, которая лишена этих недостатков:

```
error_page 404 = @drupal;
log_not_found off;
```

Напротив, в следующем примере:

```
location ~ /\.php$ {
    try_files $uri @drupal;

    fastcgi_pass ...;

    fastcgi_param SCRIPT_FILENAME /path/to$fastcgi_script_name;

    # ...
}
```

Директива `try_files` проверяет существование PHP-файла, прежде чем передать запрос настроенному в том же блоке FastCGI-серверу; здесь использование `try_files` оправдано.

Пример использования при проксировании Mongrel:

```
location / {
    try_files /system/maintenance.html
             $uri $uri/index.html $uri.html
```

```
@mongrel;
}

location @mongrel {
    proxy_pass http://mongrel;
}
```

Пример использования вместе с Drupal/FastCGI:

```
location / {
    error_page 404 = @drupal;
}

location ~ /\.php$ {
    try_files $uri @drupal;

    fastcgi_pass ...;

    fastcgi_param SCRIPT_FILENAME /path/to$fastcgi_script_name;
    fastcgi_param SCRIPT_NAME      $fastcgi_script_name;
    fastcgi_param QUERY_STRING     $args;

    # ... прочие fastcgi_param
}

location @drupal {
    fastcgi_pass ...;

    fastcgi_param SCRIPT_FILENAME /path/to/index.php;
    fastcgi_param SCRIPT_NAME     /index.php;
    fastcgi_param QUERY_STRING    q=$uri&$args;

    # ... прочие fastcgi_param
}
```

Пример использования вместе с Wordpress и Joomla:

```
location / {
    error_page 404 = @wordpress;
}

location ~ /\.php$ {
    try_files $uri @wordpress;

    fastcgi_pass ...;

    fastcgi_param SCRIPT_FILENAME /path/to$fastcgi_script_name;
    # ... прочие fastcgi_param
}

location @wordpress {
    fastcgi_pass ...;

    fastcgi_param SCRIPT_FILENAME /path/to/index.php;
    # ... прочие fastcgi_param
}
```

types

Синтаксис	<code>types { ... }</code>
По умолчанию	<code>types text/html html; image/gif gif; image/jpeg jpg;</code>
Контекст	http, server, location

Задаёт соответствие расширений имен файлов и MIME-типов ответов. Расширения нечувствительны к регистру символов. Одному MIME-типу может соответствовать несколько расширений, например:

```
types {
    application/octet-stream bin exe dll;
    application/octet-stream deb;
    application/octet-stream dmg;
}
```

Достаточно полная таблица соответствий входит в дистрибутив Angie и находится в файле `conf/mime.types`.

Для того чтобы для определённого `location`'а для всех ответов выдавался MIME-тип "application/octet-stream", можно использовать следующее:

```
location /download/ {
    types { }
    default_type application/octet-stream;
}
```

types_hash_bucket_size

Синтаксис	<code>types_hash_bucket_size</code> размер;
По умолчанию	<code>types_hash_bucket_size 64;</code>
Контекст	http, server, location

Задаёт размер корзины в хэш-таблицах типов. Подробнее настройка хэш-таблиц обсуждается отдельно.

types_hash_max_size

Синтаксис	<code>types_hash_max_size</code> размер;
По умолчанию	<code>types_hash_max_size 1024;</code>
Контекст	http, server, location

Задаёт максимальный размер хэш-таблиц типов. Подробнее настройка хэш-таблиц обсуждается отдельно.

underscores_in_headers

Синтаксис	<code>underscores_in_headers on off;</code>
По умолчанию	<code>underscores_in_headers off;</code>
Контекст	http, server

Разрешает или запрещает использование символов подчеркивания в полях заголовка запроса клиента. Если использование символов подчеркивания запрещено, поля заголовка запроса, в именах которых есть подчеркивания, помечаются как недопустимые и подпадают под действие директивы *ignore_invalid_headers*.

Если директива указана на уровне *server*, то может использоваться значение из сервера по умолчанию.

variables_hash_bucket_size

Синтаксис	<code>variables_hash_bucket_size</code> размер;
По умолчанию	<code>variables_hash_bucket_size 64</code> ;
Контекст	http

Задаёт размер корзины в хэш-таблице переменных. Подробнее настройка хэш-таблиц обсуждается отдельно.

variables_hash_max_size

Синтаксис	<code>variables_hash_max_size</code> размер;
По умолчанию	<code>variables_hash_max_size 2048</code> ;
Контекст	http

Задаёт максимальный размер хэш-таблиц переменных. Подробнее настройка хэш-таблиц обсуждается отдельно.

Встроенные переменные

Модуль **http_core** поддерживает встроенные переменные, имена которых совпадают с именами переменных веб-сервера Apache. Прежде всего, это переменные, представляющие из себя поля заголовка запроса клиента, такие как `$http_user_agent`, `$http_cookie` и тому подобное. Кроме того, есть и другие переменные:

`$angie_version`

версия Angie

`$arg_<имя>`

аргумент *имя* в строке запроса

`$args`

аргументы в строке запроса

`$binary_remote_addr`

адрес клиента в бинарном виде, длина значения всегда 4 байта для IPv4-адресов или 16 байт для IPv6-адресов

`$body_bytes_sent`

число байт, переданное клиенту, без учета заголовка ответа; переменная совместима с параметром "%B" модуля Apache `mod_log_config`

`$bytes_sent`

число байт, переданных клиенту

`$connection`

порядковый номер соединения

`$connection_requests`

текущее число запросов в соединении

`$connection_time`

время соединения в секундах с точностью до миллисекунд

`$content_length`

поле `Content-Length` заголовка запроса

`$content_type`

поле `Content-Type` заголовка запроса

`$cookie_<имя>`

cookie с указанным *именем*

`$document_root`

значение директивы *root* или *alias* для текущего запроса

`$document_uri`

то же, что и `$uri`

`$host`

в порядке приоритета: имя хоста из строки запроса, или имя хоста из поля "Host" заголовка запроса, или имя сервера, соответствующего запросу

`$hostname`

имя хоста

`$http_<имя>`

произвольное поле заголовка запроса; последняя часть имени переменной соответствует имени поля, приведенному к нижнему регистру, с заменой символов тире на символы подчеркивания

В запросах по протоколу HTTP/3 переменная `$http_host` инициализируется из значения псевдо-заголовка `:authority`, если заголовок `Host` не был передан клиентом.

`$https`

on если соединение работает в режиме SSL, либо пустая строка

`$is_args`

?, если в строке запроса есть аргументы, и пустая строка, если их нет

`$is_request_port`

:, если значение `$request_port` непустое, либо пустая строка

`$limit_rate`

установка этой переменной позволяет ограничивать скорость передачи ответа, см. *limit_rate*

`$msec`

текущее время в секундах с точностью до миллисекунд

`$nginx_version`

версия nginx

`$pid`

номер (PID) рабочего процесса

`$pipe`

p если запрос был pipelined, иначе .

`$proxy_protocol_addr`

Адрес клиента, полученный из заголовка протокола PROXY.

Протокол PROXY должен быть предварительно включен при помощи установки параметра `proxy_protocol` в директиве *listen*.

`$proxy_protocol_port`

Порт клиента, полученный из заголовка протокола PROXY.

Протокол PROXY должен быть предварительно включен при помощи установки параметра `proxy_protocol` в директиве *listen*.

`$proxy_protocol_server_addr`

Адрес сервера, полученный из заголовка протокола PROXY.

Протокол PROXY должен быть предварительно включен при помощи установки параметра `proxy_protocol` в директиве *listen*.

`$proxy_protocol_server_port`

Порт сервера, полученный из заголовка протокола PROXY.

Протокол PROXY должен быть предварительно включен при помощи установки параметра `proxy_protocol` в директиве *listen*.

`$proxy_protocol_tlv_<имя>`

TLV, полученный из заголовка протокола PROXY. *Имя* может быть именем типа TLV или его числовым значением. В последнем случае значение задается в шестнадцатеричном виде и должно начинаться с 0x:

```
$proxy_protocol_tlv_alpn
$proxy_protocol_tlv_0x01
```

SSL TLV могут также быть доступны как по имени типа TLV, так и по его числовому значению, оба должны начинаться с `ssl_`:

```
$proxy_protocol_tlv_ssl_version
$proxy_protocol_tlv_ssl_0x21
```

Поддерживаются следующие имена типов TLV:

- `alpn` (0x01) - протокол более высокого уровня, используемый поверх соединения
- `authority` (0x02) - значение имени хоста, передаваемое клиентом
- `unique_id` (0x05) - уникальный идентификатор соединения
- `netns` (0x30) - имя пространства имен
- `ssl` (0x20) - структура SSL TLV в бинарном виде

Поддерживаются следующие имена типов SSL TLV:

- `ssl_version` (0x21) - версия SSL, используемая в клиентском соединении
- `ssl_cn` (0x22) - Common Name сертификата
- `ssl_cipher` (0x23) - имя используемого шифра
- `ssl_sig_alg` (0x24) - алгоритм, используемый для подписи сертификата
- `ssl_key_alg` (0x25) - алгоритм публичного ключа

Также поддерживается следующее специальное имя типа SSL TLV:

- `ssl_verify` - результат проверки клиентского сертификата: 0, если клиент предоставил сертификат и он был успешно верифицирован, либо ненулевое значение

Протокол PROXY должен быть предварительно включен при помощи установки параметра `proxy_protocol` в директиве *listen*.

`$query_string`

то же, что и `$args`

`$realpath_root`

абсолютный путь, соответствующий значению директивы *root* или *alias* для текущего запроса, в котором все символические ссылки преобразованы в реальные пути

`$remote_addr`

адрес клиента

`$remote_port`

порт клиента

`$remote_user`

имя пользователя, использованное в Basic аутентификации

`$request`

первоначальная строка запроса целиком

`$request_body`

Тело запроса. Значение переменной *появляется* в *location*'ах, обрабатываемых директивами *proxy_pass*, *fastcgi_pass*, *uwsgi_pass* и *scgi_pass*, когда тело было прочитано в буфер в памяти.

`$request_body_file`

Имя временного файла, в котором хранится тело запроса. По завершении обработки файл необходимо удалить. Для того чтобы тело запроса всегда записывалось в файл, следует включить *client_body_in_file_only*. При передаче имени временного файла в проксированном запросе или в запросе к FastCGI/uwsgi/SCGI-серверу следует запретить передачу самого тела директивами *proxy_pass_request_body off*, *fastcgi_pass_request_body off*, *uwsgi_pass_request_body off* или *scgi_pass_request_body off* соответственно.

`$request_completion`

ОК, если запрос завершился, либо пустая строка

`$request_filename`

путь к файлу для текущего запроса, формируемый из директив *root* или *alias* и URI запроса

`$request_id`

уникальный идентификатор запроса, сформированный из 16 случайных байт, в шестнадцатеричном виде

`$request_length`

длина запроса в байтах (включая строку запроса, заголовок и тело запроса)

`$request_method`

метод запроса, обычно GET или POST

`$request_port`

в порядке приоритета: номер порта из компонента authority URI, или номер порта из поля заголовка запроса Host

`$request_time`

время обработки запроса в секундах с точностью до миллисекунд; время, прошедшее с момента чтения первых байт от клиента

`$request_uri`

первоначальный URI запроса целиком (с аргументами), не изменяется в процессе обработки запроса; см. `$uri` — текущий URI с учётом всех преобразований

`$scheme`

схема запроса, "http" или "https"

`$sent_body`

тело ответа подзапроса или внешнего запроса, если оно сохранено в памяти; иначе пустая строка

`$sent_http_<имя>`

произвольное поле заголовка ответа; последняя часть имени переменной соответствует имени поля, приведенному к нижнему регистру, с заменой символов тире на символы подчеркивания

`$sent_trailer_<имя>`

произвольное поле, отправленное в конце ответа; последняя часть имени переменной соответствует имени поля, приведенному к нижнему регистру, с заменой символов тире на символы подчеркивания

`$server_addr`

Адрес сервера, принявшего запрос.

Получение значения этой переменной обычно требует одного системного вызова. Чтобы избежать системного вызова, в директивах *listen* следует указывать адреса и использовать параметр `bind`.

`$server_name`

имя сервера, принявшего запрос

`$server_port`

порт сервера, принявшего запрос

`$server_protocol`

протокол запроса, обычно "HTTP/1.0", "HTTP/1.1" или "HTTP/2.0"

`$status`

статус ответа

`$tcpinfo_rtt`, `$tcpinfo_rttvar`, `$tcpinfo_snd_cwnd`, `$tcpinfo_rcv_space`

информация о клиентском TCP-соединении; доступна на системах, поддерживающих параметр сокета `TCP_INFO`; в Linux `$tcpinfo_rtt` и `$tcpinfo_rttvar` измеряются в микросекундах, `$tcpinfo_snd_cwnd` — в сегментах, а `$tcpinfo_rcv_space` — в байтах

`$time_iso8601`

локальное время в формате по стандарту ISO 8601

`$time_local`

локальное время в Common Log Format

\$uri

Текущий URI запроса в нормализованном виде.

Значение **\$uri** может изменяться в процессе обработки запроса, например, при перезаписи с помощью *rewrite*, при внутренних перенаправлениях или при использовании индексных файлов. О том, как это значение сопоставляется с путем в файловой системе, см. *Выбор локаций*.

Access

Модуль управляет доступом к ресурсам сервера на основе IP-адресов клиентов или сетей. Он позволяет разрешать или блокировать доступ для определенных IP-адресов, IP-диапазонов или UNIX-сокетов, чтобы повысить безопасность, ограничивая доступ к важным разделам веб-сайта или приложения.

Доступ также можно ограничить с помощью пароля, используя модуль Auth Basic, или на основе результата подзапроса, используя модуль Auth Request. Чтобы одновременно применять ограничения по адресу и паролю, используйте директиву *satisfy*.

Пример конфигурации

```
location / {
    deny 192.168.1.1;
    allow 192.168.1.0/24;
    allow 10.1.1.0/16;
    allow 2001:0db8::/32;
    deny all;
}
```

Правила обрабатываются последовательно до первого совпадения. В этом примере доступ разрешен только для IPv4-сетей 10.1.1.0/16 и 192.168.1.0/24, за исключением отдельного адреса 192.168.1.1, и для IPv6-сети 2001:0db8::/32. Когда правил много, предпочтительно использовать переменные из модуля Geo.

Директивы

allow

Синтаксис	<code>allow адрес CIDR unix: all;</code>
По умолчанию	—
Контекст	http, server, location, limit_except

Разрешает доступ для указанной сети или адреса. Специальное значение **all** означает все IP-адреса клиентов.

Специальное значение **unix:** разрешает доступ для любых UNIX-сокетов.

deny

Синтаксис	<code>deny адрес CIDR unix: all;</code>
По умолчанию	—
Контекст	http, server, location, limit_except

Запрещает доступ для указанной сети или адреса. Специальное значение `all` означает все IP-адреса клиентов.

Специальное значение `unix`: запрещает доступ для любых UNIX-сокетов.

ACME

Обеспечивает автоматическое получение сертификатов с использованием протокола ACME.

Пример конфигурации

В этом примере ACME-клиент с именем `example` автоматически получает и обновляет сертификат для `example.com` и `www.example.com`, используя HTTP-проверку по умолчанию:

```
http {
    acme_client example https://acme-v02.api.letsencrypt.org/directory;

    server {
        listen 80; # Необязательно, если нет сервера,
                  # слушающего порт HTTP-проверки
                  # (см. директиву 'acme_http_port')

        listen 443 ssl;

        server_name example.com www.example.com;
        acme example;

        ssl_certificate $acme_cert_example;
        ssl_certificate_key $acme_cert_key_example;
    }
}
```

Другие методы проверки (DNS, ALPN, с помощью хуков) и подробные инструкции по настройке см. в разделе *Настройка ACME*.

Директивы

acme

Синтаксис	<code>acme имя;</code>
По умолчанию	—
Контекст	<code>server</code>

Указывает клиент ACME, который получает сертификат для допустимых идентификаторов сертификата этого блока `server`. Единый сертификат охватывает все допустимые доменные имена и IP-адреса, указанные в директивах `server_name` всех блоков `server`, которые ссылаются на клиент с именем `имя`; если изменится конфигурация `server_name`, сертификат обновляется для учета изменений.

При каждом запуске Angie или перезагрузке его конфигурации для всех идентификаторов, у которых отсутствует действующий сертификат, запрашиваются новые сертификаты. Возможные причины включают истечение срока действия сертификатов, отсутствие файлов или невозможность прочитать их, изменение в настройках сертификатов, а также предыдущий запрос, завершившийся ошибкой.

Примечание

Эта директива определяет только то, какие допустимые идентификаторы сертификата включаются в запросы; она не влияет на то, где можно использовать сертификат. Любой блок **server** может ссылаться на сертификат через переменную `$acme_cert_<имя>`, независимо от того, содержит ли блок директиву **acme**. Удаление **acme** из блока **server** просто исключает значения `server_name` этого блока из последующих запросов сертификатов, но не запрещает блоку использовать сертификат.

Примечание

Сейчас домены, заданные через регулярные выражения, не поддерживаются и будут пропускаться.

Домены со звездочкой поддерживаются только в режиме **challenge=dns** в **acme_client**.

Адреса IPv4 и IPv6 поддерживаются, если клиент не использует **challenge=dns**. При выборе DNS-проверки IP-адреса пропускаются.

Эта директива может быть указана несколько раз для загрузки сертификатов разных типов, например RSA и ECDSA:

```
server {
    listen 443 ssl;
    server_name example.com www.example.com;

    ssl_certificate $acme_cert_rsa;
    ssl_certificate_key $acme_cert_key_rsa;

    ssl_certificate $acme_cert_ecdsa;
    ssl_certificate_key $acme_cert_key_ecdsa;

    acme rsa;
    acme ecdsa;
}
```

acme_client

Синтаксис	<code>acme_client имя uri [enabled=on off] [key_type=тип] [key_bits=число] [email=email] [max_cert_size=размер] [max_key_auth_size=размер] [renew_before_expiry=время] [renew_on_load] [retry_after_error=off время] [challenge=dns http alpn] [profile=имя] [account_key=файл] [eab=id[:alg]:key];</code>
По умолчанию	—
Контекст	http

Определяет клиент АСМЕ с глобально уникальным *именем*. Оно должно быть допустимым для каталога, представляет собой строку с переменными и будет использоваться без учета регистра.

Каждый клиент управляет одним сертификатом; чтобы получить отдельные сертификаты, настройте несколько блоков **acme_client** (см. *Отдельные сертификаты для разных доменов*).

Совет

Задаваемое здесь имя клиента идентифицирует его в конфигурации Angie, позволяя сопоставить между собой директивы `acme_client`, `acme` и переменные модуля, использующие это имя; не следует путать его с именем вашего домена или сервера.

Вторым обязательным параметром является `uri` каталога ACME. Например, URI каталога Let's Encrypt ACME указан как <https://acme-v02.api.letsencrypt.org/directory>.

Примечание

Модуль ACME добавляет в контекст `client` именованный `location @acme`, который можно использовать для настройки запросов к каталогу ACME; По умолчанию в этом `location` задана директива `proxy_pass` с `uri` каталога, к которой можно добавить другие настройки из модуля Proxy.

Клиент определяет адрес каталога по его имени через `resolver`, который по умолчанию обращается к DNS-серверам из `/etc/resolv.conf`.

Примечание

Для тестирования удостоверяющие центры обычно предоставляют отдельные тестовые среды. Например, среда тестирования Let's Encrypt — <https://acme-staging-v02.api.letsencrypt.org/directory>.

enabled	Включает или отключает обновление сертификатов для клиента; это полезно, например, для временной приостановки без удаления клиента из конфигурации. По умолчанию: on .
key_type	Тип алгоритма закрытого ключа для сертификата. Допустимые значения: rsa , ecdsa . По умолчанию: ecdsa .
key_bits	Количество битов в ключе сертификата. По умолчанию: 256 для ecdsa , 2048 для rsa .
email	Необязательный адрес электронной почты для обратной связи; используется при создании учетной записи на сервере СА.
max_cert_size	Задаёт максимальный допустимый размер файла нового сертификата, чтобы зарезервировать место для нового сертификата в разделяемой памяти; чем больше число доменов, для которых запрашивается сертификат, тем больше требуется места. Размер задается в байтах; также допустимы суффиксы k и m . Этот параметр не ограничивает размер ответа АСМЕ-сервера; для этого используется acme_max_response_size . Если параметр не задан, Angie вычисляет приблизительный размер на основе списка доменов и использует его при выделении разделяемой памяти. Если в момент запуска сертификат уже существует, но его размер превышает значение max_cert_size , значение max_cert_size динамически увеличивается до размера существующего файла сертификата. Если размер сертификата, полученного при обновлении, превышает max_cert_size , процесс обновления завершится с ошибкой. По умолчанию: вычисляется автоматически.
max_key_auth_size	Ограничивает размер строки авторизации ключа, которую Angie хранит в разделяемой памяти для АСМЕ-проверки. Если АСМЕ-сервер возвращает строку авторизации ключа большего размера, запрос завершается ошибкой с рекомендацией увеличить max_key_auth_size . Хотя параметр задается в строке acme_client , это единая настройка, общая для всех клиентов в блоке http . По умолчанию: 2k .
renew_before_exp	Время до истечения срока действия сертификата, когда должно начаться его обновление. По умолчанию: 30d .
renew_on_load	Указывает, что сертификат следует принудительно обновлять при каждой загрузке конфигурации.
retry_after_error	Время до повторной попытки, если получить сертификат не удалось. Если задано значение off , клиент не будет снова пытаться получить сертификат после ошибки. По умолчанию: 2h . Эта задержка действует, пока Angie работает; перезагрузка конфигурации сразу же повторно запрашивает сертификат.
challenge	Задаёт тип верификации для АСМЕ-клиента. Допустимые значения: dns , http , alpn . Значение alpn включает проверку TLS-ALPN-01 и требует, чтобы Angie был собран с OpenSSL с поддержкой ALPN (не поддерживается сборками с BoringSSL и AWS-LC). По умолчанию: http .
profile	Запрашивает у удостоверяющего центра определенный АСМЕ-профиль. Профиль — это заданный удостоверяющим центром вариант настроек сертификата и проверки, например срок действия сертификата или типы идентификаторов, которые можно запрашивать. Так, доступные профили Let's Encrypt описаны в документации по профилям . АСМЕ-сервер должен объявлять этот профиль в метаданных своего каталога; иначе клиент не сможет получить сертификат.
account_key	Указывает полный путь к файлу, содержащему ключ в формате PEM. Это удобно, если вы хотите использовать существующий ключ аккаунта вместо

4.10. Справочная информация

Поддерживаемые типы ключей:

- RSA-ключи с длиной, кратной 8, в диапазоне от 2048 до 8192 бит.

acme_client_path

Синтаксис	<code>acme_client_path <i>путь</i>;</code>
По умолчанию	—
Контекст	http

Переопределяет *путь* к каталогу для хранения сертификатов и ключей, заданному при сборке с помощью параметра сборки `--http-acme-client-path`.

acme_dns_port

Синтаксис	<code>acme_dns_port <i>порт</i> <i>ip[:порт]</i> [<i>ip6</i>][:<i>порт</i>];</code>
Значение по умолчанию	<code>acme_dns_port 53;</code>
Контекст	http

Указывает порт, который модуль использует для обработки DNS-запросов от ACME-сервера по UDP. Номер порта должен быть в диапазоне от 1 до 65535.

Также поддерживается указание IP-адреса вместе с опциональным портом. Могут быть использованы как IPv4-адреса в виде `ip:порт`, так и IPv6-адреса в виде `[ip6]:порт`:

```
acme_dns_port 8053;
acme_dns_port 127.0.0.1;
acme_dns_port [::1];
```

Чтобы использовать номер порта 1024 или ниже, мастер-процесс Angie должен работать с привилегиями суперпользователя.

acme_dns_ttl

Синтаксис	<code>acme_dns_ttl <i>число</i>;</code>
По умолчанию	<code>acme_dns_ttl 1;</code>
Контекст	http

Задаёт время жизни (TTL, в секундах) TXT-записей, которые модуль возвращает в ответах на запросы DNS-проверки от ACME-серверов. Допустимые значения — от 0 до 2147483647, согласно RFC 2181.

Увеличьте значение, если проверка не завершается: некоторые DNS-провайдеры отфильтровывают ответы с таким низким TTL, как значение по умолчанию.

acme_hook

Синтаксис	<code>acme_hook <i>имя</i> [<i>uri</i>];</code>
По умолчанию	—
Контекст	location

Включает проверку домена с помощью хуков для ACME-клиента, заданного параметром *имя*. Когда для выпуска или обновления сертификата требуется проверка домена, Angie формирует

внутренний запрос к именованному `location`, в котором размещена эта директива. Способ обработки запроса полностью зависит от других директив, заданных в том же `location`, таких как `fastcgi_pass`, `proxy_pass` или любого другого обработчика запросов.

<i>имя</i>	Имя АСМЕ-клиента, для которого этот хук обрабатывает проверку домена.
<i>uri</i>	Строка с переменными; задает URI запроса для вызовов хука. По умолчанию: <code>/</code> .

Например, следующая конфигурация передает значения переменных хука в приложение FastCGI через URI запроса:

```
acme_hook example uri=/acme_hook/$acme_hook_name?domain=$acme_hook_domain&key=$acme_
hook_keyauth;
fastcgi_param REQUEST_URI $request_uri;
fastcgi_pass ...;
```

acme_http_port

Синтаксис	<code>acme_http_port <i>порт</i> <i>ip[:порт]</i> [<i>ip6</i>][:<i>порт</i>];</code>
Значение по умолчанию	<code>acme_http_port 80;</code>
Контекст	<code>http</code>

Указывает порт, который модуль использует для обработки HTTP-проверок АСМЕ. Номер порта должен быть в диапазоне от 1 до 65535.

Также поддерживается указание IP-адреса вместе с опциональным портом. Могут быть использованы как IPv4-адреса в виде `ip:порт`, так и IPv6-адреса в виде `[ip6]:порт`:

```
acme_http_port 8080;
acme_http_port 127.0.0.1;
acme_http_port [::1];
```

Если ни один сервер не слушает указанный адрес и порт, модуль создаст отдельный слушающий сокет для HTTP-проверок.

Чтобы использовать номер порта 1024 или ниже, мастер-процесс Angie должен работать с привилегиями суперпользователя.

acme_max_response_size

Синтаксис	<code>acme_max_response_size <i>размер</i>;</code>
По умолчанию	<code>acme_max_response_size 32k;</code>
Контекст	<code>http</code>

Ограничивает максимальный размер тела ответа АСМЕ-сервера. Если ответ превышает это значение, запрос завершится ошибкой. Увеличьте значение, если появляются ошибки вида `too big subrequest response while sending to client`.

Встроенные переменные

`$acme_cert_<имя>`

Содержимое последнего файла сертификата (если он есть), полученного клиентом с этим *именем*.

`$acme_cert_key_<имя>`

Содержимое файла ключа сертификата, используемого клиентом с этим *именем*.

Примечание

Файл сертификата доступен, только если клиент АСМЕ получил хотя бы один сертификат, а вот файл ключа доступен сразу после запуска.

`$acme_hook_challenge`

Тип проверки. Возможные значения: `dns`, `http`, `alpn`.

`$acme_hook_client`

Имя АСМЕ-клиента, инициирующего запрос.

`$acme_hook_domain`

Проверяемый домен. Если это wildcard-домен, он будет передан без префикса `*..`

`$acme_hook_keyauth`

Строка авторизации:

- При DNS-проверке используется как значение TXT-записи, имя которой формируется как `_acme-challenge. + $acme_hook_domain + ..`
- При HTTP-проверке эта строка должна использоваться в качестве содержимого ответа, запрашиваемого АСМЕ-сервером.

`$acme_hook_name`

Имя хука. Для разных типов проверки оно может иметь разные значения и смысл:

Значение	Смысл при DNS-проверке	Смысл при HTTP-проверке
<code>add</code> (добавление хука)	Необходимо добавить соответствующую TXT-запись в конфигурацию DNS.	Необходимо подготовить ответ на соответствующий HTTP-запрос.
<code>remove</code> (удаление хука)	Можно удалить TXT-запись из конфигурации DNS.	Данный HTTP-запрос более не актуален; можно удалить ранее созданный файл со строкой авторизации.

`$acme_hook_token`

Токен для проверки. При HTTP-проверке используется как имя запрашиваемого файла: `/.well-known/acme-challenge/ + $acme_hook_token`.

Addition

Фильтр, добавляющий текст до и после ответа.

Пример конфигурации

```
location / {
    add_before_body /before_action;
    add_after_body /after_action;
}
```

Директивы

add_after_body

Синтаксис	<code>add_after_body uri;</code>
По умолчанию	—
Контекст	http, server, location

Добавляет после тела ответа текст, выдаваемый в результате работы заданного подзапроса. Пустая строка ("") в качестве параметра отменяет добавление, унаследованное с предыдущего уровня конфигурации.

add_before_body

Синтаксис	<code>add_before_body uri;</code>
По умолчанию	—
Контекст	http, server, location

Добавляет перед телом ответа текст, выдаваемый в результате работы заданного подзапроса. Пустая строка ("") в качестве параметра отменяет добавление, унаследованное с предыдущего уровня конфигурации.

addition_types

Синтаксис	<code>addition_types mime-mun ...;</code>
По умолчанию	<code>addition_types text/html;</code>
Контекст	http, server, location

Разрешает добавлять текст в ответах с указанными MIME-типами в дополнение к `text/html`. Специальное значение `*` соответствует любому MIME-типу.

API

Модуль API реализует HTTP RESTful интерфейс для получения базовой информации о веб-сервере в формате JSON, а также статистики по клиентским соединениям, зонам разделяемой памяти, DNS-запросам, HTTP-запросам, кэшу HTTP-ответов, сессиям модуля Stream и зонам HTTP-модуля Limit Conn, потокового модуля Limit Conn, модуля Limit Req и HTTP-модуля Upstream, а также информацию о сертификатах, заданных в конфигурации и полученных ACME-клиентами.

Примечание

Живой пример вывода API доступен по адресу <https://console.angie.software/api/>.

Интерфейс принимает HTTP-методы GET и HEAD; запрос с другим методом вызовет ошибку:

```
{
  "error": "MethodNotAllowed",
  "description": "The POST method is not allowed for the requested API element \"/\
  ↪ ."
}
```

В этом интерфейсе есть раздел *динамической конфигурации*, позволяющий менять настройки без перезагрузки конфигурации или перезапуска; сейчас доступна конфигурация отдельных серверов в составе *upstream*.

Директивы

api

Синтаксис	<code>api <i>путь</i>;</code>
По умолчанию	—
Контекст	location

Включает HTTP RESTful интерфейс в location.

Параметр *путь* является обязательным. Подобно директиве *alias*, задает путь для замены указанного в location, но по дереву API, а не файловой системы.

Если указан в префиксном location:

```
location /stats/ {
    api /status/http/server_zones/;
}
```

часть URI запроса, совпадающая с префиксом */stats/*, будет заменена на путь, указанный в параметре *путь*: */status/http/server_zones/*. К примеру, по запросу */stats/foo/* будет доступен элемент API */status/http/server_zones/foo/*.

Допускается использование переменных: *api /status/\$module/server_zones/\$name/* и использование внутри regex location:

```
location ~~/api/([~/>+)/(.*)$ {
    api /status/http/$1_zones/$2;
}
```

Здесь параметр *путь* определяет полный путь к элементу API; так, из запроса к */api/location/data/* будут выделены переменные:

```
$1 = "location"
$2 = "data/"
```

И конечный запрос будет иметь вид */status/http/location_zones/data/*.

Примечание

Можно разделить *API динамической конфигурации* и неизменяемый *API статуса*, отражающий текущее состояние:

```
location /config/ {
    api /config/;
}
```

```
location /status/ {
    api /status/;
}
```

Также параметр *путь* позволяет управлять доступом к API:

```
location /status/ {
    api /status/;

    allow 127.0.0.1;
    deny all;
}
```

Или же:

```
location /blog/requests/ {
    api /status/http/server_zones/blog/requests/;

    auth_basic "blog";
    auth_basic_user_file conf/htpasswd;
}
```

i Примечание

Если `api` стоит в `location` с косой чертой в конце префикса (например, `location /name/`), и при этом в директиве `auto_redirect` указано `default`, запросы без косой черты в конце будут перенаправляться (`/name -> /name/`).

api_config_files

Синтаксис	<code>api_config_files on off;</code>
По умолчанию	<code>off</code>
Контекст	<code>location</code>

Включает или отключает добавление объекта `config_files`, перечисляющего содержимое всех файлов конфигурации Angie, загруженных сейчас экземпляром сервера, в состав раздела API `/status/angie/`. Например, при такой конфигурации:

```
location /status/ {
    api /status/;
    api_config_files on;
}
```

Запрос к `/status/angie/` возвращает приблизительно следующее:

```
{
  "version": "1.1.1",
  "address": "192.168.16.5",
  "generation": 1,
  "load_time": "2026-09-17T12:58:39.789Z",
  "config_files": {
    "/etc/angie/angie.conf": "...",
```

```
    "/etc/angie/mime.types": "..."  
  }  
}
```

По умолчанию вывод отключен, так как файлы конфигурации могут содержать особо чувствительные, конфиденциальные сведения.

Метрики

Angie публикует статистику использования в разделе API `/status/`; открыть доступ к ней можно, задав соответствующий `location`. Полный доступ:

```
location /status/ {  
    api /status/;  
}
```

Пример частичного доступа, уже приводившийся выше:

```
location /stats/ {  
    api /status/http/server_zones/;  
}
```

Пример конфигурации

С конфигурацией, включающей `location /status/`, зоны `resolver`, `http` в `upstream`, `http server`, `location`, `cache`, `limit_conn` в `http` и `limit_req`:

```
http {  
  
    resolver 127.0.0.53 status_zone=resolver_zone;  
    proxy_cache_path /var/cache/angie/cache keys_zone=cache_zone:2m;  
    limit_conn_zone $binary_remote_addr zone=limit_conn_zone:10m;  
    limit_req_zone $binary_remote_addr zone=limit_req_zone:10m rate=1r/s;  
  
    upstream upstream {  
        zone upstream 256k;  
        server backend.example.com service=_example._tcp resolve max_conns=5;  
        keepalive 4;  
    }  
  
    server {  
        server_name www.example.com;  
        listen 443 ssl;  
  
        status_zone http_server_zone;  
        proxy_cache cache_zone;  
        proxy_cache_valid 200 10m;  
  
        access_log /var/log/access.log main;  
  
        location / {  
            root /usr/share/angie/html;  
            status_zone location_zone;  
            limit_conn limit_conn_zone 1;  
            limit_req zone=limit_req_zone burst=5;  
        }  
        location /status/ {  
            api /status/;  
        }  
    }  
}
```

```

        allow 127.0.0.1;
        deny all;
    }

}

```

В ответ на запрос `curl https://www.example.com/status/` Angie возвращает:

дерево JSON

```

{
  "angie": {
    "version": "1.1.1",
    "address": "192.168.16.5",
    "generation": 1,
    "load_time": "2026-09-17T12:58:39.789Z"
  },
  "connections": {
    "accepted": 2257,
    "dropped": 0,
    "active": 3,
    "idle": 1
  },
  "slabs": {
    "cache_zone": {
      "pages": {
        "used": 2,
        "free": 506
      },
      "slots": {
        "64": {
          "used": 1,
          "free": 63,
          "reqs": 1,
          "fails": 0
        },
        "512": {
          "used": 1,
          "free": 7,
          "reqs": 1,
          "fails": 0
        }
      }
    },
    "limit_conn_zone": {
      "pages": {
        "used": 2,
        "free": 2542
      },
    },
  },
}

```

```

        "slots": {
            "64": {
                "used":1,
                "free":63,
                "reqs":74,
                "fails":0
            },

            "128": {
                "used":1,
                "free":31,
                "reqs":1,
                "fails":0
            }
        }
    },

    "limit_req_zone": {
        "pages": {
            "used":2,
            "free":2542
        },

        "slots": {
            "64": {
                "used":1,
                "free":63,
                "reqs":1,
                "fails":0
            },

            "128": {
                "used":2,
                "free":30,
                "reqs":3,
                "fails":0
            }
        }
    }
},

"http": {
    "server_zones": {
        "http_server_zone": {
            "ssl": {
                "handshaked":4174,
                "reuses":0,
                "timedout":0,
                "failed":0
            },

            "requests": {
                "total":4327,
                "processing":0,
                "discarded":8
            },
        },
    },

```

```

        "responses": {
            "200":4305,
            "302":12,
            "404":4
        },

        "data": {
            "received":733955,
            "sent":59207757
        }
    },

    "location_zones": {
        "location_zone": {
            "requests": {
                "total":4158,
                "discarded":0
            },

            "responses": {
                "200":4157,
                "304":1
            },

            "data": {
                "received":538200,
                "sent":177606236
            }
        }
    },

    "caches": {
        "cache_zone": {
            "size":0,
            "cold":false,
            "hit": {
                "responses":0,
                "bytes":0
            },

            "stale": {
                "responses":0,
                "bytes":0
            },

            "updating": {
                "responses":0,
                "bytes":0
            },

            "revalidated": {
                "responses":0,
                "bytes":0
            },

            "miss": {
                "responses":0,

```



```

        "bytes":0,
        "responses_written":0,
        "bytes_written":0
    },

    "expired": {
        "responses":0,
        "bytes":0,
        "responses_written":0,
        "bytes_written":0
    },

    "bypass": {
        "responses":0,
        "bytes":0,
        "responses_written":0,
        "bytes_written":0
    }
}

},

"limit_conns": {
    "limit_conn_zone": {
        "passed":73,
        "skipped":0,
        "rejected":0,
        "exhausted":0
    }
},

"limit_reqs": {
    "limit_req_zone": {
        "passed":54816,
        "skipped":0,
        "delayed":65,
        "rejected":26,
        "exhausted":0
    }
},

"upstreams": {
    "upstream": {
        "peers": {
            "192.168.16.4:80": {
                "server":"backend.example.com",
                "service":"_example._tcp",
                "backup":false,
                "weight":5,
                "state":"up",
                "selected": {
                    "current":2,
                    "total":232
                },
            },

            "max_conns":5,
            "responses": {
                "200":222,
            },
        },
    },
}
```

```

        "302":12
    },
    "data": {
        "sent":543866,
        "received":27349934
    },
    "health": {
        "fails":0,
        "unavailable":0,
        "downtime":0,
        "header_time":21,
        "response_time":42
    },
    "sid":"<server_id>"
    }
},
    "keepalive":2
}
},
    "resolvers": {
        "resolver_zone": {
            "queries": {
                "name":442,
                "srv":2,
                "addr":0
            },
            "responses": {
                "success":440,
                "timedout":1,
                "format_error":0,
                "server_failure":1,
                "not_found":1,
                "unimplemented":0,
                "refused":1,
                "other":0
            }
        }
    }
}
}

```

Набор метрик можно запросить по отдельной ветви JSON, построив соответствующий запрос. Например:

```

$ curl https://www.example.com/status/angie
$ curl https://www.example.com/status/connections
$ curl https://www.example.com/status/slabs
$ curl https://www.example.com/status/slabs/<зона>/slots
$ curl https://www.example.com/status/slabs/<зона>/slots/64
$ curl https://www.example.com/status/http/
$ curl https://www.example.com/status/http/acme_clients
$ curl https://www.example.com/status/http/acme_clients/<клиент>

```

```
$ curl https://www.example.com/status/http/metric_zones
$ curl https://www.example.com/status/http/metric_zones/<зона>/metrics
$ curl https://www.example.com/status/http/server_zones
$ curl https://www.example.com/status/http/server_zones/<http_server_zone>
$ curl https://www.example.com/status/http/server_zones/<http_server_zone>/ssl
```

Аргументы запроса

Ответы API в формате JSON можно изменять с помощью следующих аргументов строки запроса:

pretty

По умолчанию API форматирует ответы JSON с отступами и переносами строк. Чтобы получить компактный однострочный вывод, добавьте **pretty=off** к строке запроса:

```
$ curl https://www.example.com/status/connections?pretty=off

{"accepted":2257,"dropped":0,"active":3,"idle":1}
```

date

По умолчанию модуль использует для дат строки в формате ISO 8601; чтобы вместо этого использовать целочисленный формат эпохи UNIX, добавьте **date=epoch** к строке запроса:

```
$ curl https://www.example.com/status/angie/load_time

"2024-03-31T23:59:59Z"

$ curl https://www.example.com/status/angie/load_time?date=epoch

1711929599
```

Время в строковом формате всегда указывается в UTC, а миллисекунды в нем присутствуют, только если они ненулевые. В формате эпохи время выражается целым числом секунд, без миллисекунд.

defaults

Получить параметры по умолчанию можно с аргументом **defaults=on**:

```
$ curl http://127.0.0.1/config/http/upstreams/backend/servers?defaults=on
```

```
{
  "backend.example.com": {
    "weight": 1,
    "max_conns": 5,
    "max_fails": 1,
    "fail_timeout": 10,
    "slow_start": 0,
    "backup": false,
    "down": false,
    "sid": ""
  }
}
```

Состояние сервера

/status/angie

```
{
  "version": "1.1.1",
```

```
{
  "build_time": "2026-09-17T16:05:43.805Z",
  "address": "192.168.16.5",
  "generation": 1,
  "load_time": "2026-09-17T16:15:43.805Z"
  "config_files": {
    "/etc/angie/angie.conf": "...",
    "/etc/angie/mime.types": "..."
  }
}
```

version	Строка; версия запущенного сервера Angie
build	Строка; сборка, если указана при компиляции
build_time	Строка; время сборки исполняемого файла Angie в формате даты
address	Строка; адрес сервера, принявшего запрос к API
generation	Число; версия (поколение) конфигурации, отсчитываемая с последнего запуска Angie
load_time	Строка; время последней перезагрузки конфигурации в формате даты
config_files	Объект; его члены — абсолютные имена всех файлов конфигурации Angie, загруженных сейчас экземпляром сервера, а их значения — строковые представления содержимого файлов, например: <pre>{ "/etc/angie/angie.conf": "server {\n listen 80;\n # ... \n\n}\n\n}" }</pre> ⚠ Предупреждение Объект <code>config_files</code> есть в <code>/status/angie/</code>, только если включена директива <code>api_config_files</code>.

`/status/angie/license`

```
{
  "path": "/etc/angie/license.pem",
  "status": "valid",
  "owner": "Example Corp",
  "days_left": 30,
  "since": "2026-01-01",
  "until": "2027-01-01",
  "limits": {
    "worker_processes": 16,
    "worker_connections": 65535
  }
}
```

path	Строка; полный путь к файлу лицензии
status	Строка; статус лицензии: <code>missing</code> , <code>invalid</code> , <code>valid</code> , <code>grace</code> , <code>expired</code> или <code>pending</code>
owner	Строка; владелец лицензии из subject сертификата
days_left	Число; дни до смены состояния лицензии. Отрицательное значение означает, что лицензия истекла, и значение показывает число дней после истечения
since	Строка; дата начала действия лицензии
until	Строка; дата окончания действия лицензии
limits	Объект; лимиты лицензии для текущего экземпляра

Соединения

`/status/connections`

```
{
  "accepted": 2257,
  "dropped": 0,
  "active": 3,
  "idle": 1
}
```

accepted	Число; суммарное количество принятых клиентских соединений
dropped	Число; суммарное количество сброшенных клиентских соединений
active	Число; текущее количество активных клиентских соединений
idle	Число; текущее количество бездействующих клиентских соединений

Зоны разделяемой памяти с распределением slab

`/status/slabs/<зона>`

Статистика для зон разделяемой памяти с **распределением slab**. Учитывается любая зона, использующая распределение slab, в том числе зоны `limit_conn`, `limit_req`, HTTP cache, а также зона разделяемой памяти HTTP или stream апстрима:

```
limit_conn_zone $binary_remote_addr zone=limit_conn_zone:10m;
limit_req_zone $binary_remote_addr zone=limit_req_zone:10m rate=1r/s;
proxy_cache cache_zone;
proxy_cache_valid 200 10m;
```

В указанной таким образом зоне разделяемой памяти будет собираться следующая статистика:

pages	Объект; статистика по страницам памяти. Здесь страница — это системная страница памяти, обычно 4К
used	Число; текущее количество используемых страниц памяти
free	Число; текущее количество свободных страниц памяти
slots	Объект; статистика по слотам памяти, по каждому из размеров. slots содержит данные по размеру слота памяти (8, 16, 32, и т.д., вплоть до половины размера страницы памяти в байтах)
used	Число; текущее количество используемых слотов памяти заданного размера
free	Число; текущее количество свободных слотов памяти заданного размера
reqs	Число; суммарное количество попыток выделения памяти указанного размера
fails	Число; количество неудавшихся попыток выделения памяти указанного размера

Пример:

```
{
  "pages": {
    "used": 2,
    "free": 506
  },
  "slots": {
    "64": {
      "used": 1,
      "free": 63,
      "reqs": 1,
      "fails": 0
    }
  }
}
```

DNS-запросы к резолверу

/status/resolvers/<зона>

Для сбора статистики в директиве *resolver* нужно задать параметр **status_zone** (HTTP или Stream):

```
resolver 127.0.0.53 status_zone=resolver_zone;
```

В указанной таким образом зоне разделяемой памяти будет собираться следующая статистика:

queries	Объект; статистика запросов
name	Число; количество запросов на преобразование имен в адреса (A- и AAAA-запросы)
srv	Число; количество запросов на преобразование сервисов в адреса (SRV запросы)
addr	Число; количество запросов на преобразование адресов в имена (PTR-запросы)
responses	Объект; статистика ответов
success	Число; количество успешных ответов
timedout	Число; количество запросов, не дождавшихся ответа
format_error	Число; количество ответов с кодом 1 (Format Error)
server_failure	Число; количество ответов с кодом 2 (Server Failure)
not_found	Число; количество ответов с кодом 3 (Name Error)
unimplemented	Число; количество ответов с кодом 4 (Not Implemented)
refused	Число; количество ответов с кодом 5 (Refused)
other	Число; количество запросов, завершенных с другим ненулевым кодом
sent	Объект; статистика отправленных DNS-запросов
a	Число; количество запросов типа A
aaaa	Число; количество запросов типа AAAA
ptr	Число; количество запросов типа PTR
srv	Число; количество запросов типа SRV

Примечание

queries и **responses** учитывают каждый запрос на разрешение имён, который Angie выполняет внутренне, включая ответы из TTL-кэша. **sent** учитывает пакеты, фактически отправленные на сервер имён; разница между ними отражает обращения к кэшу.

Коды ответов описаны в [RFC 1035](#), часть 4.1.1.

Различные типы DNS-записей описаны в [RFC 1035](#), [RFC 2782](#) и [RFC 3596](#).

Пример:

```
{
  "queries": {
    "name": 442,
    "srv": 2,
    "addr": 0
  },

  "responses": {
    "success": 440,
    "timedout": 1,
    "format_error": 0,
    "server_failure": 1,
    "not_found": 1,
    "unimplemented": 0,
    "refused": 1,
  },

  "sent": {
    "a": 185,
    "aaaa": 245,
    "srv": 2,
    "ptr": 12
  }
}
```

```
}  
}
```

HTTP server и location

/status/http/server_zones/<зона>

Для сбора статистики в контексте *server* нужно задать директиву *status_zone*:

```
server {  
    ...  
    status_zone server_zone;  
}
```

Для группировки метрик по пользовательскому значению используйте альтернативный синтаксис. В этом примере метрики агрегируются по *\$host*, и каждая группа выводится как отдельная зона:

```
status_zone $host zone=server_zone:5;
```

Число после имени зоны ограничивает количество групп, а не объем памяти (подробнее см. *status_zone*); метрики значений сверх этого ограничения выводятся под именем самой зоны.

В указанной таким образом зоне разделяемой памяти будет собираться следующая статистика:

ssl	Объект; SSL-метрики. Присутствует, если в server есть listen ssl ;
handshaked	Число; суммарное количество успешных SSL-рукопожатий
reuses	Число; суммарное количество повторных использований SSL-сессий во время SSL-рукопожатий
timedout	Число; суммарное количество SSL-рукопожатий с истекшим таймаутом
failed	Число; суммарное количество неуспешных SSL-рукопожатий
requests	Объект; метрики запросов
total	Число; суммарное количество клиентских запросов
processing	Число; текущее количество обслуживаемых клиентских запросов
discarded	Число; суммарное количество запросов завершенных без отправки ответа
responses	Объект; метрики ответов
<code>	Число; ненулевое количество ответов со статусом <code> (100-599)
xxx	Число; ненулевое количество ответов с другим кодом статуса
data	Объект; метрики данных
received	Число; суммарное количество байт, полученное от клиентов
sent	Число; суммарное количество байт, отправленное клиентам

Примечание

Счетчики **responses** учитывают только реальные ответы, фактически отправленные клиенту. Запрос, обработка которого завершилась без отправки ответа, считается в **discarded** и не увеличивает счетчики **responses**. В частности, 499 — это не ответ, а служебный внутренний код, который записывается только в журнал доступа и сигнализирует о том, что обработка запроса прекратилась без отправки ответа; такие запросы попадают в **discarded**, но никогда в **responses**.

Пример:

```
{  
    "ssl":{  
        "handshaked":4174,
```



```

        "reuses":0,
        "timeout":0,
        "failed":0
    },

    "requests":{
        "total":4327,
        "processing":0,
        "discarded":0
    },

    "responses":{
        "200":4305,
        "302":6,
        "304":12,
        "404":4
    },

    "data":{
        "received":733955,
        "sent":59207757
    }
}

```

/status/http/location_zones/<зона>

Для сбора статистики в контексте *location* или *if* в *location* нужно задать директиву *status_zone*:

```

location / {
    root /usr/share/angie/html;
    status_zone location_zone;

    if ($request_uri ~* "~/condition") {
        # ...
        status_zone if_location_zone;
    }
}

```

Для группировки метрик по пользовательскому значению используйте альтернативный синтаксис. В этом примере метрики агрегируются по *\$host*, и каждая группа выводится как отдельная зона:

```
status_zone $host zone=server_zone:5;
```

В указанной таким образом зоне разделяемой памяти будет собираться следующая статистика:

requests	Объект; метрики запросов
total	Число; суммарное количество клиентских запросов
discarded	Число; суммарное количество запросов завершенных без отправки ответа
responses	Объект; метрики ответов
<code>	Число; ненулевое количество ответов со статусом <code> (100-599)
xxx	Число; ненулевое количество ответов с другим кодом статуса
data	Объект; метрики данных
received	Число; суммарное количество байт, полученное от клиентов
sent	Число; суммарное количество байт, отправленное клиентам

i Примечание

Счетчики **responses** учитывают только реальные ответы, фактически отправленные клиенту. Запрос, обработка которого завершилась без отправки ответа, считается в **discarded** и не увеличивает счетчики **responses**. В частности, 499 — это не ответ, а служебный внутренний код, который записывается только в журнал доступа и сигнализирует о том, что обработка запроса прекратилась без отправки ответа; такие запросы попадают в **discarded**, но никогда в **responses**.

Пример:

```
{
  "requests": {
    "total": 4158,
    "discarded": 0
  },
  "responses": {
    "200": 4157,
    "304": 1
  },
  "data": {
    "received": 538200,
    "sent": 177606236
  }
}
```

/status/http/metric_zones/<зона>

Пользовательские метрики, созданные директивами *metric_zone* или *metric_complex_zone* в контексте **http**. Метрики обновляются через директиву *metric* или переменные модуля.

discarded	Число; количество отброшенных записей метрик из-за нехватки памяти в зоне.
metrics	Объект; метрики по ключам. Для зон с одной метрикой значения — числа. Для сложных зон значения — объекты с именами метрик. Для режима histogram значения — объекты с именами бакетов.

Если задан **discard_key** и часть записей была истекшей, агрегированные метрики доступны под этим ключом.

Пример:

```
{
  "discarded": 3,
  "metrics": {
    "example.com": {
      "count": 42,
      "max": 8
    },
    "expired": {
      "count": 10,
      "max": 3.2
    }
  }
}
```

Stream server

/status/stream/server_zones/<зона>

Для сбора статистики в контексте *server* нужно задать директиву *status_zone*:

```
server {
    ...
    status_zone server_zone;
}
```

Для группировки метрик по пользовательскому значению используйте альтернативный синтаксис. В этом примере метрики агрегируются по *\$server_addr*, и каждая группа выводится как отдельная зона:

```
status_zone $server_addr zone=server_zone:5;
```

В указанной таким образом зоне разделяемой памяти будет собираться следующая статистика:

ssl	Объект; SSL-метрики. Присутствует, если в server есть listen ssl ;
handshaked	Число; суммарное количество успешных SSL-рукопожатий
reuses	Число; суммарное количество повторных использований SSL-сессий во время SSL-рукопожатий
timedout	Число; суммарное количество SSL-рукопожатий с истекшим таймаутом
failed	Число; суммарное количество неуспешных SSL-рукопожатий
connections	Объект; метрики соединений
total	Число; суммарное количество клиентских соединений
processing	Число; текущее количество обслуживаемых клиентских соединений
discarded	Число; суммарное количество клиентских соединений, завершенных без создания сессии
passed	Число; суммарное количество клиентских соединений, переданных на другой прослушивающий порт директивами pass
sessions	Объект; метрики сессий
success	Число; количество сессий, завершенных с кодом 200, что означает успешное завершение
invalid	Число; количество сессий, завершенных с кодом 400, случается, когда сервер не может прочитать данные от клиента, например, заголовок PROXY protocol
forbidden	Число; количество сессий, завершенных с кодом 403, когда доступ запрещен, например, ограничен для определенного адреса клиента
internal_error	Число; количество сессий, завершенных с кодом 500, внутренняя ошибка сервера
bad_gateway	Число; количество сессий, завершенных с кодом 502, Bad Gateway, если, например, сервер в upstream недоступен или не может быть выбран
service_unavailable	Число; количество сессий, завершенных с кодом 503, Service Unavailable, если, например, доступ ограничен числом входящих соединений
data	Объект; метрики данных
received	Число; суммарное количество байт, полученное от клиентов
sent	Число; суммарное количество байт, отправленное клиентам

Пример:

```
{
  "ssl": {
    "handshaked": 24,
    "reuses": 0,
```

```
{
  "timedout": 0,
  "failed": 0
},
"connections": {
  "total": 24,
  "processing": 1,
  "discarded": 0,
  "passed": 2
},
"sessions": {
  "success": 24,
  "invalid": 0,
  "forbidden": 0,
  "internal_error": 0,
  "bad_gateway": 0,
  "service_unavailable": 0
},
"data": {
  "received": 2762947,
  "sent": 53495723
}
}
```

/status/stream/metric_zones/<зона>

Пользовательские метрики, созданные директивами `metric_zone` или `metric_complex_zone` в контексте `stream`. Метрики обновляются через директиву `metric` или переменные модуля.

discarded	Число; количество отброшенных записей метрик из-за нехватки памяти в зоне.
metrics	Объект; метрики по ключам. Для зон с одной метрикой значения — числа. Для сложных зон значения — объекты с именами метрик. Для режима <code>histogram</code> значения — объекты с именами бакетов.

Если задан `discard_key` и часть записей была истекшей, агрегированные метрики доступны под этим ключом.

Пример:

```
{
  "discarded": 3,
  "metrics": {
    "127.0.0.1": {
      "count": 42,
      "max": 8
    },
    "expired": {
      "count": 10,
      "max": 3.2
    }
  }
}
```

HTTP caches

```
proxy_cache cache_zone;
proxy_cache_valid 200 10m;
```

/status/http/caches/<cache>

Для каждой зоны, сконфигурированной в *proxy_cache*, хранятся следующие данные:

```
{
  "name_zone": {
    "size": 0,
    "cold": false,
    "hit": {
      "responses": 0,
      "bytes": 0
    },
    "stale": {
      "responses": 0,
      "bytes": 0
    },
    "updating": {
      "responses": 0,
      "bytes": 0
    },
    "revalidated": {
      "responses": 0,
      "bytes": 0
    },
    "miss": {
      "responses": 0,
      "bytes": 0,
      "responses_written": 0,
      "bytes_written": 0
    },
    "expired": {
      "responses": 0,
      "bytes": 0,
      "responses_written": 0,
      "bytes_written": 0
    },
    "bypass": {
      "responses": 0,
      "bytes": 0,
      "responses_written": 0,
      "bytes_written": 0
    }
  }
}
```

size	Число; объем, который кэш сейчас занимает на диске, в байтах. Каждая запись кэша учитывается как целое число блоков файловой системы, занятых ее файлом, поэтому значение всегда превышает суммарный размер тел кэшированных ответов
max_size	Число; заданное в конфигурации ограничение на размер кэша, в байтах; отсутствует, если ограничение не задано
cold	Логическое значение; true , пока <i>загрузчик кэша</i> подгружает данные с диска
hit	Объект; метрики возвращенных из кэша ответов (<i>proxy_cache_valid</i>)
responses	Число; суммарное количество ответов, прочитанных из кэша
bytes	Число; суммарное количество байт, прочитанных из кэша
stale	Объект; метрики просроченных ответов, возвращенных из кэша (<i>proxy_cache_use_stale</i>)
responses	Число; суммарное количество ответов, прочитанных из кэша
bytes	Число; суммарное количество байт, прочитанных из кэша
updating	Объект; метрики просроченных ответов, возвращенных из кэша, пока данные в кэше обновляются (<i>proxy_cache_use_stale updating</i>)
responses	Число; суммарное количество ответов, прочитанных из кэша
bytes	Число; суммарное количество байт, прочитанных из кэша
revalidated	Объект; метрики просроченных и ревалидированных ответов, возвращенных из кэша (<i>proxy_cache_revalidate</i>)
responses	Число; суммарное количество ответов, прочитанных из кэша
bytes	Число; суммарное количество байт, прочитанных из кэша
miss	Объект; метрики ответов, не найденных в кэше
responses	Число; суммарное количество соответствующих ответов
bytes	Число; суммарное количество байт, прочитанных с проксируемого сервера
responses_written	Число; суммарное количество ответов, записанных в кэш
bytes_written	Число; суммарное количество байт, записанных в кэш
expired	Объект; количество ответов, возвращенных не из кэша, т.к. просрочены
responses	Число; суммарное количество соответствующих ответов
bytes	Число; суммарное количество байт, прочитанных с проксируемого сервера
responses_written	Число; суммарное количество ответов, записанных в кэш
bytes_written	Число; суммарное количество байт, записанных в кэш
bypass	Объект; статистика ответов, возвращенных в обход кэша (<i>proxy_cache_bypass</i>)
responses	Число; суммарное количество соответствующих ответов
bytes	Число; суммарное количество байт, прочитанных с проксируемого сервера
responses_written	Число; суммарное количество ответов, записанных в кэш
bytes_written	Число; суммарное количество байт, записанных в кэш

При включении *шардинга кэша* с помощью директив *proxy_cache_path* отдельные шарды указываются как объекты-члены в объекте **shards**; при этом **size**, **max_size** и **cold** выводятся для каждого шарда в отдельности, а на уровне зоны отсутствуют:

shards	Объект; его члены — отдельные шарды
<shard>	Объект; представляет отдельный шард, а имя объекта — путь кэша
size	Число; объем, который шард сейчас занимает на диске, в байтах; каждая запись кэша учитывается как целое число блоков файловой системы, занятых ее файлом
max_size	Число; заданное в конфигурации ограничение на размер шарда, в байтах; отсутствует, если ограничение не задано
cold	Логическое значение; true , пока <i>загрузчик кэша</i> подгружает данные с диска

```
{
  "name_zone": {
    "shards": {
      "/path/to/shard1": {
        "size": 0,
        "cold": false
      },
      "/path/to/shard2": {
        "size": 0,
        "cold": false
      }
    }
  }
}
```

ACME-клиенты

`/status/http/acme_clients/<клиент>`

Для каждого настроенного *acme_client* в блоке **http** возвращается текущее состояние клиента и сертификата:

```
{
  "state": "ready",
  "certificate": "valid",
  "details": "The client is ready to request a certificate.",
  "next_run": "2026-09-17T16:15:43.805Z"
}
```

state	Строка; состояние ACME-клиента. Возможные значения: ready , requesting , disabled , failed .
certificate	Строка; состояние сертификата. Возможные значения: valid , expired , missing , mismatch , error .
details	Строка; краткие сведения о последнем действии ACME.
next_run	Дата; ближайшая запланированная попытка запроса или обновления сертификата. Не возвращается, когда state равно disabled или requesting .

limit_conn

```
limit_conn_zone $binary_remote_addr zone=limit_conn_zone:10m;
```

`/status/http/limit_conns/<зона>`, `/status/stream/limit_conns/<зона>`

Каждая из сконфигурированных зон: **limit_conn** в **http** или **limit_conn** в **stream** содержит следующие данные:

```
{
  "passed": 73,
  "skipped": 0,
  "rejected": 0,
  "exhausted": 0
}
```

passed	Число; суммарное количество переданных на проксируемый сервер соединений
skipped	Число; суммарное количество соединений, переданных с нулевым или превосходящим 255 байт <key>
rejected	Число; суммарное количество соединений сверх сконфигурированного ограничения
exhausted	Число; суммарное количество соединений, сброшенных из-за переполнения хранилища зоны

limit_req

```
limit_req_zone $binary_remote_addr zone=limit_req_zone:10m rate=1r/s;
```

/status/http/limit_reqs/<зона>

Каждая из сконфигурированных зон *limit_req* содержит следующие данные:

```
{
  "passed": 54816,
  "skipped": 0,
  "delayed": 65,
  "rejected": 26,
  "exhausted": 0
}
```

passed	Число; суммарное количество проксированных соединений
skipped	Число; суммарное количество соединений, переданных с нулевым или превосходящим 255 байт <key>
delayed	Число; суммарное количество задержанных соединений
rejected	Число; суммарное количество сброшенных соединений
exhausted	Число; суммарное количество соединений, сброшенных из-за переполнения хранилища зоны

HTTP upstream

Чтобы включить сбор следующих метрик, задайте директиву *zone* в контексте *upstream*, например:

```
upstream upstream {
  zone upstream 256k;
  server backend.example.com service=_example._tcp resolve max_conns=5;
  keepalive 4;
}
```

Статистика использования памяти этой зоны доступна в разделе API /status/slabs/ под её именем.

/status/http/upstreams/<upstream>

где <upstream> — имя апстрима, в конфигурации которого указана директива *zone*.

```
{
  "peers": {
    "192.168.16.4:80": {
      "server": "backend.example.com",
      "service": "_example._tcp",
      "backup": false,

```



```

    "weight": 5,
    "state": "up",
    "selected": {
      "current": 2,
      "total": 232
    },

    "max_conns": 5,
    "responses": {
      "200": 222,
      "302": 12
    },

    "data": {
      "sent": 543866,
      "received": 27349934
    },

    "health": {
      "fails": 0,
      "unavailable": 0,
      "downtime": 0,
      "header_time": 21,
      "response_time": 42
    },

    "sid": "<server_id>"
  },
  "keepalive": 2
}

```

peers	Объект; содержит метрики всех пиров апстрима во вложенных объектах, имена которых — канонические представления адресов этих пиров. Внутри каждого вложенного объекта:
server	Строка; сервер, как он указан в директиве <code>server</code>
service	Строка; имя сервиса, указанное в директиве <code>server</code> , если сконфигурировано
backup	Логическое значение; <code>true</code> для backup-серверов.
weight	Число; сконфигурированный <code>weight</code>
state	Строка; текущее состояние пира, и какие запросы ему отправляются: • busy: указывает, что число запросов на сервер достигло ограничения, заданного <code>max_conns</code>, и новые запросы на него не отправляются; • down: отключен вручную, не отправляются никакие запросы; • draining: аналогичен down, но отправляются запросы сессий, привязанных ранее через <code>sticky</code>; • recovering: восстанавливается после сбоя согласно <code>slow_start</code>, отправляются все больше запросов; • unavailable: достиг предела <code>max_fails</code>, отправляются пробные клиентские запросы с интервалом <code>fail_timeout</code>; • up: работоспособен, запросы отправляются как обычно; • checking: настроен как essential и проверяется, отправляются только проверочные запросы; • unhealthy: неработающий, отправляются только проверочные запросы.
selected	Объект; статистика выбора пиров
current	Число; текущее количество соединений к пиру
total	Число; общее количество запросов переданных пиру
last	Строка или число; время последнего выбора пира в формате даты
max_conns	Число; максимальное количество одновременных активных соединений к пиру, если сконфигурировано
responses	Объект; статистика ответов
<code>	Число; ненулевое количество ответов со статусом <code><code></code> (100-599)
xxx	Число; ненулевое количество ответов с другим кодом статуса
data	Объект; метрики данных
received	Число; суммарное количество байт, полученное от пира
sent	Число; суммарное количество байт, отправленное пиру
health	Объект; статистика по состоянию пира
fails	Число; общее количество неудачных попыток работы с пиром
unavailable	Число; столько раз пир становился unavailable по достижении значения <code>max_fails</code>
downtime	Число; суммарное время (в миллисекундах), в течение которого пир был недоступен для выбора как unavailable
downstart	Строка или число; время, когда пир стал unavailable , в формате даты. Поле присутствует, только пока пир находится в состоянии unavailable ; в остальных случаях оно отсутствует
header_time	Число; среднее время (в миллисекундах) получения заголовков ответа от сервера; см. директиву response_time_factor
response_time	Число; среднее время (в миллисекундах) получения ответа от сервера; см. директиву response_time_factor
sid	Строка; id сервера, указанный в конфигурации апстрима
feedback	Число; текущее среднее значение обратной связи, используемое методом балансировки <code>feedback</code> , присутствует, только если используется этот метод
keepalive	Число; текущее количество кэшированных соединений
backup_switch	Объект; содержит текущее состояние логики активного резервирования, присутствует, если для апстрима настроен backup_switch
active	Число; уровень активной группы, которая сейчас используется для балансировки запросов. Если активная группа является основной, значе-

4.10. Справочная информация

149

timeout	Число; оставшееся время ожидания в миллисекундах, после которого балансировщик перепроверит на наличие здоровых узлов группы с меньшим уровнем, начиная с основной, а группы с большим уровнем не
----------------	---

Если для апстрима используется метод балансировки *least_bandwidth*, у пира дополнительно доступно поле:

bandwidth	Число; текущее среднее значение использования пропускной способности в байтах в миллисекунду (экспоненциальное скользящее среднее), используемое методом балансировки <i>least_bandwidth</i> ; значение -1 означает, что сервер еще не выбирался
------------------	--

health/probes

Если для апстрима настроены проверки *upstream_probe*, то в объекте **health** также есть вложенный объект **probes**, содержащий счетчики проверок работоспособности сервера, а **state**, помимо значений из таблицы выше, может принимать значения **checking** и **unhealthy**:

```
{
  "192.168.16.4:80": {
    "state": "unhealthy",
    "...": "...",
    "health": {
      "...": "...",
      "probes": {
        "count": 10,
        "fails": 10,
        "last": "2026-09-17T09:56:07Z"
      }
    }
  }
}
```

Значение **checking** у **state** не учитывается в **downtime** и означает, что сервер, проверка которого настроена с параметром **essential**, еще не проверялся; значение **unhealthy** — что сервер неработающий. Оба эти состояния также означают, что сервер не участвует в балансировке. Детали проверок см. в описании *upstream_probe*.

Счетчики в **probes**:

count	Число; общее количество проверок этого сервера
fails	Число; количество неуспешных проверок
last	Строка или число; время последней проверки в формате даты

queue

Если для апстрима настроена очередь запросов, то в объекте апстрима также есть вложенный объект **queue**, содержащий счетчики запросов в очереди:

```
{
  "queue": {
    "queued": 20112,
    "waiting": 1011,
    "dropped": 6031,
    "timedout": 560,
    "overflows": 13
  }
}
```

Значения счетчиков суммируются по всем рабочим процессам:

queued	Число; общее количество запросов, попавших в очередь
waiting	Число; текущее количество запросов в очереди
dropped	Число; общее количество запросов, удаленных из очереди из-за того, что клиент преждевременно закрыл соединение
timedout	Число; общее количество запросов, удаленных из очереди по таймауту
overflows	Число; общее количество случаев переполнения очереди

Stream upstream

Чтобы включить сбор следующих метрик, задайте директиву `zone` в контексте *upstream*, например:

```
upstream upstream {
    zone upstream 256k;
    server backend.example.com service=_example._tcp resolve max_conns=5;
    keepalive 4;
}
```

Статистика использования памяти этой зоны доступна в разделе API `/status/slabs/` под её именем.

`/status/stream/upstreams/<upstream>`

Здесь *<upstream>* — имя апстрима, в конфигурации которого использована директива `zone`.

```
{
  "peers": {
    "192.168.16.4:1935": {
      "server": "backend.example.com",
      "service": "_example._tcp",
      "backup": false,
      "weight": 5,
      "state": "up",
      "selected": {
        "current": 2,
        "total": 232
      },
      "max_conns": 5,
      "data": {
        "sent": 543866,
        "received": 27349934
      },
      "health": {
        "fails": 0,
        "unavailable": 0,
        "downtime": 0,
        "connect_time": 3,
        "first_byte_time": 21,
        "last_byte_time": 42
      }
    }
  }
}
```

peers	Объект; содержит метрики всех пиров апстрима во вложенных объектах, имена которых — канонические представления адресов этих пиров. Внутри каждого вложенного объекта:
server	Строка; адрес, заданный директивой <code>server</code>
service	Строка; имя сервиса, если оно указано в директиве <code>server</code>
backup	Логическое значение; <code>true</code> для backup-серверов
weight	Число; заданный для пира вес
state	Строка; текущее состояние пира, и какие запросы ему отправляются: • busy: указывает, что число запросов на сервер достигло ограничения, заданного <code>max_conns</code>, и новые запросы на него не отправляются; • down: отключен вручную, не отправляются никакие запросы; • draining: аналогичен down, но отправляются запросы сессий, привязанных ранее через <code>sticky</code>; • recovering: восстанавливается после сбоя согласно <code>slow_start</code>, отправляются все больше запросов; • unavailable: достиг предела <code>max_fails</code>, отправляются пробные клиентские запросы с интервалом <code>fail_timeout</code>; • up: работоспособен, запросы отправляются как обычно. • checking: настроен как essential и проверяется, отправляются только проверочные запросы; • unhealthy: неработающий, отправляются только проверочные запросы.
selected	Объект; статистика выбора этого пира для подключения
current	Число; текущее количество подключений к пиру
total	Число; общее количество подключений, направленных этому пиру
last	Строка или число; время, когда пир был выбран в последний раз, в формате даты
max_conns	Число; максимальное количество одновременных активных подключений к пиру, если оно задано
data	Объект; статистика передачи данных
received	Число; общее количество байт, полученное от пира
sent	Число; общее количество байт, отправленное пиру
health	Объект; статистика по состоянию пира
fails	Число; общее количество неудачных попыток связаться с пиром
unavailable	Число; общее количество переходов в состояние unavailable по достижении значения <code>max_fails</code>
downtime	Число; общее время в миллисекундах, в течение которого пир находился в состоянии unavailable (недоступен для выбора)
downstart	Строка или число; время, когда пир последний раз стал unavailable , в формате даты. Поле присутствует, только пока пир находится в состоянии unavailable ; в остальных случаях оно отсутствует
connect_time	Число; среднее время (в миллисекундах) установления соединения с сервером; см. директиву <code>response_time_factor</code>
first_byte_time	Число; среднее время (в миллисекундах) получения первого байта ответа от сервера; см. директиву <code>response_time_factor</code>
last_byte_time	Число; среднее время (в миллисекундах) получения полного ответа от сервера; см. директиву <code>response_time_factor</code>
feedback	Число; текущее среднее значение обратной связи, используемое методом балансировки <code>feedback</code> , присутствует, только если используется этот метод
backup_switch	Объект; содержит текущее состояние логики активного резервирования, присутствует, если для апстрима настроен <code>backup_switch</code>
active	Число; уровень активной группы, которая сейчас используется для балансировки запросов. Если активная группа является основной, значение равно 0
timeout	Число; оставшееся время ожидания в миллисекундах, после которого балансировщик перепроверит на наличие здоровых узлов группы с

Если для апстрима используется метод балансировки *least_bandwidth* или *least_packets*, у пира дополнительно доступны поля:

bandwidth	Число; текущее среднее значение использования пропускной способности в байтах в миллисекунду (экспоненциальное скользящее среднее), используемое методом балансировки <i>least_bandwidth</i> ; значение -1 означает, что сервер еще не выбирался
packets	Число; текущее среднее количество переданных пакетов в миллисекунду (экспоненциальное скользящее среднее), используемое методом балансировки <i>least_packets</i> ; значение -1 означает, что сервер еще не выбирался

Если для апстрима настроены проверки *upstream_probe*, то в объекте **health** также есть вложенный объект **probes**, содержащий счетчики проверок работоспособности сервера, а **state**, помимо значений из таблицы выше, может принимать значения **checking** и **unhealthy**:

```
{
  "192.168.16.4:80": {
    "state": "unhealthy",
    "...": "...",
    "health": {
      "...": "...",
      "probes": {
        "count": 2,
        "fails": 2,
        "last": "2026-09-17T11:03:54Z"
      }
    }
  }
}
```

Значение **checking** у **state** означает, что сервер, проверка которого настроена с параметром **essential**, еще не проверялся; значение **unhealthy** — что сервер неработающий. Оба эти состояния также означают, что сервер не участвует в балансировке. Детали проверок см. в описании *upstream_probe*.

Счетчики в **probes**:

count	Число; общее количество проверок этого сервера
fails	Число; количество неуспешных проверок
last	Строка или число; время последней проверки в формате даты

Сертификаты

/certificates/

Объект API верхнего уровня, отдельный от дерева статистики */status/*, отображает TLS-сертификаты, загруженные Angie.

Сертификат не отображается в объекте **static**, если имя файла сертификата или ключа содержит переменную. В частности, это относится к сертификату, предоставленному АСМЕ-клиентом через переменную *\$acme_cert_<имя>*. Сертификат, используемый несколькими серверами, указывается один раз.

В следующем примере Angie загружает статический сертификат и получает еще один с помощью АСМЕ-клиента. Для разрешения доменного имени сервера каталога АСМЕ-клиенту требуется настроенная директива *resolver*:

```
http {
    resolver 127.0.0.53;

    acme_client example https://acme.example.com/directory;

    server {
        listen      443 ssl;
        server_name www.example.com;

        ssl_certificate      example.com.crt;
        ssl_certificate_key  example.com.key;

        location /certificates/ {
            api /certificates/;
        }
    }

    server {
        listen      443 ssl;
        server_name acme.example.com;

        acme          example;
        ssl_certificate $acme_cert_example;
        ssl_certificate_key $acme_cert_key_example;
    }
}
```

В ответ на запрос `curl https://www.example.com/certificates/` Angie возвращает:

```
{
  "static": {
    "example.com.crt": {
      "key": "RSA (2048 bits)",
      "chain": [
        {
          "subject": {
            "common_name": "example.com",
            "alt_names": [
              "example.com",
              "www.example.com"
            ],
            "organization": "Example, Inc."
          },
          "issuer": {
            "common_name": "Example Root CA",
            "country": "US",
            "organization": "Example, Inc."
          },
          "validity": {
            "since": "Sep 18 19:46:19 2022 GMT",
            "until": "Jun 15 19:46:19 2025 GMT"
          }
        }
      ]
    }
  },
}
```

```

"acme_clients": {
  "example": {
    "key": "EC (prime256v1)",
    "chain": [
      {
        "subject": {
          "common_name": "acme.example.com"
        },

        "issuer": {
          "common_name": "Example ACME CA"
        },

        "validity": {
          "since": "Sep 18 19:46:19 2022 GMT",
          "until": "Dec 17 19:46:19 2022 GMT"
        }
      }
    ]
  }
}

```

static	Объект; серверные сертификаты, заданные директивой <i>ssl_certificate</i> в HTTP-модуле, <i>ssl_certificate</i> в потоковом модуле или <i>ssl_certificate</i> в почтовом модуле; ключом каждого элемента служит имя файла сертификата. В сборках с <i>--with-ntls</i> объект также содержит сертификаты, заданные директивой <i>proxy_ssl_certificate</i> в HTTP или <i>proxy_ssl_certificate</i> в потоковом модуле для аутентификации на проксируемых серверах
acme_clients	Объект; доступен в сборках Angie с HTTP-модулем АСМЕ, который также необходим для потокового модуля АСМЕ. Содержит сертификаты, полученные АСМЕ-клиентами для HTTP- и потоковых серверов; ключом каждого элемента служит имя <i>acme_client</i> . Отображаются только клиенты, уже получившие сертификат

Каждая запись в объектах **static** и **acme_clients** описывает один комплект сертификатов:

key	Строка; тип и размер (или кривая) закрытого ключа, например RSA (2048 bits), EC (prime256v1) или DH (2048 bits)
chain	Массив; сертификаты комплекта, начиная с конечного (серверного) сертификата, затем промежуточные сертификаты, по направлению к корневому. Каждый элемент — объект, описывающий один сертификат:
subject	Объект; отличительное имя (DN) субъекта
common_name	Строка; Common Name (CN)
alt_names	Массив строк; DNS-имена и IP-адреса из расширения Subject Alternative Name
country	Строка; Country (C)
state_or_province	Строка; State or Province Name (ST)
organization	Строка; Organization (O)
issuer	Объект; отличительное имя (DN) издателя, с теми же полями, что и subject
validity	Объект; срок действия сертификата
since	Строка; начало срока действия (<i>notBefore</i>)
until	Строка; окончание срока действия (<i>notAfter</i>)

Поля вложенных объектов **subject** и **issuer** возвращаются, только если соответствующие значения есть в сертификате.

API динамической конфигурации

В составе API есть раздел **/config**, позволяющий динамически менять конфигурацию Angie в формате JSON с помощью HTTP-запросов **PUT**, **PATCH** и **DELETE**. Все изменения атомарны: новые настройки применяются целиком либо не применяются вовсе. При ошибке Angie сообщит, в чем причина.

Подразделы /config

Сейчас в разделе **/config** доступна настройка отдельных серверов в составе апстримов для модулей HTTP и stream; число настроек, к которым применима динамическая конфигурация, плавно увеличивается.

/config/http/upstreams/<upstream>/servers/<имя>

Позволяет настраивать отдельные серверы в составе апстрима, в том числе добавлять новые и удалять настроенные.

Параметры в составе пути URI:

<upstream>	Имя блока upstream ; чтобы настраивать его через /config , в нем должна быть задана директива zone , определяющая зону разделяемой памяти.
<имя>	Имя конкретного сервера в составе указанного <upstream> ; задается в формате <service>@<host> , где: <service>@ — необязательная часть, задающая имя сервиса в целях разрешения SRV-записей. <host> — доменное имя сервиса (при наличии <i>resolve</i>) или IP-адрес; также можно указать порт.

Например, для следующей конфигурации:

```
upstream backend {
    server backend.example.com service=_http._tcp resolve;
    server 127.0.0.1;
    zone backend 1m;
}
```

Допустимы такие имена серверов:

```
$ curl http://127.0.0.1/config/http/upstreams/backend/servers/_http._tcp@backend.
→example.com/
$ curl http://127.0.0.1/config/http/upstreams/backend/servers/127.0.0.1:80/
```

Этот подраздел API позволяет задавать параметры **weight**, **max_conns**, **max_fails**, **fail_timeout**, **slow_start**, **backup**, **down** и **sid**, описанные в разделе **server**.

Примечание

Отдельного параметра **drain** здесь нет; включить режим **drain** можно, задав для **down** строковое значение **drain**:

```
$ curl -X PUT -d \"drain\" \
http://127.0.0.1/config/http/upstreams/backend/servers/backend.example.com/down
```

Пример:

```
$ curl http://127.0.0.1/config/http/upstreams/backend/servers/backend.example.com?
→defaults=on
```

```
{
  "weight": 1,
  "max_conns": 0,
  "max_fails": 1,
  "fail_timeout": 10,
  "slow_start": 0,
  "backup": true,
  "down": false,
  "sid": ""
}
```

Фактически доступны будут только те параметры, которые поддерживает текущий метод балансировки нагрузки апстрима. Так, если апстрим настроен с методом балансировки `random`:

```
upstream backend {
  zone backend 256k;
  server backend.example.com resolve max_conns=5;
  random;
}
```

То добавить в него новый сервер с параметром `backup` невозможно:

```
$ curl -X PUT -d '{ "backup": true }' \
  http://127.0.0.1/config/http/upstreams/backend/servers/backend1.example.com
```

```
{
  "error": "FormatError",
  "description": "The \"backup\" field is unknown."
}
```

`/config/stream/upstreams/<upstream>/servers/<имя>`

Позволяет настраивать отдельные серверы в составе апстрима, в том числе добавлять новые и удалять настроенные.

Параметры в составе пути URI:

<code><upstream></code>	Имя блока <code>upstream</code> ; чтобы настраивать его через <code>/config</code> , в нем должна быть задана директива <code>zone</code> , определяющая зону разделяемой памяти.
<code><имя></code>	Имя конкретного сервера в составе указанного <code><upstream></code> ; задается в формате <code><service>@<host></code> , где: • <code><service>@</code> — необязательная часть, задающая имя сервиса в целях разрешения SRV-записей. • <code><host></code> — доменное имя сервиса (при наличии <code>resolve</code>) или IP-адрес; также можно указать порт.

Например, для следующей конфигурации:

```
upstream backend {
  server backend.example.com:8080 service=_example._tcp resolve;
```

```
server 127.0.0.1:12345;
zone backend 1m;
}
```

Допустимы такие имена серверов:

```
$ curl http://127.0.0.1/config/stream/upstreams/backend/servers/_example._tcp@backend.
↪example.com:8080/
$ curl http://127.0.0.1/config/stream/upstreams/backend/servers/127.0.0.1:12345/
```

Этот подраздел API позволяет задавать параметры `weight`, `max_conns`, `max_fails`, `fail_timeout`, `slow_start`, `backup` и `down`, описанные в разделе `server`.

Примечание

Отдельного параметра `drain` здесь нет; включить режим `drain` можно, задав для `down` строковое значение `drain`:

```
$ curl -X PUT -d "\"drain\" \" \
http://127.0.0.1/config/stream/upstreams/backend/servers/backend.example.com/down
```

Пример:

```
$ curl http://127.0.0.1/config/stream/upstreams/backend/servers/backend.example.com?
↪defaults=on
```

```
{
  "weight": 1,
  "max_conns": 0,
  "max_fails": 1,
  "fail_timeout": 10,
  "slow_start": 0,
  "backup": true,
  "down": false,
}
```

Фактически доступны будут только те параметры, которые поддерживает текущий метод балансировки нагрузки апстрима. Так, если апстрим настроен с методом балансировки `random`:

```
upstream backend {
  zone backend 256k;
  server backend.example.com resolve max_conns=5;
  random;
}
```

То добавить в него новый сервер с параметром `backup` невозможно:

```
$ curl -X PUT -d '{"backup": true}' \
http://127.0.0.1/config/stream/upstreams/backend/servers/backend1.example.com
```

```
{
  "error": "FormatError",
  "description": "The \"backup\" field is unknown."
}
```

При удалении серверов можно установить аргумент `connection_drop=<значение>`, чтобы переопределить настройки `proxy_connection_drop`:

```
$ curl -X DELETE \
  http://127.0.0.1/config/stream/upstreams/backend/servers/backend1.example.com?
↪connection_drop=off

$ curl -X DELETE \
  http://127.0.0.1/config/stream/upstreams/backend/servers/backend2.example.com?
↪connection_drop=on

$ curl -X DELETE \
  http://127.0.0.1/config/stream/upstreams/backend/servers/backend3.example.com?
↪connection_drop=1000
```

HTTP-методы

Рассмотрим семантику каждого из применимых к этому разделу HTTP-методов на примере следующей конфигурации апстрима:

```
http {
    # ...

    upstream backend {
        zone upstream 256k;
        server backend.example.com resolve max_conns=5;
        # ...
    }

    server {
        # ...

        location /config/ {
            api /config/;

            allow 127.0.0.1;
            deny all;
        }
    }
}
```

GET

HTTP-метод **GET** позволяет запросить сущность по любому существующему пути в пределах `/config` так же, как это делается в других разделах API.

Например, для ветки серверов апстрима `/config/http/upstreams/backend/servers/` допустимы такие запросы:

```
$ curl http://127.0.0.1/config/http/upstreams/backend/servers/backend.example.com/max_
↪conns
$ curl http://127.0.0.1/config/http/upstreams/backend/servers/backend.example.com
$ curl http://127.0.0.1/config/http/upstreams/backend/servers
$ # ...
$ curl http://127.0.0.1/config
```

Получить параметры по умолчанию можно с аргументом `defaults=on`; см. [Аргументы запроса](#).

PUT

HTTP-метод PUT позволяет создать новую JSON-сущность по указанному пути или *полностью* заменить существующую.

Например, чтобы добавить не заданный ранее параметр `max_fails` у сервера `backend.example.com` в апстриме `backend`:

```
$ curl -X PUT -d '2' \
  http://127.0.0.1/config/http/upstreams/backend/servers/backend.example.com/max_
↪fails
```

```
{
  "success": "Updated",
  "description": "Existing configuration API entity \"/config/http/upstreams/
↪backend/servers/backend.example.com/max_fails\" was updated with replacing."
}
```

Проверим изменения:

```
$ curl http://127.0.0.1/config/http/upstreams/backend/servers/backend.example.com
```

```
{
  "max_conns": 5,
  "max_fails": 2
}
```

DELETE

HTTP-метод DELETE удаляет *ранее заданные* настройки по указанному пути; при этом восстанавливаются значения по умолчанию, если они есть.

Например, чтобы удалить измененный ранее параметр `max_fails` у сервера `backend.example.com` в апстриме `backend`:

```
$ curl -X DELETE \
  http://127.0.0.1/config/http/upstreams/backend/servers/backend.example.com/max_
↪fails
```

```
{
  "success": "Reset",
  "description": "Configuration API entity \"/config/http/upstreams/backend/servers/
↪backend.example.com/max_fails\" was reset to default."
}
```

Проверим изменения с аргументом `defaults=on`:

```
$ curl http://127.0.0.1/config/http/upstreams/backend/servers/backend.example.com?
↪defaults=on
```

```
{
  "weight": 1,
  "max_conns": 5,
  "max_fails": 1,
  "fail_timeout": 10,
  "slow_start": 0,
  "backup": false,
  "down": false,
}
```

```
"sid": ""
}
```

Параметр `max_fails` вернулся к значению по умолчанию.

При удалении серверов можно установить аргумент `connection_drop=<значение>`, чтобы переопределить настройки `proxy_connection_drop`, `grpc_connection_drop`, `fastcgi_connection_drop`, `scgi_connection_drop` и `uwsgi_connection_drop`:

```
$ curl -X DELETE \
  http://127.0.0.1/config/http/upstreams/backend/servers/backend1.example.com?
↪connection_drop=off

$ curl -X DELETE \
  http://127.0.0.1/config/http/upstreams/backend/servers/backend2.example.com?
↪connection_drop=on

$ curl -X DELETE \
  http://127.0.0.1/config/http/upstreams/backend/servers/backend3.example.com?
↪connection_drop=1000
```

PATCH

HTTP-метод PATCH позволяет создать новую сущность по указанному пути либо частично заменить или дополнить существующую (RFC 7386), отправив в данных JSON-определение.

Метод работает так: если сущности, указанные в новом определении, уже есть в конфигурации, они будут перезаписаны; если их нет, то они будут добавлены.

Например, чтобы поменять значение параметра `down` у сервера `backend.example.com` в апстриме `backend`, оставив прочее без изменений:

```
$ curl -X PATCH -d '{ "down": true }' \
  http://127.0.0.1/config/http/upstreams/backend/servers/backend.example.com
```

```
{
  "success": "Updated",
  "description": "Existing configuration API entity \"/config/http/upstreams/
↪backend/servers/backend.example.com\" was updated with merging."
}
```

Проверим изменения:

```
$ curl http://127.0.0.1/config/http/upstreams/backend/servers/backend.example.com
```

```
{
  "max_conns": 5,
  "down": true
}
```

Обратите внимание, что переданный с запросом PATCH JSON-объект *слился* с уже существующим, а не заменил его целиком, как было бы с PUT.

Особый случай представляют значения `null`; они используются для удаления отдельных элементов конфигурации в ходе такого слияния.

Примечание

Такое удаление аналогично действию DELETE; в частности, восстанавливаются значения по умолчанию.

Например, чтобы удалить добавленный ранее параметр `down` и одновременно с этим изменить `max_conns`:

```
$ curl -X PATCH -d '{ "down": null, "max_conns": 6 }' \
  http://127.0.0.1/config/http/upstreams/backend/servers/backend.example.com
```

```
{
  "success": "Updated",
  "description": "Existing configuration API entity \"/config/http/upstreams/
  ↪ backend/servers/backend.example.com\" was updated with merging."
}
```

Проверим изменения:

```
$ curl http://127.0.0.1/config/http/upstreams/backend/servers/backend.example.com
```

```
{
  "max_conns": 6
}
```

Параметр `down`, для которого было передано значение `null`, удален; значение `max_conns` изменено.

Auth Basic

Позволяет ограничить доступ к ресурсам с проверкой имени и пароля пользователя по протоколу "HTTP Basic Authentication".

Ограничить доступ можно также по адресу или по результату подзапроса. Одновременное ограничение доступа по адресу и паролю управляется директивой *satisfy*.

Пример конфигурации

```
location / {
  auth_basic          "closed site";
  auth_basic_user_file conf/htpasswd;
}
```

Директивы

auth_basic

Синтаксис	<code>auth_basic строка off;</code>
По умолчанию	<code>auth_basic off;</code>
Контекст	<code>http, server, location, limit_except</code>

Включает проверку имени и пароля пользователя по протоколу "HTTP Basic Authentication". Заданный параметр используется в качестве *realm*. В значении параметра можно использовать переменные.

<code>off</code>	отменяет действие унаследованной с предыдущего уровня конфигурации директивы <i>auth_basic</i>
------------------	--

auth_basic_user_file

Синтаксис	auth_basic_user_file <i>файл</i> ;
По умолчанию	—
Контекст	http, server, location, limit_except

Задаёт *файл*, в котором хранятся имена и пароли пользователей. Формат файла следующий:

```
# комментарий
имя1:пароль1
имя2:пароль2:комментарий
имя3:пароль3
```

В имени *файла* можно использовать переменные.

Поддерживаются следующие типы паролей:

- зашифрованные функцией *crypt()*; могут быть созданы с помощью утилиты **htpasswd** из дистрибутива HTTP-сервера Apache или команды "*openssl passwd*";
- хэшированные с помощью алгоритма, основанного на MD5, по версии Apache (**apr1**); могут быть созданы теми же инструментами;
- заданные согласно синтаксису "*{схема}данные*" как описано в [RFC 2307](#); в настоящий момент реализованы схемы *PLAIN* (в качестве примера, не следует применять), *SHA* (простое SHA-1 хэширование, не следует применять) и *SSHA* (SHA-1 хэширование с солью, используется в некоторых программах, в частности OpenLDAP и Dovecot).

Предупреждение

Поддержка схемы SHA была добавлена лишь для облегчения процесса миграции файлов паролей с других веб-серверов. Ее не следует применять для новых паролей, т.к. используемое при этом SHA-1 хэширование без соли уязвимо к взлому при помощи [радужных таблиц](#).

Auth Request

Предоставляет возможность авторизации клиента, основанной на результате подзапроса. Если подзапрос возвращает код ответа 2xx, доступ разрешается. Если 401 или 403 — доступ запрещается с соответствующим кодом ошибки. Любой другой код ответа, возвращаемый подзапросом, считается ошибкой.

При ошибке 401 клиенту также передается заголовок **WWW-Authenticate** из ответа подзапроса.

Пример конфигурации

```
location /private/ {
    auth_request /auth;
    # ...
}

location = /auth {
    proxy_pass ...;
    proxy_pass_request_body off;
    proxy_set_header Content-Length "";
    proxy_set_header X-Original-URI $request_uri;
}
```


Директивы

auth_request

Синтаксис	<code>auth_request uri off;</code>
По умолчанию	<code>auth_request off;</code>
Контекст	<code>http, server, location</code>

Включает авторизацию, основанную на результате выполнения подзапроса, и задает URI, на который будет отправлен подзапрос.

auth_request_set

Синтаксис	<code>auth_request_set \$переменная значение;</code>
По умолчанию	—
Контекст	<code>http, server, location</code>

Устанавливает переменную в запросе в заданное значение после завершения запроса авторизации. Значение может содержать переменные из запроса авторизации, например, `$upstream_http_*`.

AutoIndex

Обслуживает запросы, оканчивающиеся косой чертой (/), и выдает листинг каталога. Обычно запрос попадает к модулю AutoIndex, когда модуль Index не нашел индексный файл.

Пример конфигурации

```
location / {
    autoindex on;
}
```

Директивы

autoindex

Синтаксис	<code>autoindex on off;</code>
По умолчанию	<code>autoindex off;</code>
Контекст	<code>http, server, location</code>

Разрешает или запрещает вывод листинга каталога.

autoindex_exact_size

Синтаксис	<code>autoindex_exact_size on off;</code>
По умолчанию	<code>autoindex_exact_size on;</code>
Контекст	<code>http, server, location</code>

Для формата HTML определяет, как выводить размеры файлов в листинге каталога: точно или округляя до килобайт, мегабайт и гигабайт.

autoindex_format

Синтаксис	autoindex_format html xml json jsonp;
По умолчанию	autoindex_format html;
Контекст	http, server, location

Задаёт формат вывода листинга каталога.

При использовании формата JSONP имя callback-функции задается в аргументе запроса `callback`. Если аргумент отсутствует или имеет пустое значение, то используется формат JSON.

Вывод в формате XML может быть преобразован при помощи модуля XSLT.

Форматы вывода

Поля объекта в ответах содержат следующие данные:

Поле	Описание
name	Имя файла или каталога
type	Тип объекта: <code>file</code> или <code>directory</code>
size	Размер файла в байтах; для каталогов отсутствует. Директива <code>autoindex_exact_size</code> влияет только на формат <code>html</code>
mtime	Время последнего изменения в формате Unix time

HTML

```
<html>
<head>
  <title>Index of /files/</title>
</head>
<body>
  <h1>Index of /files/</h1>
  <hr>
  <pre>
    <a href="..">../</a>
    <a href="example.txt">example.txt</a>          12-Jun-2025 14:21  ┐
→1234
    <a href="image.png">image.png</a>          12-Jun-2025 14:21  ┐
→4321
  </pre>
  <hr>
</body>
</html>
```

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<listing>
<file>
  <name>example.txt</name>
  <type>file</type>
  <size>1234</size>
  <mtime>2025-06-12T14:21:00Z</mtime>
</file>
<file>
  <name>image.png</name>
```

```
<type>file</type>
<size>4321</size>
<mtime>2025-06-12T14:21:00Z</mtime>
</file>
</listing>
```

JSON

```
[
{
  "name": "example.txt",
  "type": "file",
  "size": 1234,
  "mtime": "2025-06-12T14:21:00Z"
},
{
  "name": "image.png",
  "type": "file",
  "size": 4321,
  "mtime": "2025-06-12T14:21:00Z"
}
]
```

JSONP

```
callback([
{
  "name": "example.txt",
  "type": "file",
  "size": 1234,
  "mtime": "2025-06-12T14:21:00Z"
},
{
  "name": "image.png",
  "type": "file",
  "size": 4321,
  "mtime": "2025-06-12T14:21:00Z"
}
]);
```

autoindex_localtime

Синтаксис	autoindex_localtime on off;
По умолчанию	autoindex_localtime off;
Контекст	http, server, location

Для формата HTML определяет, в какой временной зоне выводить время в листинге каталога: в локальной или в UTC.

Browser

Создает переменные, значения которых зависят от значения поля **User-Agent** в заголовке запроса.

Переменные

`$ancient_browser`

равна значению, заданному директивой `ancient_browser_value`, если браузер опознан как устаревший;

`$modern_browser`

равна значению, заданному директивой `modern_browser_value`, если браузер опознан как современный;

`$msie`

равна "1", если браузер опознан как MSIE любой версии.

Пример конфигурации

Выбор индексного файла:

```
modern_browser_value "modern.";

modern_browser msie      5.5;
modern_browser gecko     1.0.0;
modern_browser opera     9.0;
modern_browser safari    413;
modern_browser konqueror 3.0;

index index.${modern_browser}.html index.html;
```

Перенаправление для старых браузеров:

```
modern_browser msie      5.0;
modern_browser gecko     0.9.1;
modern_browser opera     8.0;
modern_browser safari    413;
modern_browser konqueror 3.0;

modern_browser unlisted;

ancient_browser Links Lynx netscape4;

if ($ancient_browser) {
    rewrite ^ /ancient.html;
}
```

Директивы

ancient_browser

Синтаксис	<code>ancient_browser строка ...;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт подстроки, при нахождении которых в поле **User-Agent** заголовка запроса браузер считается

устаревшим. Специальная строка "netscape4" соответствует регулярному выражению "^Mozilla/[1-4]\$".

ancient_browser_value

Синтаксис	<code>ancient_browser_value</code> строка;
По умолчанию	<code>ancient_browser_value 1;</code>
Контекст	http, server, location

Задаёт значение для переменных `$ancient_browser`.

modern_browser

Синтаксис	<code>modern_browser</code> браузер версия; <code>modern_browser</code> <code>unlisted</code> ;
По умолчанию	—
Контекст	http, server, location

Задаёт версию браузера, начиная с которой он считается современным. В качестве браузера можно задать *msie*, *gecko* (браузеры, созданные на основе Mozilla), *opera*, *safari* или *konqueror*.

Версии можно задать в форматах X, X.X, X.X.X или X.X.X.X. Максимальные значения для каждого из форматов соответственно — 4000, 4000.99, 4000.99.99 и 4000.99.99.99.

Специальное значение `unlisted` указывает считать современным браузер, не описанный директивами `modern_browser` и `ancient_browser`. В противном случае неперечисленный браузер будет считаться устаревшим. Если в заголовке запроса нет поля **User-Agent**, то браузер считается неперечисленным.

modern_browser_value

Синтаксис	<code>modern_browser_value</code> строка;
По умолчанию	<code>modern_browser_value 1;</code>
Контекст	http, server, location

Задаёт значение для переменных `$modern_browser`.

Charset

Добавляет указанную кодировку в поле **Content-Type** заголовка ответа. Кроме того, модуль может перекодировать данные из одной кодировки в другую с некоторыми ограничениями:

- перекодирование осуществляется только в одну сторону — от сервера к клиенту,
- перекодироваться могут только однобайтные кодировки
- или однобайтные кодировки в UTF-8 и обратно.

Пример конфигурации

```
include conf/koi-win;
```

```
charset windows-1251;
source_charset koi8-r;
```

Директивы

charset

Синтаксис	charset <i>кодировка</i> off ;
По умолчанию	charset off ;
Контекст	http, server, location, if в location

Добавляет указанную кодировку в поле **Content-Type** заголовка ответа. Если эта кодировка отличается от указанной в директиве *source_charset*, то выполняется перекодирование.

Параметр **off** отменяет добавление кодировки в поле **Content-Type** заголовка ответа.

Кодировка может быть задана с помощью переменной:

```
charset $charset;
```

В этом случае необходимо, чтобы все возможные значения переменной присутствовали хотя бы один раз в любом месте конфигурации в виде директив *charset_map*, *charset* или *source_charset*. Для кодировок *utf-8*, *windows-1251* и *koi8-r* для этого достаточно включить в конфигурацию файлы *conf/koi-win*, *conf/koi-utf* и *conf/win-utf*. Для других кодировок можно просто сделать фиктивную таблицу перекодировки, например:

```
charset_map iso-8859-5 _ { }
```

Кроме того, кодировка может быть задана в поле **X-Accel-Charset** заголовка ответа. Эту возможность можно запретить с помощью директив *proxy_ignore_headers*, *fastcgi_ignore_headers*, *uwsgi_ignore_headers*, *scgi_ignore_headers* и *grpc_ignore_headers*.

charset_map

Синтаксис	charset_map <i>кодировка1 кодировка2</i> { ... }
По умолчанию	—
Контекст	http

Описывает таблицу перекодирования из одной кодировки в другую. Таблица для обратного перекодирования строится на основании тех же данных. Коды символов задаются в шестнадцатеричном виде. Неописанные символы в пределах 80-FF заменяются на "?". При перекодировании из UTF-8 символы, отсутствующие в однобайтной кодировке, заменяются на "*Э#XXXX*".

Пример:

```
charset_map koi8-r windows-1251 {
    C0 FE ; # small yu
    C1 E0 ; # small a
    C2 E1 ; # small b
    C3 F6 ; # small ts
}
```

При описании таблицы перекодирования в UTF-8, коды кодировки UTF-8 должны быть указаны во второй колонке, например:

```
charset_map koi8-r utf-8 {
    C0 D18E ; # small yu
    C1 D0B0 ; # small a
    C2 D0B1 ; # small b
    C3 D186 ; # small ts
}
```

Полные таблицы преобразования из *koi8-r* в *windows-1251* и из *koi8-r* и *windows-1251* в *utf-8* входят в дистрибутив и находятся в файлах *conf/koi-win*, *conf/koi-utf* и *conf/win-utf*.

charset_types

Синтаксис	<code>charset_types mime-mun ...;</code>
По умолчанию	<code>charset_types text/html text/xml text/plain text/vnd.wap.wml application/javascript application/rss+xml;</code>
Контекст	<code>http, server, location</code>

Разрешает работу модуля в ответах с указанными MIME-типами в дополнение к `text/html`. Специальное значение `*` соответствует любому MIME-типу.

override_charset

Синтаксис	<code>override_charset on off;</code>
По умолчанию	<code>override_charset off;</code>
Контекст	<code>http, server, location, if в location</code>

Определяет, выполнять ли перекодирование для ответов, полученных от проксированного сервера или от FastCGI/uwsgi/SCGI/gRPC-сервера, если в ответах уже указана кодировка в поле **Content-Type** заголовка ответа. Если перекодирование разрешено, то в качестве исходной кодировки используется кодировка, указанная в полученном ответе.

Примечание

Если ответ был получен в подзапросе, то, независимо от значения директивы *override_charset*, всегда выполняется перекодирование из кодировки ответа в кодировку основного запроса.

source_charset

Синтаксис	<code>source_charset кодировка;</code>
По умолчанию	—
Контекст	<code>http, server, location, if в location</code>

Задаёт исходную кодировку ответа. Если эта кодировка отличается от указанной в директиве *charset*, то выполняется перекодирование.

DAV

Предназначен для автоматизации задач управления файлами на сервере по протоколу WebDAV. Модуль обрабатывает HTTP- и WebDAV-методы PUT, DELETE, MKCOL, COPY и MOVE.

Примечание

WebDAV-клиенты, которые требуют для работы дополнительных WebDAV-методов, не будут работать с этим модулем.

Пример конфигурации

```
location / {
    root                /data/www;

    client_body_temp_path /data/client_temp;

    dav_methods PUT DELETE MKCOL COPY MOVE;

    create_full_put_path on;
    dav_access          group:rw all:r;

    limit_except GET {
        allow 192.168.1.0/32;
        deny  all;
    }
}
```

Директивы

create_full_put_path

Синтаксис	<code>create_full_put_path on off;</code>
По умолчанию	<code>create_full_put_path off;</code>
Контекст	http, server, location

По спецификации WebDAV-метод PUT может создавать файл только в уже существующем каталоге. Данная директива разрешает создавать все необходимые промежуточные каталоги.

dav_access

Синтаксис	<code>dav_access <i>пользователи:права</i> ...;</code>
По умолчанию	<code>dav_access user:rw;</code>
Контекст	http, server, location

Задаёт права доступа для создаваемых файлов и каталогов, например,

```
dav_access user:rw group:rw all:r;
```

Если заданы какие-либо права для group или all, то права для user указывать необязательно:


```
dav_access group:rw all:r;
```

dav_methods

Синтаксис	<code>dav_methods off <i>метод</i> ...;</code>
По умолчанию	<code>dav_methods off;</code>
Контекст	<code>http, server, location</code>

Разрешает указанные HTTP- и WebDAV-методы. Параметр `off` запрещает все методы, обрабатываемые данным модулем. Поддерживаются следующие методы: PUT, DELETE, MKCOL, COPY и MOVE.

Файл, загружаемый методом PUT, записывается во временный файл, а потом этот файл переименовывается. Временный файл и его постоянное место хранения могут располагаться на разных файловых системах. Однако нужно учитывать, что в этом случае вместо дешевой операции переименовывания в пределах одной файловой системы файл копируется с одной файловой системы на другую. Поэтому лучше, если сохраняемые файлы будут находиться на той же файловой системе, что и каталог с временными файлами, задаваемый директивой `client_body_temp_path` для данного `location`.

При создании файла с помощью метода PUT можно задать дату модификации, передав ее в поле заголовка `Date`.

min_delete_depth

Синтаксис	<code>min_delete_depth <i>число</i>;</code>
По умолчанию	<code>min_delete_depth 0;</code>
Контекст	<code>http, server, location</code>

Разрешает методу DELETE удалять файлы при условии, что число элементов в пути запроса не меньше заданного. Например, директива

```
min_delete_depth 4;
```

разрешает удалять файлы по запросам

```
/users/00/00/name
/users/00/00/name/pic.jpg
/users/00/00/page.html
```

и запрещает удаление

```
/users/00/00
```

DoH

Реализует сервер DNS over HTTPS (DoH) в соответствии с [RFC 8484](#). Принимает DNS-запросы, передаваемые по HTTP/HTTPS, и проксирует их к группе DNS-серверов по UDP или TCP.

DoH-клиент отправляет DNS-запрос либо методом GET, передавая DNS-сообщение в закодированном Base64URL виде в аргументе строки запроса `dns`, либо методом POST, передавая необработанное DNS-сообщение в теле запроса. В обоих случаях тип содержимого DNS-сообщения — `application/dns-message`.

Принятый модулем DNS-ответ с ненулевым значением `RCODE` возвращается клиенту со статусом 200 (OK). Значение `RCODE` сохраняется в DNS-сообщении и не приводит к повторной попытке по условию `invalid_response` директивы `doh_next_upstream`.

Директива `doh_transport` определяет использование UDP или TCP для передачи запросов. Для защиты от подмены ответов сторонним источником модуль подставляет случайный идентификатор запроса, отправляемого DNS-серверу, и восстанавливает исходный идентификатор клиента в ответе.

Примечание

Эта рандомизация идентификатора запроса использует некриптографический генератор и обеспечивает лишь ограниченную защиту; используйте DNS-серверы, доступные в защищенной доверенной сети, как это рекомендуется для директивы `resolver`.

Поле заголовка ответа `Cache-Control` формируется на основе минимального TTL из ответа: оно устанавливается в `max-age=<ttl>` либо в `no-store`, если пригодное для кэширования значение TTL определить не удалось.

Пример конфигурации

Проксирование запросов DNS over HTTPS к группе DNS-серверов:

```
http {
    upstream dns_upstream {
        server 127.0.0.1:53;
    }

    server {
        listen 443 ssl;
        server_name dns.example.com;

        ssl_certificate     dns.example.com.crt;
        ssl_certificate_key dns.example.com.key;

        location /dns-query {
            doh_pass dns_upstream;
        }
    }
}
```

Директивы

`doh_bind`

Синтаксис	<code>doh_bind адрес [transparent] off;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт локальный IP-адрес с необязательным портом, который будет использоваться в исходящих соединениях с DNS-сервером. В значении параметра можно использовать переменные. Специальное значение `off` отменяет действие унаследованной с более высокого уровня конфигурации директивы `doh_bind`, позволяя системе самостоятельно выбирать локальный IP-адрес и порт.

Параметр `transparent` позволяет задать нелокальный IP-адрес, который будет использоваться в исходящих соединениях с DNS-сервером, например, реальный IP-адрес клиента:

```
doh_bind $remote_addr transparent;
```

doh_buffer_size

Синтаксис	doh_buffer_size размер;
По умолчанию	doh_buffer_size 16k;
Контекст	http, server, location

Задаёт размер буфера, в который читается ответ, получаемый от DNS-сервера. Значение не может быть меньше 512 байт.

doh_connect_timeout

Синтаксис	doh_connect_timeout время;
По умолчанию	doh_connect_timeout 60s;
Контекст	http, server, location

Задаёт таймаут для установления соединения с DNS-сервером.

Примечание

Этот таймаут не управляет неактивными соединениями: время, в течение которого уже открытое соединение сохраняется для повторного использования, задается директивой *keepalive_timeout* в блоке upstream, соединения которого кэшируются директивой *keepalive*. Кэшируются только TCP-соединения, см. *doh_transport*.

doh_max_size

Синтаксис	doh_max_size размер;
По умолчанию	размер одной страницы памяти операционной системы
Контекст	http, server, location

Ограничивает размер DNS-запроса, принимаемого от клиента. Значение **размера** должно быть не меньше 12 байт. Запрос отклоняется, если его он меньше 12 байт (это размер заголовка DNS-сообщения) или превышает **размер**.

doh_next_upstream

Синтаксис	doh_next_upstream error timeout invalid_response off ...;
По умолчанию	doh_next_upstream error timeout;
Контекст	http, server, location

Определяет, в каких случаях запрос будет передан следующему серверу в группе upstream:

error	произошла ошибка при установлении соединения с сервером, передаче ему запроса или чтении ответа;
timeout	произошел таймаут при установлении соединения с сервером, передаче ему запроса или чтении ответа;
invalid_response	сервер вернул некорректный DNS-ответ;
off	запрещает передачу запроса следующему серверу.

Передача запроса следующему серверу может быть ограничена по количеству попыток и по времени.

doh_next_upstream_timeout

Синтаксис	doh_next_upstream_timeout время;
По умолчанию	doh_next_upstream_timeout 0;
Контекст	http, server, location

Ограничивает время, в течение которого запрос может быть передан следующему серверу в группе. Значение 0 отключает это ограничение.

doh_next_upstream_tries

Синтаксис	doh_next_upstream_tries <i>число</i> ;
По умолчанию	doh_next_upstream_tries 0;
Контекст	http, server, location

Ограничивает число попыток передачи запроса следующему серверу в группе. Значение 0 отключает это ограничение.

doh_pass

Синтаксис	doh_pass <i>адрес</i> ;
По умолчанию	—
Контекст	location, if в location

Задаёт адрес DNS-сервера, которому проксируются DNS-запросы. Адрес может быть доменным именем или IP-адресом с указанием порта:

```
doh_pass 127.0.0.1:53;
```

или в виде группы серверов:

```
doh_pass dns_upstream;
```

Запросы передаются серверу через UDP или TCP в соответствии с директивой *doh_transport*. В одной локации может быть указана только одна директива *doh_pass*.

doh_read_timeout

Синтаксис	<code>doh_read_timeout</code> время;
По умолчанию	<code>doh_read_timeout 60s</code> ;
Контекст	<code>http, server, location</code>

Задаёт таймаут при чтении ответа DNS-сервера. Таймаут устанавливается не на всю передачу ответа, а только между двумя операциями чтения. Если по истечении этого времени DNS-сервер ничего не передаст, соединение закрывается.

doh_send_timeout

Синтаксис	<code>doh_send_timeout</code> время;
По умолчанию	<code>doh_send_timeout 60s</code> ;
Контекст	<code>http, server, location</code>

Задаёт таймаут при передаче запроса DNS-серверу. Таймаут устанавливается не на всю передачу запроса, а только между двумя операциями записи. Если по истечении этого времени DNS-сервер ничего не получит, соединение закрывается.

doh_socket_keepalive

Синтаксис	<code>doh_socket_keepalive on off</code> ;
По умолчанию	<code>doh_socket_keepalive off</code> ;
Контекст	<code>http, server, location</code>

Включает параметр сокета `SO_KEEPALIVE` для исходящих соединений с DNS-сервером. При указании `off` действуют системные настройки сокета.

doh_transport

Синтаксис	<code>doh_transport tcp udp auto</code> ;
По умолчанию	<code>doh_transport auto</code> ;
Контекст	<code>http, server, location</code>

Выбирает транспортный протокол, используемый для передачи DNS-запросов DNS-серверу:

<code>auto</code>	если в кэше соединений <code>keepalive</code> группы серверов есть свободное TCP-соединение, запрос отправляется по нему (TCP); в противном случае он сначала отправляется по UDP, а при усеченном ответе сервера (установлен флаг <code>TC</code>) повторяется по TCP;
<code>udp</code>	запросы отправляются только по UDP; усеченный ответ считается ошибкой, а запрос не повторяется по TCP;
<code>tcp</code>	запросы отправляются только по TCP.

Empty GIF

Выдает однопиксельный прозрачный GIF.

Пример конфигурации

```
location = /_gif {
    empty_gif;
}
```

Директивы

empty_gif

Синтаксис	<code>empty_gif;</code>
По умолчанию	—
Контекст	<code>location</code>

Разрешает в содержащем `location` выдавать однопиксельный прозрачный GIF.

FastCGI

Позволяет передавать запросы FastCGI-серверу.

Пример конфигурации

```
location / {
    fastcgi_pass localhost:9000;
    fastcgi_index index.php;

    fastcgi_param SCRIPT_FILENAME /home/www/scripts/php$fastcgi_script_name;
    fastcgi_param QUERY_STRING $query_string;
    fastcgi_param REQUEST_METHOD $request_method;
    fastcgi_param CONTENT_TYPE $content_type;
    fastcgi_param CONTENT_LENGTH $content_length;
}
```

Директивы

fastcgi_bind

Синтаксис	<code>fastcgi_bind адрес [transparent] off;</code>
По умолчанию	—
Контекст	<code>http</code> , <code>server</code> , <code>location</code>

Задаёт локальный IP-адрес с необязательным портом, который будет использоваться в исходящих соединениях с FastCGI-сервером. В значении параметра можно использовать переменные. Специальное значение `off` отменяет действие унаследованной с предыдущего уровня конфигурации директивы `fastcgi_bind`, позволяя системе самостоятельно выбирать локальный IP-адрес и порт.

По умолчанию это значение вычисляется один раз при инициализации запроса, до выбора FastCGI-сервера. В Angie PRO вычисление происходит после каждого выбора FastCGI-сервера (в том числе

при повторных попытках), поэтому адрес привязки может зависеть от выбранного сервера; например, с помощью переменной `$upstream_current_addr`. Если вычисленное значение пусто, привязка не выполняется.

Параметр `transparent` позволяет задать нелокальный IP-адрес, который будет использоваться в исходящих соединениях с FastCGI-сервером, например, реальный IP-адрес клиента:

```
fastcgi_bind $remote_addr transparent;
```

Для работы параметра обычно требуется запустить рабочие процессы Angie с привилегиями суперпользователя. В Linux это не требуется, так как если указан параметр `transparent`, то рабочие процессы наследуют `capability CAP_NET_RAW` из главного процесса.

Примечание

Необходимо настроить таблицу маршрутизации ядра для перехвата сетевого трафика с FastCGI-сервера.

`fastcgi_buffer_size`

Синтаксис	<code>fastcgi_buffer_size размер;</code>
По умолчанию	<code>fastcgi_buffer_size 4k 8k;</code>
Контекст	<code>http, server, location</code>

Задаёт размер буфера, в который будет читаться первая часть ответа, получаемого от FastCGI-сервера. В этой части ответа обычно находится небольшой заголовок ответа. По умолчанию размер одного буфера равен размеру страницы памяти. В зависимости от платформы это или 4К, или 8К, однако его можно сделать меньше.

`fastcgi_buffering`

Синтаксис	<code>fastcgi_buffering on off;</code>
По умолчанию	<code>fastcgi_buffering on;</code>
Контекст	<code>http, server, location</code>

Разрешает или запрещает использовать буферизацию ответов FastCGI-сервера.

on	Angie отправляет ответ клиенту параллельно с чтением ответа от FastCGI-сервера; клиент не ждет ответа целиком. Ответ читается в буферы, заданные директивами <i>fastcgi_buffer_size</i> и <i>fastcgi_buffers</i> . Буфер передается на отправку, как только заполнится целиком, причем суммарный размер буферов, занятых отправкой, ограничен значением <i>fastcgi_busy_buffers_size</i> . Заполненный частично буфер ждет, пока не заполнится или пока не закончится ответ, поэтому данные обычно попадают к клиенту целыми буферами. Часть ответа записывается на диск во временный файл, когда все буферы заполнены, а клиент читает медленнее, чем отдает FastCGI-сервер. Ответы, которые кэшируются или сохраняются на диске, всегда проходят через временный файл. Запись во временные файлы контролируется директивами <i>fastcgi_max_temp_file_size</i> и <i>fastcgi_temp_file_write_size</i> . При буферизации FastCGI-сервер освобождается сразу, как только отдаст ответ целиком: соединение с ним не удерживается все то время, пока ответ читает медленный клиент.
off	Каждая порция данных, прочитанная от FastCGI-сервера, сразу же передается клиенту, без какого-либо минимального размера: прочитанные 10 байт из буфера 4K отправляются немедленно. Angie использует один буфер, заданный директивой <i>fastcgi_buffer_size</i> , поэтому за один раз он читает не больше этого размера. Временный файл не используется, поэтому ответы не кэшируются и не сохраняются на диске, а <i>fastcgi_limit_rate</i> и <i>limit_rate</i> не действуют. Режим off подходит для ответов, которые должны доходить до клиента по мере их формирования: Server-Sent Events, длинные опросы (long polling), потоковая передача.

Буферизация может быть также включена или выключена путем передачи значения "yes" или "no" в поле **X-Accel-Buffering** заголовка ответа. Эту возможность можно запретить с помощью директивы *fastcgi_ignore_headers*.

fastcgi_buffers

Синтаксис	fastcgi_buffers <i>число</i> размер;
По умолчанию	fastcgi_buffers 8 4k 8k;
Контекст	http, server, location

Задаёт число и размер буферов для одного соединения, в которые будет читаться ответ, получаемый от FastCGI-сервера.

По умолчанию размер одного буфера равен размеру страницы. В зависимости от платформы это или 4K, или 8K.

fastcgi_busy_buffers_size

Синтаксис	fastcgi_busy_buffers_size размер;
По умолчанию	fastcgi_busy_buffers_size 8k 16k;
Контекст	http, server, location

При включенной буферизации ответов FastCGI-сервера, ограничивает суммарный размер буферов, которые могут быть заняты для отправки ответа клиенту, пока ответ еще не прочитан целиком. Оставшиеся буферы тем временем могут использоваться для чтения ответа и, при необходимости, буферизации части ответа во временный файл.

По умолчанию размер ограничен величиной двух буферов, заданных директивами `fastcgi_buffer_size` и `fastcgi_buffers`.

`fastcgi_cache`

Синтаксис	<code>fastcgi_cache зона off;</code>
По умолчанию	<code>fastcgi_cache off;</code>
Контекст	<code>http, server, location</code>

Задаёт зону разделяемой памяти, используемую для кэширования. Одна и та же зона может использоваться в нескольких местах. В значении параметра можно использовать переменные. Параметр `off` запрещает кэширование, унаследованное с предыдущего уровня конфигурации.

Примечание

При использовании `fastcgi_cache` обычно необходимо также задать директиву `fastcgi_cache_valid`, чтобы явно указать время действия кэшированных ответов. Если она не задана, Angie не использует значения по умолчанию, а определяет время хранения ответа в кэше на основе HTTP-заголовков ответа от сервера в следующем порядке приоритета:

1. Заголовок `X-Accel-Expires` (наивысший приоритет).
2. Заголовок `Cache-Control` с параметрами `max-age` или `s-maxage`.
3. Заголовок `Expires`.

Если ни один заголовок не содержит допустимого значения или их нет вовсе, ответ не будет кэшироваться, так как определить срок его действия нельзя.

`fastcgi_cache_background_update`

Синтаксис	<code>fastcgi_cache_background_update on off;</code>
По умолчанию	<code>fastcgi_cache_background_update off;</code>
Контекст	<code>http, server, location</code>

Позволяет запустить фоновый подзапрос для обновления просроченного элемента кэша, в то время как клиенту возвращается устаревший кэшированный ответ. Использование устаревшего кэшированного ответа в момент его обновления должно быть разрешено.

`fastcgi_cache_bypass`

Синтаксис	<code>fastcgi_cache_bypass строка ...;</code>
По умолчанию	—
Контекст	<code>http, server, location</code>

Задаёт условия, при которых ответ не будет браться из кэша. Если значение хотя бы одного из строковых параметров непустое и не равно "0", то ответ не берется из кэша:

```
fastcgi_cache_bypass $cookie_nocache $arg_nocache$arg_comment;
fastcgi_cache_bypass $http_pragma $http_authorization;
```

В значении параметров можно использовать переменные.

Можно использовать совместно с директивой *fastcgi_no_cache*.

fastcgi_cache_key

Синтаксис	<code>fastcgi_cache_key строка;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт ключ для кэширования. В значении ключа можно использовать переменные. Например,

```
fastcgi_cache_key localhost:9000$request_uri;
```

fastcgi_cache_lock

Синтаксис	<code>fastcgi_cache_lock on off;</code>
По умолчанию	<code>fastcgi_cache_lock off;</code>
Контекст	http, server, location

Если включено, одновременно только одному запросу будет позволено заполнить новый элемент кэша, идентифицируемый согласно директиве *fastcgi_cache_key*, передав запрос на FastCGI-сервер. Остальные запросы этого же элемента будут либо ожидать появления ответа в кэше, либо освобождения блокировки этого элемента, в течение времени, заданного директивой *fastcgi_cache_lock_timeout*.

fastcgi_cache_lock_age

Синтаксис	<code>fastcgi_cache_lock_age время;</code>
По умолчанию	<code>fastcgi_cache_lock_age 5s;</code>
Контекст	http, server, location

Если последний запрос, переданный на FastCGI-сервер для заполнения нового элемента кэша, не завершился за указанное время, на FastCGI-сервер может быть передан ещё один запрос.

fastcgi_cache_lock_timeout

Синтаксис	<code>fastcgi_cache_lock_timeout время;</code>
По умолчанию	<code>fastcgi_cache_lock_timeout 5s;</code>
Контекст	http, server, location

Задаёт таймаут для *fastcgi_cache_lock*. По истечении указанного времени запрос будет передан на FastCGI-сервер, однако ответ не будет кэширован.

fastcgi_cache_max_range_offset

Синтаксис	<code>fastcgi_cache_max_range_offset <i>число</i>;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт смещение в байтах для запросов с указанием диапазона запрашиваемых байт (byte-range requests). Если диапазон находится за указанным смещением, range-запрос будет передан на FastCGI-сервер и ответ не будет кэширован.

fastcgi_cache_methods

Синтаксис	<code>fastcgi_cache_methods GET HEAD POST ...;</code>
По умолчанию	<code>fastcgi_cache_methods GET HEAD;</code>
Контекст	http, server, location

Если метод запроса клиента указан в этой директиве, то ответ будет кэширован. Методы "GET" и "HEAD" всегда добавляются в список, но тем не менее рекомендуется перечислять их явно. См. также директиву `fastcgi_no_cache`.

fastcgi_cache_min_uses

Синтаксис	<code>fastcgi_cache_min_uses <i>число</i>;</code>
По умолчанию	<code>fastcgi_cache_min_uses 1;</code>
Контекст	http, server, location

Задаёт число запросов, после которого ответ будет кэширован.

Предупреждение

Метаданные кэша хранятся в разделяемой памяти. Ручное удаление файлов кэша не сбрасывает счетчики и может привести к непредсказуемому поведению. Для полного сброса остановите сервер, удалите директорию кэша и запустите снова.

Примечание

Сторонние модули очистки кэша (например, Cache Purge) удаляют только файлы, но не сбрасывают счетчик `fastcgi_cache_min_uses`. Директива предназначена для защиты кэша от загрязнения редкими запросами, и сброс счетчика при очистке может негативно повлиять на производительность.

fastcgi_cache_path

Синтаксис	<code>fastcgi_cache_path <i>путь</i> [levels=<i>уровни</i>] [use_temp_path=on off] [keys_zone=<i>имя:размер</i>[:<i>file=файл</i>] [inactive=<i>время</i>] [max_size=<i>размер</i>] [min_free=<i>размер</i>] [manager_files=<i>число</i>] [manager_sleep=<i>время</i>] [manager_threshold=<i>время</i>] [loader_files=<i>число</i>] [loader_sleep=<i>время</i>] [loader_threshold=<i>время</i>];</code>
По умолчанию	—
Контекст	http, server, location

Задаёт путь и другие параметры кэша. Данные кэша хранятся в файлах. Ключом и именем файла в кэше является результат функции MD5 от проксированного URL.

levels	задаёт уровни иерархии кэша: можно задать от 1 до 3 уровней, на каждом уровне допускаются значения 1 или 2.
---------------	---

Например, при использовании

```
fastcgi_cache_path /data/angie/cache levels=1:2 keys_zone=one:10m;
```

имена файлов в кэше будут такого вида:

```
/data/angie/cache/c/29/b7f54b2df7773722d382f4809d65029c
```

Кэшируемый ответ сначала записывается во временный файл, а потом этот файл переименовывается. Временные файлы и кэш могут располагаться на разных файловых системах. Однако нужно учитывать, что в этом случае вместо дешевой операции переименовывания в пределах одной файловой системы файл копируется с одной файловой системы на другую. Поэтому лучше, если кэш будет находиться на той же файловой системе, что и каталог с временными файлами.

<code>use_temp_path=on</code>	определяет каталог, который будет использоваться для временных файлов
<code> off</code>	
<code>on</code>	Если параметр не задан или установлен в значение <code>on</code> , будет использоваться каталог, задаваемый директивой <code>fastcgi_temp_path</code> для данного <code>location</code>
<code>off</code>	временные файлы будут располагаться непосредственно в каталоге кэша
<code>keys_zone</code>	Задаёт имя и размер зоны разделяемой памяти для хранения всех активных ключей и информации о данных. Метаданные кэша хранятся в разделяемой памяти. Размер зоны задается в байтах и должен составлять не менее двух системных страниц памяти. Также допустимы суффиксы <code>k</code> и <code>m</code>; суффикс <code>g</code>, в отличие от <code>max_size</code>, здесь не поддерживается. Зоны размером в 1 мегабайт достаточно для хранения около 8000 ключей. Когда с <code>keys_zone</code> задан необязательный <i>файл</i>, Angie сбрасывает содержимое этой зоны на диск при завершении работы основного процесса и пытается восстановить ее по тому же адресу памяти при следующем запуске или после обновления бинарных файлов, чтобы обеспечить более надежную сохранность данных и сократить время загрузки кэша. Если восстановление области невозможно из-за изменения ее размера, несовместимости версий бинарных файлов или по другим причинам, Angie запишет в журнал предупреждение (<code>failed to restore zone at address</code>) и не будет использовать механизм восстановления зоны. Вместо этого несовместимый файл будет переименован в <code>.old</code>; вы можете либо удалить его, либо восстановить его имя и вернуть Angie к конфигурации и версии, в которых этот файл был изначально создан.
 Предупреждение Убедитесь, что путь к <i>файлу</i> указан правильно и имеет необходимые права доступа для использования Angie, а также защищен от несанкционированного доступа; относительные пути основаны на префиксе.	
<code>inactive</code>	Если к данным кэша не обращаются в течение времени, заданного этим параметром, они удаляются независимо от их свежести; каждое обращение сбрасывает этот таймер. Это отличается от вытеснения по размеру, выполняемого менеджером кэша (см. ниже): удаление по неактивности происходит, даже когда кэш не достиг <code>max_size</code>, и не зависит от сроков кэширования, заданных директивой <code>fastcgi_cache_valid</code>. По умолчанию <code>inactive</code> равен 10 минутам.

Специальный процесс **менеджера кэша** следит за максимальным размером кэша, а также за минимальным объемом свободного места на файловой системе с кэшем, и удаляет наименее востребованные данные при превышении максимального размера кэша или недостаточном объеме свободного места. Удаление данных происходит итерациями.

max_size	максимальное пороговое значение размера кэша Значение задается в байтах; допустимы суффиксы k , m и g . По умолчанию размер кэша не ограничен.
min_free	минимальное пороговое значение объема свободного места на файловой системе с кэшем Задается в тех же единицах, что и max_size . По умолчанию 0 — отслеживание свободного места отключено. На платформах, где статистика файловой системы недоступна, параметр игнорируется, а Angie записывает в журнал предупреждение min_free is not supported on this platform, ignored .
manager_files	максимальное количество удаляемых элементов кэша за одну итерацию По умолчанию: 100
manager_threshold	ограничивает время работы одной итерации По умолчанию: 200 миллисекунд
manager_sleep	время, в течение которого выдерживается пауза между итерациями По умолчанию: 50 миллисекунд

Через минуту после старта Angie активируется специальный процесс **загрузчика кэша**, который загружает в зону кэша информацию о ранее кэшированных данных, хранящихся на файловой системе. Загрузка также происходит итерациями.

loader_files	максимальное количество элементов кэша к загрузке в одну итерацию По умолчанию: 100
loader_threshold	ограничивает время работы одной итерации По умолчанию: 200 миллисекунд
loader_sleep	время, в течение которого выдерживается пауза между итерациями По умолчанию: 50 миллисекунд

fastcgi_cache_revalidate

Синтаксис	fastcgi_cache_revalidate on off;
По умолчанию	fastcgi_cache_revalidate off;
Контекст	http, server, location

Разрешает ревалидацию просроченных элементов кэша при помощи условных запросов с полями заголовка **If-Modified-Since** и **If-None-Match**.

fastcgi_cache_use_stale

Синтаксис	fastcgi_cache_use_stale error timeout invalid_header updating http_500 http_503 http_403 http_404 http_429 non_idempotent off ...;
По умолчанию	fastcgi_cache_use_stale off;
Контекст	http, server, location

Определяет, в каких случаях можно использовать устаревший кэшированный ответ. Параметры директивы совпадают с параметрами директивы *fastcgi_next_upstream*.

error	позволяет использовать устаревший кэшированный ответ при невозможности выбора FastCGI-сервера для обработки запроса.
updating	дополнительный параметр, разрешает использовать устаревший кэшированный ответ, если на данный момент он уже обновляется. Это позволяет минимизировать число обращений к FastCGI-серверам при обновлении кэшированных данных.
non_idempotent	позволяет использовать устаревший кэшированный ответ, если запрос с неидемпотентным методом уже был отправлен FastCGI-серверу, а при дальнейшей работе с этим сервером произошла любая ошибка; из неидемпотентных методов кэшировать можно только POST, и он должен быть разрешен директивой <i>fastcgi_cache_methods</i> .

Использование устаревшего кэшированного ответа может также быть разрешено непосредственно в заголовке ответа на определенное количество секунд после того, как ответ устарел.

- Расширение **stale-while-revalidate** поля заголовка **Cache-Control** разрешает использовать устаревший кэшированный ответ, если на данный момент он уже обновляется.
- Расширение **stale-if-error** поля заголовка **Cache-Control** разрешает использовать устаревший кэшированный ответ в случае ошибки.

Примечание

Такой способ менее приоритетен, чем задание параметров директивы.

Чтобы минимизировать число обращений к FastCGI-серверам при заполнении нового элемента кэша, можно воспользоваться директивой *fastcgi_cache_lock*.

fastcgi_cache_valid

Синтаксис	fastcgi_cache_valid [<i>код ...</i>] время;
По умолчанию	—
Контекст	http, server, location

Задаёт время кэширования для разных кодов ответа. Например, директивы

```
fastcgi_cache_valid 200 302 10m;
fastcgi_cache_valid 404 1m;
```

задают время кэширования 10 минут для ответов с кодами 200 и 302 и 1 минуту для ответов с кодом 404.

Если указано только время кэширования,

```
fastcgi_cache_valid 5m;
```

то кэшируются только ответы 200, 301 и 302.

Кроме того, можно кэшировать любые ответы с помощью параметра **any**:

```
fastcgi_cache_valid 200 302 10m;
fastcgi_cache_valid 301 1h;
fastcgi_cache_valid any 1m;
```

Примечание

Параметры кэширования могут также быть заданы непосредственно в заголовке ответа. Такой способ приоритетнее, чем задание времени кэширования с помощью директивы.

- Поле заголовка **X-Accel-Expires** задает время кэширования ответа в секундах. Значение *0* запрещает кэшировать ответ. Если значение начинается с префикса *@*, оно задает абсолютное время в секундах с начала эпохи, до которого ответ может быть кэширован.
- Если в заголовке нет поля **X-Accel-Expires**, параметры кэширования определяются по полям заголовка **Expires** или **Cache-Control**.
- Ответ, в заголовке которого есть поле **Set-Cookie**, не будет кэшироваться.
- Ответ, в заголовке которого есть поле **Vary** со специальным значением *"*"*, не будет кэшироваться. Ответ, в заголовке которого есть поле **Vary** с другим значением, будет кэширован с учетом соответствующих полей заголовка запроса.

Обработка одного или более из этих полей заголовка может быть отключена при помощи директивы *fastcgi_ignore_headers*.

fastcgi_catch_stderr

Синтаксис	fastcgi_catch_stderr <i>строка</i> ;
По умолчанию	—
Контекст	http, server, location

Задаёт строку для поиска в потоке ошибок ответа, полученного от FastCGI-сервера. Если строка найдена, то считается, что FastCGI-сервер вернул неверный ответ. Это позволяет обрабатывать ошибки приложений в Angie, например:

```
location /php/ {
    fastcgi_pass backend:9000;
    ...
    fastcgi_catch_stderr "PHP Fatal error";
    fastcgi_next_upstream error timeout invalid_header;
}
```

fastcgi_connect_timeout

Синтаксис	fastcgi_connect_timeout время;
По умолчанию	fastcgi_connect_timeout 60s;
Контекст	http, server, location

Задаёт таймаут для установления соединения с FastCGI-сервером. Необходимо иметь в виду, что этот таймаут обычно не может превышать 75 секунд.

Примечание

Этот таймаут не управляет неактивными соединениями: время, в течение которого уже открытое соединение сохраняется для повторного использования, задается директивой *keepalive_timeout* в блоке *upstream*, соединения которого кэшируются директивой

keepalive. Соединения с FastCGI-серверами кэшируются только при включенной директиве *fastcgi_keep_conn*.

fastcgi_connection_drop

Синтаксис	<code>fastcgi_connection_drop время on off;</code>
По умолчанию	<code>fastcgi_connection_drop off;</code>
Контекст	http, server, location

Настраивает завершение всех активных соединений с проксируемым сервером, если он был удален из группы или помечен как постоянно недоступный в результате процесса reresolve или команды API DELETE. Неактивные соединения в кэше не затрагиваются.

Соединение завершается, когда обрабатывается следующее событие чтения или записи для клиента или проксируемого сервера.

Установка *времени* включает таймаут до завершения соединения; при выборе значения **on** соединения завершаются немедленно.

fastcgi_force_ranges

Синтаксис	<code>fastcgi_force_ranges on off;</code>
По умолчанию	<code>fastcgi_force_ranges off;</code>
Контекст	http, server, location

Включает поддержку диапазонов запрашиваемых байт (byte-range) для кэшированных и некешированных ответов FastCGI-сервера вне зависимости от наличия поля **Accept-Ranges** в заголовках этих ответов.

fastcgi_hide_header

Синтаксис	<code>fastcgi_hide_header поле;</code>
По умолчанию	—
Контекст	http, server, location

По умолчанию Angie не передает клиенту поля заголовка **Status** и **X-Accel-...** из ответа FastCGI-сервера. Директива *fastcgi_hide_header* задает дополнительные поля, которые не будут передаваться. Если же передачу полей нужно разрешить, можно воспользоваться директивой *fastcgi_pass_header*.

fastcgi_ignore_client_abort

Синтаксис	<code>fastcgi_ignore_client_abort on off;</code>
По умолчанию	<code>fastcgi_ignore_client_abort off;</code>
Контекст	http, server, location

Определяет, закрывать ли соединение с FastCGI-сервером в случае, если клиент закрыл соединение, не дождавшись ответа.

fastcgi_ignore_headers

Синтаксис	<code>fastcgi_ignore_headers поле;</code>
По умолчанию	—
Контекст	http, server, location

Запрещает обработку некоторых полей заголовка из ответа FastCGI-сервера. В директиве можно указать поля X-Accel-Redirect, X-Accel-Expires, X-Accel-Limit-Rate, X-Accel-Buffering, X-Accel-Charset, Expires, Cache-Control, Set-Cookie и Vary.

Если не запрещено, обработка этих полей заголовка заключается в следующем:

- X-Accel-Expires, Expires, Cache-Control, Set-Cookie и Vary задают параметры кэширования ответа;
- X-Accel-Redirect производит внутреннее перенаправление на указанный URI;
- X-Accel-Limit-Rate задает ограничение скорости передачи ответа клиенту;
- X-Accel-Buffering включает или выключает буферизацию ответа;
- X-Accel-Charset задает желаемую кодировку ответа.

fastcgi_index

Синтаксис	<code>fastcgi_index имя;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт имя файла, который при создании переменной `$fastcgi_script_name` будет добавляться после URI, если URI заканчивается косой чертой. Например, при таких настройках

```
fastcgi_index index.php;
fastcgi_param SCRIPT_FILENAME /home/www/scripts/php$fastcgi_script_name;
```

и запросе `/page.php` параметр `SCRIPT_FILENAME` будет равен `/home/www/scripts/php/page.php`,

а при запросе `/` - `/home/www/scripts/php/index.php`.

fastcgi_intercept_errors

Синтаксис	<code>fastcgi_intercept_errors on off;</code>
По умолчанию	<code>fastcgi_intercept_errors off;</code>
Контекст	http, server, location

Определяет, передавать ли клиенту ответы FastCGI-сервера с кодом больше либо равным 300, или же перехватывать их и перенаправлять на обработку Angie с помощью директивы `error_page`.

fastcgi_interface

Синтаксис	<code>fastcgi_interface имя;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт сетевой интерфейс, который будет использоваться в исходящих соединениях с FastCGI-сервером; сокет привязывается к нему параметром сокета `SO_BINDTODEVICE`.

fastcgi_keep_conn

Синтаксис	<code>fastcgi_keep_conn on off;</code>
По умолчанию	<code>fastcgi_keep_conn off;</code>
Контекст	http, server, location

По умолчанию FastCGI-сервер будет закрывать соединение сразу же после отправки ответа. При установке значения `on` Angie указывает FastCGI-серверу оставлять соединения открытыми. Это в частности требуется для функционирования постоянных соединений с FastCGI-серверами.

fastcgi_limit_rate

Синтаксис	<code>fastcgi_limit_rate скорость;</code>
По умолчанию	<code>fastcgi_limit_rate 0;</code>
Контекст	http, server, location

Ограничивает скорость чтения ответа от FastCGI-сервера. *Скорость* задается в байтах в секунду; можно использовать переменные.

0	отключает ограничение скорости
---	--------------------------------

Примечание

Ограничение устанавливается на запрос, поэтому, если Angie одновременно откроет два соединения к FastCGI-серверу, суммарная скорость будет вдвое выше заданного ограничения. Ограничение работает только в случае, если включена буферизация ответов FastCGI-сервера.

fastcgi_max_temp_file_size

Синтаксис	<code>fastcgi_max_temp_file_size размер;</code>
По умолчанию	<code>fastcgi_max_temp_file_size 1024m;</code>
Контекст	http, server, location

Если включена буферизация ответов FastCGI-сервера, и ответ не помещается целиком в буферы, заданные директивами `fastcgi_buffer_size` и `fastcgi_buffers`, часть ответа может быть записана во временный файл. Эта директива задает максимальный размер временного файла. Размер данных, сбрасываемых во временный файл за один раз, задается директивой `fastcgi_temp_file_write_size`.

0	отключает возможность буферизации ответов во временные файлы
---	--

Примечание

Данное ограничение не распространяется на ответы, которые будут кэшированы или сохранены на диске.

fastcgi_next_upstream

Синтаксис	<code>fastcgi_next_upstream error timeout invalid_header http_500 http_503 http_403 http_404 http_429 non_idempotent off ...;</code>
По умолчанию	<code>fastcgi_next_upstream error timeout;</code>
Контекст	http, server, location

Определяет, в каких случаях запрос будет передан следующему серверу:

error	произошла ошибка соединения с сервером, передачи ему запроса или чтения заголовка ответа сервера;
timeout	произошел таймаут во время соединения с сервером, передачи ему запроса или чтения заголовка ответа сервера;
invalid_header	сервер вернул пустой или неверный ответ;
http_500	сервер вернул ответ с кодом 500;
http_503	сервер вернул ответ с кодом 503;
http_403	сервер вернул ответ с кодом 403;
http_404	сервер вернул ответ с кодом 404;
http_429	сервер вернул ответ с кодом 429;
non_idempotent	обычно запросы с неидемпотентным методом (<i>POST</i> , <i>LOCK</i> , <i>PATCH</i>) не передаются на другой сервер, если запрос серверу группы уже был отправлен; включение параметра явно разрешает повторять подобные запросы;
off	запрещает передачу запроса следующему серверу.

Примечание

Необходимо понимать, что передача запроса следующему серверу возможна только при условии, что клиенту еще ничего не передавалось. То есть, если ошибка или таймаут возникли в середине передачи ответа, то исправить это уже невозможно.

Директива также определяет, что считается неудачной попыткой работы с сервером.

error	всегда считаются неудачными попытками, даже если они не указаны в директиве
timeout	
invalid_header	
http_500	считаются неудачными попытками, только если они указаны в директиве
http_503	
http_429	
http_403	никогда не считаются неудачными попытками
http_404	

Передача запроса следующему серверу может быть ограничена по количеству попыток и по времени.

fastcgi_next_upstream_timeout

Синтаксис	<code>fastcgi_next_upstream_timeout</code> время;
По умолчанию	<code>fastcgi_next_upstream_timeout 0;</code>
Контекст	http, server, location

Ограничивает время, в течение которого возможна передача запроса следующему серверу.

0	отключает это ограничение
---	---------------------------

fastcgi_next_upstream_tries

Синтаксис	<code>fastcgi_next_upstream_tries</code> <i>число</i> ;
По умолчанию	<code>fastcgi_next_upstream_tries 0;</code>
Контекст	http, server, location

Ограничивает число допустимых попыток для передачи запроса следующему серверу.

0	отключает это ограничение
---	---------------------------

fastcgi_no_cache

Синтаксис	<code>fastcgi_no_cache</code> <i>строка</i> ...;
По умолчанию	—
Контекст	http, server, location

Задаёт условия, при которых ответ не будет сохраняться в кэш. Если значение хотя бы одного из строковых параметров непустое и не равно "0", то ответ не будет сохранен:

```
fastcgi_no_cache $cookie_nocache $arg_nocache$arg_comment;
fastcgi_no_cache $http_pragma $http_authorization;
```

В значении параметров можно использовать переменные.

Можно использовать совместно с директивой `fastcgi_cache_bypass`.

fastcgi_param

Синтаксис	<code>fastcgi_param</code> <i>параметр значение</i> [if_not_empty];
По умолчанию	—
Контекст	http, server, location

Задаёт параметр, который будет передаваться FastCGI-серверу. В качестве значения можно использовать переменные. Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы *fastcgi_param*.

i Примечание

В стандартных файлах *fastcgi.conf* и *fastcgi_params*, поставляемых с Angie, параметр *REQUEST_METHOD* задаётся как *\$upstream_request_method*. Это позволяет при конвертации *HEAD* в *GET* при кэшировании использовать корректный метод для запроса к upstream.

Ниже приведен пример минимально необходимых параметров для PHP:

```
fastcgi_param SCRIPT_FILENAME /home/www/scripts/php$fastcgi_script_name;
fastcgi_param QUERY_STRING $query_string;
```

Параметр *SCRIPT_FILENAME* используется в PHP для определения имени скрипта, а в параметре *QUERY_STRING* передаются параметры запроса.

Если скрипты обрабатывают запросы *POST*, то нужны еще три параметра:

```
fastcgi_param REQUEST_METHOD $request_method;
fastcgi_param CONTENT_TYPE $content_type;
fastcgi_param CONTENT_LENGTH $content_length;
```

Если PHP был собран с параметром конфигурации *--enable-force-cgi-redirect*, то нужно передавать параметр *REDIRECT_STATUS* со значением "200":

```
fastcgi_param REDIRECT_STATUS 200;
```

Если директива указана с *if_not_empty*, такой параметр с пустым значением передаваться на сервер не будет:

```
fastcgi_param HTTPS $https if_not_empty;
```

fastcgi_pass

Синтаксис	<code>fastcgi_pass адрес;</code>
По умолчанию	—
Контекст	location, if в location

Задаёт адрес FastCGI-сервера. Адрес может быть указан в виде доменного имени или IP-адреса, и порта:

```
fastcgi_pass localhost:9000;
```

или в виде пути UNIX-сокета:

```
fastcgi_pass unix:/tmp/fastcgi.socket;
```

Если доменному имени соответствует несколько адресов, то все они будут использоваться по очереди (round-robin). И, кроме того, адрес может быть группой серверов.

В значении параметра можно использовать переменные. В этом случае, если адрес указан в виде доменного имени, имя ищется среди описанных групп серверов и если не найдено, то определяется с помощью *resolver*'а.

Примечание

Если `fastcgi_pass` стоит в `location` с косой чертой в конце префикса (например, `location /name/`), и при этом в директиве `auto_redirect` указано `default`, запросы без косой черты в конце будут перенаправляться (`/name -> /name/`).

`fastcgi_pass_header`

Синтаксис	<code>fastcgi_pass_header поле;</code>
По умолчанию	—
Контекст	http, server, location

Разрешает передавать от FastCGI-сервера клиенту запрещенные для передачи поля заголовка.

`fastcgi_pass_request_body`

Синтаксис	<code>fastcgi_pass_request_body on off;</code>
По умолчанию	—
Контекст	http, server, location

Позволяет запретить передачу исходного тела запроса на FastCGI-сервер. См. также директиву `fastcgi_pass_request_headers`.

`fastcgi_pass_request_headers`

Синтаксис	<code>fastcgi_pass_request_headers on off;</code>
По умолчанию	—
Контекст	http, server, location

Позволяет запретить передачу полей заголовка исходного запроса на FastCGI-сервер. См. также директиву `fastcgi_pass_request_body`.

`fastcgi_read_timeout`

Синтаксис	<code>fastcgi_read_timeout время;</code>
По умолчанию	<code>fastcgi_read_timeout 60s;</code>
Контекст	http, server, location

Задаёт таймаут при чтении ответа FastCGI-сервера. Таймаут устанавливается не на всю передачу ответа, а только между двумя операциями чтения. Если по истечении этого времени FastCGI-сервер ничего не передаст, соединение закрывается.

fastcgi_request_buffering

Синтаксис	<code>fastcgi_request_buffering on off;</code>
По умолчанию	<code>fastcgi_request_buffering on;</code>
Контекст	<code>http, server, location</code>

Разрешает или запрещает использовать буферизацию тела запроса клиента.

on	тело запроса полностью читается от клиента перед отправкой запроса на FastCGI-сервер.
off	тело запроса отправляется на FastCGI-сервер сразу же по мере его поступления. В этом случае запрос не может быть передан следующему серверу, если Angie уже начал отправку тела запроса.

fastcgi_send_lowat

Синтаксис	<code>fastcgi_send_lowat размер;</code>
По умолчанию	<code>fastcgi_send_lowat 0;</code>
Контекст	<code>http, server, location</code>

При установке директивы в ненулевое значение Angie будет пытаться минимизировать число операций отправки на исходящих соединениях с FastCGI-сервером либо при помощи флага `NOTE_LOWAT` метода `kqueue`, либо при помощи параметра сокета `SO_SNDLOWAT`, с указанным размером.

Примечание

Эта директива игнорируется на Linux, Solaris и Windows.

fastcgi_send_timeout

Синтаксис	<code>fastcgi_send_timeout время;</code>
По умолчанию	<code>fastcgi_send_timeout 60s;</code>
Контекст	<code>http, server, location</code>

Задаёт таймаут при передаче запроса FastCGI-серверу. Таймаут устанавливается не на всю передачу запроса, а только между двумя операциями записи. Если по истечении этого времени FastCGI-сервер не примет новых данных, соединение закрывается.

fastcgi_socket_keepalive

Синтаксис	<code>fastcgi_socket_keepalive on off;</code>
По умолчанию	<code>fastcgi_socket_keepalive off;</code>
Контекст	<code>http, server, location</code>

Конфигурирует поведение "TCP keepalive" для исходящих соединений к FastCGI-серверу.

""	По умолчанию для сокета действуют настройки операционной системы.
on	для сокета включается параметр <i>SO_KEEPALIVE</i>

fastcgi_split_path_info

Синтаксис	<code>fastcgi_split_path_info <i>regex</i>;</code>
По умолчанию	—
Контекст	location

Задаёт регулярное выражение, выделяющее значение для переменной *\$fastcgi_path_info*. Регулярное выражение должно иметь две группы захвата, из которых первая становится значением переменной *\$fastcgi_script_name*, а вторая — значением переменной *\$fastcgi_path_info*. Например, при таких настройках

```
location ~ ^(\.+\.php)(.*)$ {
    fastcgi_split_path_info      ^(\.+\.php)(.*)$;
    fastcgi_param SCRIPT_FILENAME /path/to/php$fastcgi_script_name;
    fastcgi_param PATH_INFO      $fastcgi_path_info;
```

и запросе `/show.php/article/0001` параметр `SCRIPT_FILENAME` будет равен `/path/to/php/show.php`,

а параметр `PATH_INFO` - `/article/0001`.

fastcgi_store

Синтаксис	<code>fastcgi_store on off <i>строка</i>;</code>
По умолчанию	<code>fastcgi_store off;</code>
Контекст	http, server, location

Разрешает сохранение на диск файлов.

on	сохраняет файлы в соответствии с путями, указанными в директивах <i>alias</i> или <i>root</i>
off	запрещает сохранение файлов

Кроме того, имя файла можно задать явно с помощью строки с переменными:

```
fastcgi_store /data/www$original_uri;
```

Время изменения файлов выставляется согласно полученному полю **Last-Modified** в заголовке ответа. Ответ сначала записывается во временный файл, а потом этот файл переименовывается. Временный файл и постоянное место хранения ответа могут располагаться на разных файловых системах. Однако нужно учитывать, что в этом случае вместо дешевой операции переименовывания в пределах одной файловой системы файл копируется с одной файловой системы на другую. Поэтому лучше, если сохраняемые файлы будут находиться на той же файловой системе, что и каталог с временными файлами, задаваемый директивой *fastcgi_temp_path* для данного *location*.

Директиву можно использовать для создания локальных копий статических неизменяемых файлов, например:

```
location /images/ {
    root                /data/www;
    error_page          404 = /fetch$uri;
}

location /fetch/ {
    internal;

    fastcgi_pass        backend:9000;
    ...

    fastcgi_store        on;
    fastcgi_store_access user:rw group:rw all:r;
    fastcgi_temp_path    /data/temp;

    alias                /data/www/;
}
```

fastcgi_store_access

Синтаксис	<code>fastcgi_store_access пользователи:права ...;</code>
По умолчанию	<code>fastcgi_store_access user:rw;</code>
Контекст	http, server, location

Задаёт права доступа для создаваемых файлов и каталогов, например,

```
fastcgi_store_access user:rw group:rw all:r;
```

Если заданы какие-либо права для group или all, то права для user указывать необязательно:

```
fastcgi_store_access group:rw all:r;
```

fastcgi_tcp_congestion

Синтаксис	<code>fastcgi_tcp_congestion алгоритм;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт алгоритм управления перегрузкой TCP (параметр сокета TCP_CONGESTION) для исходящих соединений с FastCGI-сервером; к соединениям через UNIX-сокеты параметр не применяется.

Имя алгоритма не проверяется на этапе конфигурации: если алгоритм неизвестен или недоступен, при установке исходящего соединения в лог записывается сообщение уровня **alert**, а для соединения действует системный алгоритм по умолчанию.

fastcgi_temp_file_write_size

Синтаксис	<code>fastcgi_temp_file_write_size размер;</code>
По умолчанию	<code>fastcgi_temp_file_write_size 8k 16k;</code>
Контекст	http, server, location

Ограничивает размер данных, сбрасываемых во временный файл за один раз, при включенной буферизации ответов FastCGI-сервера во временные файлы. По умолчанию размер ограничен двумя буферами, заданными директивами *fastcgi_buffer_size* и *fastcgi_buffers*. Максимальный размер временного файла задается директивой *fastcgi_max_temp_file_size*.

fastcgi_temp_path

Синтаксис	<code>fastcgi_temp_path <i>путь</i> [<i>уровень1</i> [<i>уровень2</i> [<i>уровень3</i>]]]`;</code>
По умолчанию	<code>fastcgi_temp_path fastcgi_temp;</code> (путь зависит от параметра сборки <code>--http-fastcgi-temp-path</code>)
Контекст	http, server, location

Задаёт имя каталога для хранения временных файлов с данными, полученными от FastCGI-серверов. В каталоге может использоваться иерархия подкаталогов до трех уровней. Например, при такой конфигурации

```
fastcgi_temp_path /spool/angie/fastcgi_temp 1 2;
```

временный файл будет следующего вида:

```
/spool/angie/fastcgi_temp/7/45/00000123457
```

См. также параметр *use_temp_path* директивы *fastcgi_cache_path*.

Параметры, передаваемые FastCGI-серверу

Поля заголовка HTTP-запроса передаются FastCGI-серверу в виде параметров. В приложениях и скриптах, запущенных в виде FastCGI-сервера, эти параметры обычно доступны в виде переменных среды. Например, поле заголовка **User-Agent** передается как параметр `HTTP_USER_AGENT`. Кроме полей заголовка HTTP-запроса можно передавать произвольные параметры с помощью директивы *fastcgi_param*.

Встроенные переменные

В модуле *http_fastcgi* есть встроенные переменные, которые можно использовать для формирования параметров с помощью директивы *fastcgi_param*:

`$fastcgi_path_info`

значение второй группы захвата, задаваемой директивой *fastcgi_split_path_info*. Эту переменную можно использовать для задания параметра `PATH_INFO`.

`$fastcgi_script_name`

URI запроса или же, если URI заканчивается косой чертой, то URI запроса, дополненное именем индексного файла, задаваемого директивой *fastcgi_index*. Эту переменную можно использовать для задания параметров `SCRIPT_FILENAME` и `PATH_TRANSLATED`, используемых, в частности, для определения имени скрипта в PHP. Например, для запроса `/info/` и при использовании директив

```
fastcgi_index index.php;
fastcgi_param SCRIPT_FILENAME /home/www/scripts/php$fastcgi_script_name;
```

параметр `SCRIPT_FILENAME` будет равен `/home/www/scripts/php/info/index.php`.

При использовании директивы *fastcgi_split_path_info* переменная *\$fastcgi_script_name* равна значению первой группы захвата, задаваемой этой директивой.

FLV

Обеспечивает серверную поддержку псевдо-стриминга для файлов Flash Video (FLV).

Он специальным образом обрабатывает запросы с аргументом **start** в строке запроса, посылая в ответ содержимое файла с запрошенного смещения в байтах, добавив перед ним FLV-заголовок.

Пример конфигурации

```
location ~ /\.flv$ {
    flv;
}
```

Директивы

flv

Синтаксис	flv;
По умолчанию	—
Контекст	location

Включает в содержащем location обработку этим модулем.

Geo

Создает переменные, значения которых зависят от IP-адреса клиента.

Пример конфигурации

```
geo $geo {
    default 0;

    127.0.0.1 2;
    192.168.1.0/24 1;
    10.1.0.0/16 1;

    ::1 2;
    2001:0db8::/32 1;
}
```

Директивы

geo

Синтаксис	geo [<i>\$адрес</i>] <i>\$переменная</i> { ... }
По умолчанию	—
Контекст	http

Описывает для указанной переменной зависимость значения от IP-адреса клиента. По умолчанию адрес берется из переменной *\$remote_addr*, но его также можно получить из другой переменной, например:

```
geo $arg_remote_addr $geo {
    ...;
}
```

Примечание

Поскольку переменные вычисляются только в момент использования, само по себе наличие даже большого числа объявлений переменных `geo` не влечет за собой никаких дополнительных расходов на обработку запросов.

Если значение переменной не представляет собой правильный IP-адрес, то используется адрес "255.255.255.255".

Адреса задаются либо префиксами в формате CIDR (включая одиночные адреса), либо в виде диапазонов.

Также поддерживаются следующие специальные параметры:

delete	удаляет описанную сеть
default	значение переменной, если адрес клиента не соответствует ни одному из заданных адресов. При задании адресов в формате CIDR вместо <i>default</i> можно использовать "0.0.0.0/0" и "::/0". Если параметр <i>default</i> не указан, значением по умолчанию будет пустая строка.
include	включает файл с адресами и значениями. Имя файла может содержать глоб-шаблоны, в этом случае подключаются все совпадающие файлы. Включений может быть несколько.
proxy	задает доверенные адреса, при запросе с которых будет использоваться адрес в переданном поле заголовка запроса X-Forwarded-For . В отличие от обычных адресов, доверенные адреса проверяются последовательно.
proxy_recursive	включает рекурсивный поиск адреса. При выключенном рекурсивном поиске вместо исходного адреса клиента, совпадающего с одним из доверенных адресов, будет использоваться последний адрес, переданный в X-Forwarded-For . При включенном рекурсивном поиске вместо исходного адреса клиента, совпадающего с одним из доверенных адресов, будет использоваться последний не доверенный адрес, переданный в X-Forwarded-For .
ranges	указывает, что адреса задаются в виде диапазонов. Этот параметр должен быть первым. Для ускорения загрузки гео-базы нужно располагать адреса в порядке возрастания.
volatile	указывает, что переменная не кэшируется.

Пример:

```
geo $country {
    default      ZZ;
    include      conf/geo.conf;
    delete      127.0.0.0/16;
    proxy        192.168.100.0/24;
    proxy        2001:0db8::/32;

    127.0.0.0/24  US;
    127.0.0.1/32  RU;
    10.1.0.0/16   RU;
    192.168.1.0/24 UK;
}
```

В файле `conf/geo.conf` могут быть такие строки:

```
10.2.0.0/16    RU;
192.168.2.0/24 RU;
```

В качестве значения выбирается максимальное совпадение, например, для адреса 127.0.0.1 будет выбрано значение RU, а не US.

Пример описания диапазонов:

```
geo $country {
    ranges;
    default                ZZ;
    127.0.0.0-127.0.0.0    US;
    127.0.0.1-127.0.0.1    RU;
    127.0.0.2-127.0.0.255  US;
    10.1.0.0-10.1.255.255  RU;
    192.168.1.0-192.168.1.255 UK;
}
```

gRPC

Позволяет передавать запросы gRPC-серверу.

Примечание

Для работы этого модуля необходим модуль HTTP/2.

Пример конфигурации

```
server {
    listen 9000;

    http2 on;

    location / {
        grpc_pass 127.0.0.1:9000;
    }
}
```

Директивы

grpc_bind

Синтаксис	<code>grpc_bind <i>адрес</i> [transparent] off;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт локальный IP-адрес с необязательным портом, который будет использоваться в исходящих соединениях с gRPC-сервером. В значении параметра можно использовать переменные. Специальное значение `off` отменяет действие унаследованной с предыдущего уровня конфигурации директивы `grpc_bind`, позволяя системе самостоятельно выбирать локальный IP-адрес и порт.

По умолчанию это значение вычисляется один раз при инициализации запроса, до выбора gRPC-сервера. В Angie PRO вычисление происходит после каждого выбора gRPC-сервера (в том числе

при повторных попытках), поэтому адрес привязки может зависеть от выбранного сервера; например, с помощью переменной `$upstream_current_addr`. Если вычисленное значение пусто, привязка не выполняется.

Параметр `transparent` позволяет задать нелокальный IP-адрес, который будет использоваться в исходящих соединениях с gRPC-сервером, например, реальный IP-адрес клиента:

```
grpc_bind $remote_addr transparent;
```

Для работы параметра обычно требуется запустить рабочие процессы Angie с привилегиями суперпользователя. В Linux это не требуется, так как если указан параметр `transparent`, то рабочие процессы наследуют `capability CAP_NET_RAW` из главного процесса.

Примечание

Необходимо настроить таблицу маршрутизации ядра для перехвата сетевого трафика с gRPC-сервера.

`grpc_buffer_size`

Синтаксис	<code>grpc_buffer_size</code> размер;
По умолчанию	<code>grpc_buffer_size</code> 4k 8k;
Контекст	http, server, location

Задаёт размер буфера, в который будет читаться первая часть ответа, получаемого от gRPC-сервера. Ответ синхронно передается клиенту сразу же по мере его поступления.

`grpc_connect_timeout`

Синтаксис	<code>grpc_connect_timeout</code> время;
По умолчанию	<code>grpc_connect_timeout</code> 60s;
Контекст	http, server, location

Задаёт таймаут для установления соединения с gRPC-сервером. Необходимо иметь в виду, что этот таймаут обычно не может превышать 75 секунд.

Примечание

Этот таймаут не управляет неактивными соединениями: время, в течение которого уже открытое соединение сохраняется для повторного использования, задается директивой `keepalive_timeout` в блоке `upstream`, соединения которого кэшируются директивой `keepalive`.

`grpc_connection_drop`

Синтаксис	<code>grpc_connection_drop</code> время on off;
По умолчанию	<code>grpc_connection_drop</code> off;
Контекст	http, server, location

Настраивает завершение всех активных соединений с проксируемым сервером, если он был удален из группы или помечен как постоянно недоступный в результате процесса `eresolve` или команды `API DELETE`. Неактивные соединения в кэше не затрагиваются.

Соединение завершается, когда обрабатывается следующее событие чтения или записи для клиента или проксируемого сервера.

Установка *времени* включает таймаут до завершения соединения; при выборе значения `on` соединения завершаются немедленно.

`grpc_hide_header`

Синтаксис	<code>grpc_hide_header поле;</code>
По умолчанию	—
Контекст	<code>http, server, location</code>

По умолчанию Angie не передает клиенту поля заголовка `Date`, `Server` и `X-Accel-...` из ответа gRPC-сервера. Директива `grpc_hide_header` задает дополнительные поля, которые не будут передаваться. Если же передачу полей нужно разрешить, можно воспользоваться директивой `grpc_pass_header`.

`grpc_ignore_headers`

Синтаксис	<code>grpc_ignore_headers поле ...;</code>
По умолчанию	—
Контекст	<code>http, server, location</code>

Запрещает обработку некоторых полей заголовка из ответа gRPC-сервера. В директиве можно указать поля `X-Accel-Redirect` и `X-Accel-Charset`.

Если не запрещено, обработка этих полей заголовка заключается в следующем:

- `X-Accel-Redirect` производит внутреннее перенаправление на указанный URI;
- `X-Accel-Charset` задает желаемую кодировку ответа.

`grpc_intercept_errors`

Синтаксис	<code>grpc_intercept_errors on off;</code>
По умолчанию	<code>grpc_intercept_errors off;</code>
Контекст	<code>http, server, location</code>

Определяет, передавать ли клиенту ответы gRPC-сервера с кодом больше либо равным 300, или же перехватывать их и перенаправлять на обработку Angie с помощью директивы `error_page`.

`grpc_interface`

Синтаксис	<code>grpc_interface имя;</code>
По умолчанию	—
Контекст	<code>http, server, location</code>

Задаёт сетевой интерфейс, который будет использоваться в исходящих соединениях с gRPC-сервером; сокет привязывается к нему параметром сокета `SO_BINDTODEVICE`.

grpc_next_upstream

Синтаксис	<code>grpc_next_upstream error timeout invalid_header http_500 http_502 http_503 http_504 http_403 http_404 http_429 non_idempotent off ...;</code>
По умолчанию	<code>grpc_next_upstream error timeout;</code>
Контекст	<code>http, server, location</code>

Определяет, в каких случаях запрос будет передан следующему в группе upstream серверу:

<code>error</code>	произошла ошибка соединения с сервером, передачи ему запроса или чтения заголовка ответа сервера;
<code>timeout</code>	произошел таймаут во время соединения с сервером, передачи ему запроса или чтения заголовка ответа сервера;
<code>invalid_header</code>	сервер вернул пустой или неверный ответ;
<code>http_500</code>	сервер вернул ответ с кодом 500;
<code>http_502</code>	сервер вернул ответ с кодом 502;
<code>http_503</code>	сервер вернул ответ с кодом 503;
<code>http_504</code>	сервер вернул ответ с кодом 504;
<code>http_403</code>	сервер вернул ответ с кодом 403;
<code>http_404</code>	сервер вернул ответ с кодом 404;
<code>http_429</code>	сервер вернул ответ с кодом 429;
<code>non_idempotent</code>	обычно запросы с неидемпотентным методом (<i>POST, LOCK, PATCH</i>) не передаются на другой сервер, если запрос серверу группы уже был отправлен; включение параметра явно разрешает повторять подобные запросы;
<code>off</code>	запрещает передачу запроса следующему серверу.

Примечание

Необходимо понимать, что передача запроса следующему серверу возможна только при условии, что клиенту еще ничего не передавалось. То есть, если ошибка или таймаут возникли в середине передачи ответа клиенту, то действие директивы на такой запрос не распространяется.

Директива также определяет, что считается неудачной попыткой работы с сервером.

<code>error, timeout, invalid_header</code>	всегда считаются неудачными попытками, даже если они не указаны в директиве
<code>http_500, http_502, http_503, http_504, http_429</code>	считаются неудачными попытками, только если они указаны в директиве
<code>http_403, http_404</code>	никогда не считаются неудачными попытками

Передача запроса следующему серверу может быть ограничена по количеству попыток и по времени.

grpc_next_upstream_timeout

Синтаксис	grpc_next_upstream_timeout время;
По умолчанию	grpc_next_upstream_timeout 0;
Контекст	http, server, location

Ограничивает время, в течение которого возможна передача запроса следующему серверу.

0	отключает это ограничение
---	---------------------------

grpc_next_upstream_tries

Синтаксис	grpc_next_upstream_tries <i>число</i> ;
По умолчанию	grpc_next_upstream_tries 0;
Контекст	http, server, location

Ограничивает число допустимых попыток для передачи запроса следующему серверу.

0	отключает это ограничение
---	---------------------------

grpc_pass

Синтаксис	grpc_pass <i>адрес</i> ;
По умолчанию	—
Контекст	location, if в location

Задаёт адрес gRPC-сервера. Адрес может быть указан в виде доменного имени или IP-адреса, и порта:

```
grpc_pass localhost:9000;
```

или в виде пути UNIX-сокета:

```
grpc_pass unix:/tmp/grpc.socket;
```

Также может использоваться схема `grpc://`:

```
grpc_pass grpc://127.0.0.1:9000;
```

Для использования gRPC по SSL необходимо использовать схему `grpcs://`:

```
grpc_pass grpcs://127.0.0.1:443;
```

Если доменному имени соответствует несколько адресов, то все они будут использоваться по очереди (round-robin). Кроме того, в качестве адреса можно указать группу серверов.

В значении параметра можно использовать переменные. В этом случае, если адрес указан в виде доменного имени, имя ищется среди описанных групп серверов и если не найдено, то определяется с помощью *resolver*'а.

Примечание

Если `grpc_pass` стоит в `location` с косой чертой в конце префикса (например, `location /name/`), и при этом в директиве `auto_redirect` указано `default`, запросы без косой черты в конце будут перенаправляться (`/name -> /name/`).

grpc_pass_header

Синтаксис	<code>grpc_pass_header поле;</code>
По умолчанию	—
Контекст	<code>http, server, location</code>

Разрешает передавать от gRPC-сервера клиенту запрещенные для передачи поля заголовка.

grpc_read_timeout

Синтаксис	<code>grpc_read_timeout время;</code>
По умолчанию	<code>grpc_read_timeout 60s;</code>
Контекст	<code>http, server, location</code>

Задаёт таймаут при чтении ответа gRPC-сервера. Таймаут устанавливается не на всю передачу ответа, а только между двумя операциями чтения. Если по истечении этого времени gRPC-сервер ничего не передаст, соединение закрывается.

grpc_send_timeout

Синтаксис	<code>grpc_send_timeout время;</code>
По умолчанию	<code>grpc_send_timeout 60s;</code>
Контекст	<code>http, server, location</code>

Задаёт таймаут при передаче запроса gRPC-серверу. Таймаут устанавливается не на всю передачу запроса, а только между двумя операциями записи. Если по истечении этого времени gRPC-сервер не примет новых данных, соединение закрывается.

grpc_set_header

Синтаксис	<code>grpc_set_header поле значение;</code>
По умолчанию	<code>grpc_set_header Content-Length \$content_length;</code>
Контекст	<code>http, server, location</code>

Позволяет переопределять или добавлять поля заголовка запроса, передаваемые проксируемому серверу. В качестве значения можно использовать переменные. Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы `grpc_set_header`.

Если значение поля заголовка — пустая строка, то поле вообще не будет передаваться gRPC-серверу:

```
grpc_set_header Accept-Encoding "";
```

grpc_socket_keepalive

Синтаксис	grpc_socket_keepalive on off;
По умолчанию	grpc_socket_keepalive off;
Контекст	http, server, location

Конфигурирует поведение "TCP keepalive" для исходящих соединений к проксируемому серверу.

""	По умолчанию для сокета действуют настройки операционной системы.
on	для сокета включается параметр <i>SO_KEEPALIVE</i>

grpc_ssl_certificate

Синтаксис	grpc_ssl_certificate <i>файл</i> ;
По умолчанию	—
Контекст	http, server, location

Задаёт файл с сертификатом в формате PEM для аутентификации на gRPC SSL-сервере. В имени файла можно использовать переменные.

grpc_ssl_certificate_cache

Синтаксис	grpc_ssl_certificate_cache off; grpc_ssl_certificate_cache max= <i>N</i> [inactive= <i>time</i>] [valid= <i>time</i>];
Значение по умолчанию	grpc_ssl_certificate_cache off;
Контекст	http, server, location

Определяет кэш для хранения SSL-сертификатов и секретных ключей, заданных через переменные.

Директива поддерживает следующие параметры:

- **max** — устанавливает максимальное количество элементов в кэше. При переполнении кэша удаляются наименее недавно использованные (LRU) элементы.
- **inactive** — определяет время, после которого элемент будет удален, если к нему не было обращений в течение этого времени, даже если он еще находится в пределах своего периода **valid**; каждое обращение сбрасывает этот таймер. Значение по умолчанию — 10 секунд.
- **valid** — определяет время, в течение которого элемент кэша считается действительным и может использоваться повторно. Значение по умолчанию — 60 секунд. По истечении этого времени сертификаты перезагружаются или проходят повторную проверку.
- **off** — отключает кэш.

Пример:

```
grpc_ssl_certificate      $grpc_ssl_server_name.crt;  
grpc_ssl_certificate_key  $grpc_ssl_server_name.key;  
grpc_ssl_certificate_cache max=1000 inactive=20s valid=1m;
```

grpc_ssl_certificate_key

Синтаксис	grpc_ssl_certificate_key <i>файл</i> ;
По умолчанию	—
Контекст	http, server, location

Задаёт файл с секретным ключом в формате PEM для аутентификации на gRPC SSL-сервере.

Вместо файла можно указать значение "engine:*имя*:id", которое загружает ключ с указанным *id* из OpenSSL engine с заданным именем.

Вместо файла можно указать значение "store:*scheme*:id", которое используется для загрузки ключа с указанным *id* и URI-схемой *scheme*, зарегистрированной в OpenSSL provider, например `pkcs11`.

В имени файла можно использовать переменные.

grpc_ssl_ciphers

Синтаксис	grpc_ssl_ciphers <i>шифры</i> ;
По умолчанию	grpc_ssl_ciphers DEFAULT;
Контекст	http, server, location

Описывает разрешенные шифры для запросов к gRPC SSL-серверу. Шифры задаются в формате, поддерживаемом библиотекой OpenSSL.

Список шифров зависит от установленной версии OpenSSL. Полный список можно посмотреть с помощью команды `openssl ciphers`.

Предупреждение

Директива `grpc_ssl_ciphers` *не* настраивает шифры для TLS 1.3 при использовании OpenSSL. Для настройки шифров TLS 1.3 в OpenSSL используйте директиву `grpc_ssl_conf_command`, добавленную для расширенной конфигурации SSL.

- В LibreSSL шифры TLS 1.3 *можно* настраивать с помощью `grpc_ssl_ciphers`.
- В BoringSSL шифры TLS 1.3 настроить невозможно.

grpc_ssl_conf_command

Синтаксис	grpc_ssl_conf_command <i>имя значение</i> ;
По умолчанию	—
Контекст	http, server, location

Задаёт произвольные конфигурационные *команды* OpenSSL при установлении соединения с gRPC SSL-сервером.

Примечание

Директива поддерживается при использовании OpenSSL 1.0.2 и выше. Чтобы настроить шифры TLS 1.3 в OpenSSL, используйте команду `ciphersuites`.

На одном уровне может быть указано несколько директив `grpc_ssl_conf_command`. Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы `grpc_ssl_conf_command`.

Предупреждение

Следует учитывать, что изменение настроек OpenSSL напрямую может привести к неожиданному поведению.

`grpc_ssl_crl`

Синтаксис	<code>grpc_ssl_crl файл;</code>
По умолчанию	—
Контекст	http, server, location

Указывает файл с отозванными сертификатами (CRL) в формате PEM, используемыми при проверке сертификата gRPC SSL-сервера.

`grpc_ssl_name`

Синтаксис	<code>grpc_ssl_name имя;</code>
По умолчанию	<code>grpc_ssl_name `имя хоста из grpc_pass`;</code>
Контекст	http, server, location

Позволяет переопределить имя сервера, используемое при проверке сертификата gRPC SSL-сервера, а также для передачи его через SNI при установлении соединения с gRPC SSL-сервером.

Имя сервера также можно задать с помощью переменных.

По умолчанию используется имя хоста из `grpc_pass`.

`grpc_ssl_password_file`

Синтаксис	<code>grpc_ssl_password_file файл;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт файл с паролями от секретных ключей, где каждый пароль указан на отдельной строке. Пароли применяются по очереди в момент загрузки ключа.

`grpc_ssl_protocols`

Синтаксис	<code>grpc_ssl_protocols [SSLv2] [SSLv3] [TLSv1] [TLSv1.1] [TLSv1.2] [TLSv1.3];</code>
По умолчанию	<code>grpc_ssl_protocols TLSv1.2 TLSv1.3;</code>
Контекст	http, server, location

Разрешает указанные протоколы для запросов к gRPC SSL-серверу.

grpc_ssl_server_name

Синтаксис	grpc_ssl_server_name on off;
По умолчанию	grpc_ssl_server_name off;
Контекст	http, server, location

Разрешает или запрещает передачу имени сервера, заданного директивой *grpc_ssl_name*, через расширение *Server Name Indication* протокола TLS (SNI, RFC 6066) при установлении соединения с SSL-сервером gRPC.

grpc_ssl_session_reuse

Синтаксис	grpc_ssl_session_reuse on off;
По умолчанию	grpc_ssl_session_reuse on;
Контекст	http, server, location

Определяет, использовать ли повторно SSL-сессии при работе с gRPC-сервером. Если в логах появляются ошибки "*SSL3_GET_FINISHED:digest check failed*", то можно попробовать выключить повторное использование сессий.

grpc_ssl_trusted_certificate

Синтаксис	grpc_ssl_trusted_certificate <i>файл</i> ;
По умолчанию	—
Контекст	http, server, location

Задаёт файл с доверенными сертификатами СА в формате PEM, используемыми при проверке сертификата gRPC SSL-сервера.

grpc_ssl_verify

Синтаксис	grpc_ssl_verify on off;
По умолчанию	grpc_ssl_verify off;
Контекст	http, server, location

Разрешает или запрещает проверку сертификата gRPC SSL-сервера.

grpc_ssl_verify_depth

Синтаксис	grpc_ssl_verify_depth <i>число</i> ;
По умолчанию	grpc_ssl_verify_depth 1;
Контекст	http, server, location

Устанавливает глубину проверки в цепочке сертификатов gRPC SSL-сервера.

grpc_tcp_congestion

Синтаксис	grpc_tcp_congestion алгоритм;
По умолчанию	—
Контекст	http, server, location

Задаёт алгоритм управления перегрузкой TCP (параметр сокета TCP_CONGESTION) для исходящих соединений с gRPC-сервером; к соединениям через UNIX-сокеты параметр не применяется.

Имя алгоритма не проверяется на этапе конфигурации: если алгоритм неизвестен или недоступен, при установке исходящего соединения в лог записывается сообщение уровня **alert**, а для соединения действует системный алгоритм по умолчанию.

GunZIP

Фильтр, распаковывающий ответы с Content-Encoding: gzip для тех клиентов, которые не поддерживают метод сжатия "gzip". Модуль будет полезен, когда данные желательно хранить сжатыми для экономии места и сокращения затрат на ввод-вывод.

Пример конфигурации

```
location /storage/ {
    gunzip on;
    # ...
}
```

Директивы

gunzip

Синтаксис	gunzip on off;
По умолчанию	gunzip off;
Контекст	http, server, location

Разрешает или запрещает распаковку ответов, сжатых методом gzip, для тех клиентов, которые его не поддерживают. Если разрешено, то для определения, поддерживает ли клиент gzip, также учитываются следующие директивы: *gzip_http_version*, *gzip_proxied* и *gzip_disable*. См. также директиву *gzip_vary*.

gunzip_buffers

Синтаксис	gunzip_buffers число размер;
По умолчанию	gunzip_buffers 32 4k 16 8k;
Контекст	http, server, location

Задаёт число и размер буферов, в которые будет разжиматься ответ. По умолчанию размер одного буфера равен размеру страницы. В зависимости от платформы это или 4К, или 8К.

GZip

Фильтр, сжимающий ответ методом gzip, что позволяет уменьшить размер передаваемых данных в 2 и более раз.

Предупреждение

При использовании протокола SSL/TLS сжатые ответы могут быть подвержены атакам BREACH.

Пример конфигурации

```
gzip                on;
gzip_min_length    1000;
gzip_proxied       expired no-cache no-store private auth;
gzip_types         text/plain application/xml;
```

Для записи в лог достигнутого коэффициента сжатия можно использовать переменную `$gzip_ratio`.

Директивы

gzip

Синтаксис	<code>gzip on off;</code>
По умолчанию	<code>gzip off;</code>
Контекст	http, server, location, if в location

Разрешает или запрещает сжатие ответа методом gzip.

gzip_buffers

Синтаксис	<code>gzip_buffers <i>число</i> размер;</code>
По умолчанию	<code>gzip_buffers 32 4k 16 8k;</code>
Контекст	http, server, location

Задаёт число и размер буферов, в которые будет сжиматься ответ. По умолчанию размер одного буфера равен размеру страницы. В зависимости от платформы это или 4К, или 8К.

gzip_comp_level

Синтаксис	<code>gzip_comp_level <i>степень</i>;</code>
По умолчанию	<code>gzip_comp_level 1;</code>
Контекст	http, server, location

Устанавливает степень сжатия ответа методом gzip. Допустимые значения находятся в диапазоне от 1 до 9.

gzip_disable

Синтаксис	<code>gzip_disable <i>regex</i> ...;</code>
По умолчанию	—
Контекст	http, server, location

Запрещает сжатие ответа методом gzip для запросов с полями заголовка **User-Agent**, совпадающими с заданными регулярными выражениями.

Специальная маска **msie6** соответствует регулярному выражению "*MSIE [4-6].*", но работает быстрее. Из этой маски исключается "*MSIE 6.0; ... SV1*".

gzip_http_version

Синтаксис	<code>gzip_http_version 1.0 1.1;</code>
По умолчанию	<code>gzip_http_version 1.1;</code>
Контекст	http, server, location

Устанавливает минимальную HTTP-версию запроса, необходимую для сжатия ответа.

gzip_min_length

Синтаксис	<code>gzip_min_length <i>длина</i>;</code>
По умолчанию	<code>gzip_min_length 20;</code>
Контекст	http, server, location

Устанавливает минимальную длину ответа, который будет сжиматься методом gzip. Длина определяется только из поля **Content-Length** заголовка ответа.

gzip_proxied

Синтаксис	<code>gzip_proxied off expired no-cache no-store private no_last_modified no_etag auth any ...;</code>
По умолчанию	<code>gzip_proxied off;</code>
Контекст	http, server, location

Разрешает или запрещает сжатие ответа методом gzip для проксированных запросов в зависимости от запроса и ответа. То, что запрос проксированный, определяется на основании наличия поля **Via** в заголовке запроса. В директиве можно указать одновременно несколько параметров:

off	запрещает сжатие для всех проксированных запросов, игнорируя остальные параметры;
expired	разрешает сжатие, если в заголовке ответа есть поле Expires со значением, запрещающим кэширование;
no-cache	разрешает сжатие, если в заголовке ответа есть поле Cache-Control с параметром "no-cache";
no-store	разрешает сжатие, если в заголовке ответа есть поле Cache-Control с параметром "no-store";
private	разрешает сжатие, если в заголовке ответа есть поле Cache-Control с параметром "private";
no_last_modified	разрешает сжатие, если в заголовке ответа нет поля Last-Modified ;
no_etag	разрешает сжатие, если в заголовке ответа нет поля ETag ;
auth	разрешает сжатие, если в заголовке запроса есть поле Authorization ;
any	разрешает сжатие для всех проксированных запросов.

gzip_types

Синтаксис	<code>gzip_types mime-mun ...;</code>
По умолчанию	<code>gzip_types text/html;</code>
Контекст	http, server, location

Разрешает сжатие ответа методом gzip для указанных MIME-типов в дополнение к `text/html`. Специальное значение "*" соответствует любому MIME-типу. Ответы с типом `text/html` сжимаются всегда.

gzip_vary

Синтаксис	<code>gzip_vary on off;</code>
По умолчанию	<code>gzip_vary off;</code>
Контекст	http, server, location

Разрешает или запрещает выдавать в ответе поле заголовка "Vary: Accept-Encoding", если активны директивы `gzip`, `gzip_static` или `gunzip`.

Встроенные переменные

\$gzip_ratio

достигнутый коэффициент сжатия — отношение размера исходного ответа к размеру сжатого.

GZip Static

Позволяет отдавать вместо обычного файла предварительно сжатый файл с таким же именем и с расширением ".gz".

Пример конфигурации

```
gzip_static on;
gzip_proxied expired no-cache no-store private auth;
```

Директивы

gzip_static

Синтаксис	<code>gzip_static on off always;</code>
По умолчанию	<code>gzip_static off;</code>
Контекст	<code>http, server, location</code>

Разрешает (`on`) или запрещает (`off`) проверку готового сжатого файла. При использовании также учитываются директивы `gzip_http_version`, `gzip_proxied`, `gzip_disable` и `gzip_vary`.

Со значением `always` во всех случаях будет использоваться сжатый файл, без проверки поддержки на стороне клиента. Это полезно, если на диске все равно нет несжатых файлов, или используется модуль GunZIP.

Сжимать файлы можно с помощью программы `gzip` или совместимой с ней. Желательно, чтобы дата и время модификации исходного и сжатого файлов совпадали.

Headers

Позволяет выдавать поля заголовка `Expires` и `Cache-Control`, а также добавлять произвольные поля в заголовок ответа.

Пример конфигурации

```
expires 24h;
expires modified +24h;
expires @24h;
expires 0;
expires -1;
expires epoch;
expires $expires;
add_header Cache-Control private;
```

Директивы

add_header

Синтаксис	<code>add_header имя значение [always];</code>
По умолчанию	—
Контекст	<code>http, server, location, if в location</code>

Добавляет указанное поле в заголовок ответа при условии, что код ответа равен 200, 201 (1.3.10), 204, 206, 301, 302, 303, 304, 307 или 308. В значении параметра можно использовать переменные.

Директив `add_header` может быть несколько. Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы `add_header`.

Если указан параметр `always`, то поле заголовка будет добавлено независимо от кода ответа.

add_trailer

Синтаксис	<code>add_trailer имя значение [always];</code>
По умолчанию	—
Контекст	http, server, location, if в location

Добавляет указанное поле в конец ответа при условии, что код ответа равен 200, 201, 206, 301, 302, 303, 307 или 308. В значении можно использовать переменные.

Директив `add_trailer` может быть несколько. Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы `add_trailer`.

Если указан параметр `always`, то указанное поле будет добавлено независимо от кода ответа.

expires

Синтаксис	<code>expires [modified] время;</code> <code>expires epoch max off;</code>
По умолчанию	<code>expires off;</code>
Контекст	http, server, location, if в location

Разрешает или запрещает добавлять или менять поля `Expires` и `Cache-Control` в заголовке ответа при условии, что код ответа равен 200, 201, 204, 206, 301, 302, 303, 304, 307 или 308. В качестве параметра можно задать положительное или отрицательное время.

Время в поле `Expires` получается как сумма текущего времени и времени, заданного в директиве. Если используется параметр `modified`, то время получается как сумма времени модификации файла и времени, заданного в директиве.

Кроме того, с помощью префикса "@" можно задать время суток:

```
expires @15h30m;
```

Содержимое поля `Cache-Control` зависит от знака заданного времени:

- отрицательное время — "Cache-Control: no-cache".
- положительное или равное нулю время — "Cache-Control: max-age=`t`", где *t* это время в секундах, заданное в директиве.

epoch	задает время "Thu, 01 Jan 1970 00:00:01 GMT" (1 января 1970 00:00:01 GMT) для поля <code>Expires</code> и "no-cache" для поля <code>Cache-Control</code> .
max	задает время "Thu, 31 Dec 2037 23:55:55 GMT" (31 декабря 2037 23:55:55 GMT) для поля <code>Expires</code> и 10 лет для поля <code>Cache-Control</code> .
off	запрещает добавлять или менять поля <code>Expires</code> и <code>Cache-Control</code> в заголовке ответа.

В значении последнего параметра можно использовать переменные:

```
map $sent_http_content_type $expires {
    default      off;
    application/pdf 42d;
    ~image/       max;
}

expires $expires;
```

Index

Обслуживает запросы, оканчивающиеся косой чертой (/). Такие запросы также могут обслуживаться модулями AutoIndex и Random Index.

Пример конфигурации

```
location / {
    index index.$geo.html index.html;
}
```

Директивы

index

Синтаксис	<code>index <i>файл</i> ...;</code>
По умолчанию	<code>index index.html;</code>
Контекст	http, server, location

Определяет файлы, которые будут использоваться в качестве индекса. В имени файла можно использовать переменные. Наличие файлов проверяется в порядке их перечисления. В конце списка может стоять файл с абсолютным путем. Пример:

```
index index.$geo.html index.0.html /index.html;
```

Необходимо иметь в виду, что при использовании индексного файла делается внутреннее перенаправление и запрос может быть обработан уже в другом `location`. Например, в такой конфигурации:

```
location = / {
    index index.html;
}

location / {
#    ...
}
```

запрос "/" будет фактически обработан во втором `location` как `/index.html`.

Limit Conn

Позволяет ограничить число соединений по заданному ключу, в частности, число соединений с одного IP-адреса.

Учитываются не все соединения, а лишь те, в которых имеются запросы, обрабатываемые сервером, и заголовок запроса уже прочитан.

Пример конфигурации

```
http {
    limit_conn_zone $binary_remote_addr zone=addr:10m;

    ...

    server {
```

```
...

location /download/ {
    limit_conn addr 1;
}
```

Директивы

limit_conn

Синтаксис	limit_conn зона число;
По умолчанию	—
Контекст	http, server, location

Задаёт зону разделяемой памяти и максимально допустимое число соединений для одного значения ключа. При превышении этого числа в ответ на запрос сервер вернёт ошибку. Например, директивы

```
limit_conn_zone $binary_remote_addr zone=addr:10m;

server {
    location /download/ {
        limit_conn addr 1;
    }
}
```

разрешают одновременно обрабатывать не более одного соединения с одного IP-адреса.

Примечание

В HTTP/2 и HTTP/3 каждый параллельный запрос считается отдельным соединением.

Директив *limit_conn* может быть несколько. Например, следующая конфигурация ограничивает число соединений с сервером с одного клиентского IP-адреса и в то же время ограничивает общее число соединений с виртуальным сервером:

```
limit_conn_zone $binary_remote_addr zone=perip:10m;
limit_conn_zone $server_name zone=perserver:10m;

server {
    ...
    limit_conn perip 10;
    limit_conn perserver 100;
}
```

Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы *limit_conn*.

limit_conn_dry_run

Синтаксис	limit_conn_dry_run on off;
По умолчанию	limit_conn_dry_run off;
Контекст	http, server, location

Включает режим пробного запуска. В данном режиме число соединений не ограничивается, однако в зоне разделяемой памяти текущее число избыточных соединений учитывается как обычно.

limit_conn_log_level

Синтаксис	<code>limit_conn_log_level info notice warn error;</code>
По умолчанию	<code>limit_conn_log_level error;</code>
Контекст	<code>http, server, location</code>

Задаёт желаемый уровень записи в лог случаев ограничения числа соединений.

limit_conn_status

Синтаксис	<code>limit_conn_status код;</code>
По умолчанию	<code>limit_conn_status 503;</code>
Контекст	<code>http, server, location</code>

Позволяет переопределить код ответа, используемый при отклонении запросов.

limit_conn_zone

Синтаксис	<code>limit_conn_zone <i>ключ</i> zone = <i>название:размер</i>;</code>
По умолчанию	—
Контекст	<code>http</code>

Задаёт параметры зоны разделяемой памяти, которая хранит состояние для разных значений ключа. Состояние в частности содержит текущее число соединений. В качестве ключа можно использовать переменные. Запросы с пустым значением ключа не учитываются.

Пример использования:

```
limit_conn_zone $binary_remote_addr zone=addr:10m;
```

Здесь в качестве ключа используется IP-адрес клиента. Обратите внимание, что вместо переменной `$remote_addr` использована переменная `$binary_remote_addr`.

Длина значения переменной `$remote_addr` может колебаться от 7 до 15 байт, при этом размер хранимого состояния составляет либо 32, либо 64 байта на 32-битных платформах и всегда 64 байта на 64-битных.

Длина значения переменной `$binary_remote_addr` всегда равна 4 байтам для IPv4-адресов или 16 байтам для IPv6-адресов. При этом размер состояния всегда равен 32 или 64 байтам на 32-битных платформах и 64 байтам на 64-битных.

В зоне размером 1 мегабайт может разместиться около 32 тысяч состояний размером 32 байта или 16 тысяч состояний размером 64 байта. При переполнении зоны в ответ на последующие запросы сервер будет возвращать ошибку.

Встроенные переменные

\$limit_conn_status

хранит результат ограничения числа соединений: `PASSED`, `REJECTED` или `REJECTED_DRY_RUN`

Limit Req

Позволяет ограничить скорость обработки запросов по заданному ключу или, как частный случай, скорость обработки запросов, поступающих с одного IP-адреса. Ограничение обеспечивается с помощью метода "leaky bucket".

Пример конфигурации

```
http {
    limit_req_zone $binary_remote_addr zone=one:10m rate=1r/s;

    ...

    server {

        ...

        location /search/ {
            limit_req zone=one burst=5;
        }
    }
}
```

Директивы

limit_req

Синтаксис	<code>limit_req zone=название [burst=число] [nodelay delay=число];</code>
По умолчанию	—
Контекст	http, server, location

Задаёт зону разделяемой памяти (**zone**) и максимальный размер всплеска запросов (**burst**). Если скорость поступления запросов превышает описанную в зоне, то их обработка задерживается так, чтобы запросы обрабатывались с заданной скоростью. Избыточные запросы задерживаются до тех пор, пока их число не превысит максимальный размер всплеска. При превышении запрос завершается с ошибкой. По умолчанию максимальный размер всплеска равен нулю. Например, директивы

```
limit_req_zone $binary_remote_addr zone=one:10m rate=1r/s;

server {
    location /search/ {
        limit_req zone=one burst=5;
    }
}
```

позволяют в среднем не более 1 запроса в секунду со всплесками не более 5 запросов.

Если же избыточные запросы в пределах лимита всплесков задерживать не требуется, то следует использовать параметр **nodelay**:

```
limit_req zone=one burst=5 nodelay;
```

Параметр **delay** задаёт лимит, по достижении которого избыточные запросы задерживаются. Значение по умолчанию равно нулю и означает, что задерживаются все избыточные запросы.

Директив **limit_req** может быть несколько. Например, следующая конфигурация ограничивает скорость обработки запросов, поступающих с одного IP-адреса, и в то же время ограничивает скорость обработки запросов одним виртуальным сервером:

```
limit_req_zone $binary_remote_addr zone=perip:10m rate=1r/s;
limit_req_zone $server_name zone=perserver:10m rate=10r/s;

server {
    ...
    limit_req zone=perip burst=5 nodelay;
    limit_req zone=perserver burst=10;
}
```

Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы `limit_req`.

limit_req_dry_run

Синтаксис	<code>limit_req_dry_run on off;</code>
По умолчанию	<code>limit_req_dry_run off;</code>
Контекст	<code>http, server, location</code>

Включает режим пробного запуска. В данном режиме скорость обработки запросов не ограничивается, однако в зоне разделяемой памяти текущее число избыточных запросов учитывается как обычно.

limit_req_log_level

Синтаксис	<code>limit_req_log_level info notice warn error;</code>
По умолчанию	<code>limit_req_log_level error;</code>
Контекст	<code>http, server, location</code>

Задаёт желаемый уровень записи в лог случаев отказа в обработке запросов при превышении скорости и случаев задержек при обработке запроса. Задержки записываются в лог с уровнем на единицу меньше, чем отказы, например, если указано `limit_req_log_level notice;`, то задержки будут записываться в лог на уровне `info`.

limit_req_status

Синтаксис	<code>limit_req_status код;</code>
По умолчанию	<code>limit_req_status 503;</code>
Контекст	<code>http, server, location</code>

Позволяет переопределить код ответа, используемый при отклонении запросов.

limit_req_zone

Синтаксис	<code>limit_req_zone <i>ключ</i> zone=<i>название:размер</i> rate=<i>скорость</i>;</code>
По умолчанию	—
Контекст	<code>http</code>

Задаёт параметры зоны разделяемой памяти, которая хранит состояние для разных значений ключа. Состояние в частности хранит текущее число избыточных запросов. В качестве ключа можно использовать переменные. Запросы с пустым значением ключа не учитываются.

Пример использования:

```
limit_req_zone $binary_remote_addr zone=one:10m rate=1r/s;
```

В данном случае состояния хранятся в зоне `one` размером 10 мегабайт, и средняя скорость обработки запросов для этой зоны не может превышать 1 запроса в секунду.

В качестве ключа используется IP-адрес клиента. Обратите внимание, что вместо переменной `$remote_addr` используется переменная `$binary_remote_addr`.

Длина значения переменной `$binary_remote_addr` всегда равна 4 байтам для IPv4-адресов или 16 байтам для IPv6-адресов. При этом размер состояния всегда равен 64 байтам на 32-битных платформах и 128 байтам на 64-битных платформах.

В зоне размером 1 мегабайт может разместиться около 16 тысяч состояний размером 64 байта или около 8 тысяч состояний размером 128 байт.

При переполнении зоны удаляется наименее востребованное состояние. Если и это не позволяет создать новое состояние, запрос завершается с ошибкой.

Скорость `rate` задается в запросах в секунду (r/s). Если же нужна скорость меньше одного запроса в секунду, то она задается в запросах в минуту (r/m), например, ползапроса в секунду — это 30r/m.

Встроенные переменные

`$limit_req_status`

хранит результат ограничения скорости поступления запросов: `PASSED`, `DELAYED`, `REJECTED`, `DELAYED_DRY_RUN` или `REJECTED_DRY_RUN`

Log

Модуль записывает логи запросов в указанном формате.

Логи записываются в контексте `location`, где заканчивается обработка. Это может быть `location`, отличный от первоначального, если в процессе обработки запроса происходит внутреннее перенаправление.

Пример конфигурации

```
log_format compression '$remote_addr - $remote_user [$time_local] '
    '$request' $status $bytes_sent '
    '$http_referer' '$http_user_agent' '$gzip_ratio';

access_log /spool/logs/angie-access.log compression buffer=32k;
```

Директивы

access_log

Синтаксис	<code>access_log <i>путь</i> [<i>формат</i>] [<i>buffer=размер</i>] [<i>gzip=степень</i>] [<i>flush=время</i>] [<i>if=условие</i>];</code> <code>access_log off;</code>
По умолчанию	<code>access_log logs/access.log combined;</code> (путь зависит от параметра сборки <code>--http-log-path</code>)
Контекст	<code>http</code> , <code>server</code> , <code>location</code> , <code>if</code> в <code>location</code> , <code>limit_except</code>

Задаёт *путь*, *формат* и настройки буферизованной записи в лог. На одном уровне конфигурации может использоваться несколько логов. Запись в syslog настраивается указанием префикса `"syslog:"` в первом параметре. Специальное значение `off` отменяет все директивы `access_log` для текущего уровня. Если формат не указан, то используется предопределённый формат `"combined"`.

Если задан размер буфера с помощью параметра `buffer` или указан параметр `gzip`, то запись будет буферизованной.

Предупреждение

Размер буфера должен быть не больше размера атомарной записи в дисковый файл. Для FreeBSD этот размер неограничен.

При включенной буферизации данные записываются в файл:

- если очередная строка лога не помещается в буфер;
- если данные в буфере находятся дольше интервала времени, заданного параметром `flush`;
- при переоткрытии лог-файла или завершении рабочего процесса.

Если задан параметр `gzip`, то буфер будет сжиматься перед записью в файл. Степень сжатия может быть задана в диапазоне от 1 (быстрее, но хуже сжатие) до 9 (медленнее, но лучше сжатие). По умолчанию используются буфер размером *64K* байт и степень сжатия *1*. Данные сжимаются атомарными блоками, и в любой момент времени лог-файл может быть распакован или прочитан с помощью утилиты `zcat`.

Пример:

```
access_log /path/to/log.gz combined gzip flush=5m;
```

Примечание

Для поддержки gzip-сжатия логов Angie должен быть собран с библиотекой `zlib`.

В пути файла можно использовать переменные, но такие логи имеют некоторые ограничения:

- пользователь, с правами которого работают рабочие процессы, должен иметь права на создание файлов в каталоге с такими логами;
- не работает буферизация;
- файл открывается для каждой записи в лог и сразу же после записи закрывается. Следует однако иметь в виду, что поскольку дескрипторы часто используемых файлов могут храниться в кэше, то при ротации логов в течение времени, заданного параметром `valid` директивы `open_log_file_cache`, запись может продолжаться в старый файл.
- при каждой записи в лог проверяется существование корневого каталога для запроса — если этот каталог не существует, то лог не создается. Поэтому `root` и `access_log` нужно описывать на одном уровне конфигурации:

```
server {
    root      /spool/vhost/data/$host;
    access_log /spool/vhost/logs/$host;
    ...
}
```

Параметр `if` включает условную запись в лог. Запрос не будет записываться в лог, если результатом вычисления условия является `"0"` или пустая строка. В следующем примере запросы с кодами ответа 2xx и 3xx не будут записываться в лог:

```
map $status $loggable {
    ~^[23] 0;
    default 1;
}

access_log /path/to/access.log combined if=$loggable;
```

log_format

Синтаксис	log_format <i>имя</i> [escape= default json none] <i>строка</i> ...;
По умолчанию	log_format combined "...";
Контекст	http

Задаёт формат лога.

Параметр **escape** позволяет задать экранирование символов **json** или **default** в переменных, по умолчанию используется **default**. Значение **none** отключает экранирование символов.

При использовании **default** символы `"`, `\`, а также символы со значениями меньше 32 или больше 126 экранируются как `"\xXX"`. Если значение переменной не найдено, то в качестве значения в лог будет записываться дефис `-`.

При использовании **json** экранируются все символы, недопустимые в JSON строках: символы `"` и `\` экранируются как `"\"` и `\\`, символы со значениями меньше 32 экранируются как `"\n"`, `"\r"`, `"\t"`, `"\b"`, `"\f"` или `"\u00XX"`.

Строки заголовка, переданные клиенту, начинаются с префикса **sent_http_**, например, **\$sent_http_content_range**.

В конфигурации всегда существует предопределённый формат **combined**:

```
log_format combined '$remote_addr - $remote_user [$time_local] '
                    '$request' $status $body_bytes_sent '
                    '$http_referer' '$http_user_agent';
```

open_log_file_cache

Синтаксис	open_log_file_cache max= <i>N</i> [inactive= <i>время</i>] [min_uses= <i>N</i>] [valid= <i>время</i>]; open_log_file_cache off;
По умолчанию	open_log_file_cache off;
Контекст	http, server, location

Задаёт кэш, в котором хранятся дескрипторы файлов часто используемых логов, имена которых заданы с использованием переменных. Параметры:

max	Задаёт максимальное число дескрипторов в кэше; при переполнении кэша наименее востребованные (LRU) дескрипторы закрываются.
inactive	Задаёт время, после которого кэшированный дескриптор закрывается, если к нему не было обращений в течение этого времени. По умолчанию — 10 секунд.
min_uses	Задаёт минимальное число использований файла в течение времени, заданного параметром inactive , после которого дескриптор файла будет оставаться открытым в кэше. По умолчанию — 1.
valid	Указывает, через какое время нужно проверять, что файл ещё существует под тем же именем. По умолчанию — 60 секунд.
off	Запрещает кэширование.

Пример использования:

```
open_log_file_cache max=1000 inactive=20s valid=1m min_uses=2;
```

Map

Создаёт переменные, значения которых зависят от значений других переменных.

Пример конфигурации

```
map $http_host $name {
    hostnames;

    default      0;

    example.com  1;
    *.example.com 1;
    example.org   2;
    *.example.org 2;
    .example.net  3;
    wap.*         4;
}

map $http_user_agent $mobile {
    default      0;
    "~Opera Mini" 1;
}
```

Директивы

map

Синтаксис	<code>map строка \$переменная { ... }</code>
По умолчанию	—
Контекст	http

Создаёт новую переменную. Её значение зависит от первого параметра, заданного строкой с переменными, например:

```
set $var1 "foo";
set $var2 "bar";
```

```
map $var1$var2 $new_variable {
    default "foobar_value";
}
```

Здесь переменная `$new_variable` будет иметь значение, составленное из двух переменных `$var1` и `$var2`, или значение по умолчанию, если эти переменные не определены.

Примечание

Поскольку переменные вычисляются только в момент использования, само по себе наличие даже большого числа объявлений переменных "map" не влечет за собой никаких дополнительных расходов на обработку запросов.

Параметры внутри блока `map` задают соответствие между исходными и результирующими значениями.

Исходные значения задаются строками или регулярными выражениями.

Строки проверяются без учета регистра.

Перед регулярным выражением ставится символ `~`, если при сравнении следует учитывать регистр символов, либо символы `~*`, если регистр символов учитывать не нужно. Регулярное выражение может содержать именованные и позиционные группы захвата, которые могут затем использоваться в других директивах совместно с результирующей переменной.

Если исходное значение совпадает с именем одного из специальных параметров, описанных ниже, перед ним следует поставить символ `\`.

В качестве результирующего значения можно указать текст, переменную и их комбинации.

Также поддерживаются следующие специальные параметры:

default <i>значение</i>	задает результирующее значение, если исходное значение не совпадает ни с одним из перечисленных. Если параметр <i>default</i> не указан, результирующим значением по умолчанию будет пустая строка.
hostnames	указывает, что в качестве исходных значений можно использовать маску для первой или последней части имени хоста. Этот параметр следует указывать перед списком значений.

Например,

```
*.example.com 1;
example.*      1;
```

Вместо двух записей

```
example.com    1;
*.example.com  1;
```

можно использовать одну:

```
.example.com 1;
```

include <i>файл</i>	включает файл со значениями. Включений может быть несколько.
volatile	указывает, что переменная не кэшируется.

Если исходному значению соответствует несколько из указанных вариантов, например, одновременно подходят и маска, и регулярное выражение, будет выбран первый подходящий вариант в следующем порядке приоритета:

1. Строковое значение без маски.
2. Самое длинное строковое значение с маской в начале, например `"*.example.com"`.
3. Самое длинное строковое значение с маской в конце, например `"mail.*"`.
4. Первое подходящее регулярное выражение (в порядке следования в конфигурационном файле).
5. Значение по умолчанию (*default*).

map_hash_bucket_size

Синтаксис	<code>map_hash_bucket_size</code> размер;
По умолчанию	<code>map_hash_bucket_size 32 64 128;</code>
Контекст	<code>http</code>

Задаёт размер корзины в хэш-таблицах для переменных *map*. Значение по умолчанию зависит от размера строки кэша процессора. Подробнее настройка хэш-таблиц обсуждается отдельно.

map_hash_max_size

Синтаксис	<code>map_hash_max_size</code> размер;
По умолчанию	<code>map_hash_max_size 2048;</code>
Контекст	<code>http</code>

Задаёт максимальный размер хэш-таблиц для переменных *map*. Подробнее настройка хэш-таблиц обсуждается отдельно.

Memcached

Позволяет получать ответ из сервера memcached. Ключ задается в переменной *\$memcached_key*. Ответ в memcached должен быть предварительно помещен внешним по отношению к Angie способом.

Пример конфигурации

```
server {
    location / {
        set                $memcached_key "$uri?$args";
        memcached_pass     host:11211;
        error_page         404 502 504 = @fallback;
    }

    location @fallback {
        proxy_pass         http://backend;
    }
}
```


Директивы

memcached_bind

Синтаксис	<code>memcached_bind адрес [transparent] off;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт локальный IP-адрес с необязательным портом, который будет использоваться в исходящих соединениях с сервером memcached. В значении параметра можно использовать переменные. Специальное значение `off` отменяет действие унаследованной с предыдущего уровня конфигурации директивы `memcached_bind`, позволяя системе самостоятельно выбирать локальный IP-адрес и порт.

По умолчанию это значение вычисляется один раз при инициализации запроса, до выбора сервера memcached. В Angie PRO вычисление происходит после каждого выбора сервера memcached (в том числе при повторных попытках), поэтому адрес привязки может зависеть от выбранного сервера; например, с помощью переменной `$upstream_current_addr`. Если вычисленное значение пусто, привязка не выполняется.

Параметр `transparent` позволяет задать нелокальный IP-адрес, который будет использоваться в исходящих соединениях с сервером memcached, например, реальный IP-адрес клиента:

```
memcached_bind $remote_addr transparent;
```

Для работы параметра обычно требуется запустить рабочие процессы Angie с привилегиями суперпользователя. В Linux это не требуется, так как если указан параметр `transparent`, то рабочие процессы наследуют `capability CAP_NET_RAW` из главного процесса.

Примечание

Необходимо настроить таблицу маршрутизации ядра для перехвата сетевого трафика с сервера memcached.

memcached_buffer_size

Синтаксис	<code>memcached_buffer_size размер;</code>
По умолчанию	<code>memcached_buffer_size 4k 8k;</code>
Контекст	http, server, location

Задаёт размер буфера, в который будет читаться первая часть ответа, получаемого от сервера memcached. Ответ синхронно передается клиенту сразу же по мере его поступления.

memcached_connect_timeout

Синтаксис	<code>memcached_connect_timeout время;</code>
По умолчанию	<code>memcached_connect_timeout 60s;</code>
Контекст	http, server, location

Задаёт таймаут для установления соединения с сервером memcached. Необходимо иметь в виду, что этот таймаут обычно не может превышать 75 секунд.

Примечание

Этот таймаут не управляет неактивными соединениями: время, в течение которого уже открытое соединение сохраняется для повторного использования, задается директивой *keepalive_timeout* в блоке upstream, соединения которого кэшируются директивой *keepalive*.

memcached_gzip_flag

Синтаксис	<code>memcached_gzip_flag <i>флаг</i>;</code>
По умолчанию	—
Контекст	http, server, location

Включает проверку указанного флага в ответе сервера memcached и установку поля Content-Encoding заголовка ответа в "gzip", если этот флаг установлен.

memcached_interface

Синтаксис	<code>memcached_interface <i>имя</i>;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт сетевой интерфейс, который будет использоваться в исходящих соединениях с сервером memcached; сокет привязывается к нему параметром сокета SO_BINDTODEVICE.

memcached_next_upstream

Синтаксис	<code>memcached_next_upstream <i>error</i> <i>timeout</i> <i>invalid_response</i> <i>not_found</i> <i>off</i> ...;</code>
По умолчанию	<code>memcached_next_upstream <i>error</i> <i>timeout</i>;</code>
Контекст	http, server, location

Определяет, в каких случаях запрос будет передан следующему в группе upstream серверу:

error	произошла ошибка соединения с сервером, передачи ему запроса или чтения заголовка ответа сервера;
timeout	произошел таймаут во время соединения с сервером, передачи ему запроса или чтения заголовка ответа сервера;
invalid_response	сервер вернул пустой или неверный ответ;
not_found	сервер не нашел ответ;
off	запрещает передачу запроса следующему серверу.

Примечание

Необходимо понимать, что передача запроса следующему серверу возможна только при условии, что клиенту еще ничего не передавалось. То есть, если ошибка или таймаут возникли в середине передачи ответа клиенту, то действие директивы на такой запрос не распространяется.

Директива также определяет, что считается неудачной попыткой работы с сервером.

<code>error, timeout,</code>	Всегда считаются неудачными попытками, даже если они не указаны в директиве.
<code>invalid_response</code>	
<code>not_found</code>	Никогда не считается неудачными попытками.

Передача запроса следующему серверу может быть ограничена по количеству попыток и по времени.

memcached_next_upstream_timeout

Синтаксис	<code>memcached_next_upstream_timeout</code> время;
По умолчанию	<code>memcached_next_upstream_timeout 0;</code>
Контекст	http, server, location

Ограничивает время, в течение которого возможна передача запроса следующему серверу.

0	отключает это ограничение
---	---------------------------

memcached_next_upstream_tries

Синтаксис	<code>memcached_next_upstream_tries</code> число;
По умолчанию	<code>memcached_next_upstream_tries 0;</code>
Контекст	http, server, location

Ограничивает число допустимых попыток для передачи запроса следующему серверу.

0	отключает это ограничение
---	---------------------------

memcached_pass

Синтаксис	<code>memcached_pass</code> адрес;
По умолчанию	—
Контекст	location, if в location

Задаёт адрес сервера memcached. Адрес может быть указан в виде доменного имени или IP-адреса, и порта:

```
memcached_pass localhost:11211;
```

или в виде пути UNIX-сокета:

```
memcached_pass unix:/tmp/memcached.socket;
```

Если доменному имени соответствует несколько адресов, то все они будут использоваться по очереди (round-robin). Кроме того, в качестве адреса можно указать группу серверов.

i Примечание

Если `memcached_pass` стоит в `location` с косой чертой в конце префикса (например, `location /name/`), и при этом в директиве `auto_redirect` указано `default`, запросы без косой черты в конце будут перенаправляться (`/name -> /name/`).

`memcached_read_timeout`

Синтаксис	<code>memcached_read_timeout время;</code>
По умолчанию	<code>memcached_read_timeout 60s;</code>
Контекст	<code>http, server, location</code>

Задаёт таймаут при чтении ответа сервера `memcached`. Таймаут устанавливается не на всю передачу ответа, а только между двумя операциями чтения. Если по истечении этого времени сервер `memcached` ничего не передаст, соединение закрывается.

`memcached_send_timeout`

Синтаксис	<code>memcached_send_timeout время;</code>
По умолчанию	<code>memcached_send_timeout 60s;</code>
Контекст	<code>http, server, location</code>

Задаёт таймаут при передаче запроса серверу `memcached`. Таймаут устанавливается не на всю передачу запроса, а только между двумя операциями записи. Если по истечении этого времени сервер `memcached` не примет новых данных, соединение закрывается.

`memcached_socket_keepalive`

Синтаксис	<code>memcached_socket_keepalive on off;</code>
По умолчанию	<code>memcached_socket_keepalive off;</code>
Контекст	<code>http, server, location</code>

Конфигурирует поведение "TCP keepalive" для исходящих соединений к проксируемому серверу.

<code>""</code>	По умолчанию для сокета действуют настройки операционной системы.
<code>on</code>	для сокета включается параметр <code>SO_KEEPALIVE</code>

`memcached_tcp_congestion`

Синтаксис	<code>memcached_tcp_congestion алгоритм;</code>
По умолчанию	—
Контекст	<code>http, server, location</code>

Задаёт алгоритм управления перегрузкой TCP (параметр сокета `TCP_CONGESTION`) для исходящих соединений с сервером `memcached`; к соединениям через UNIX-сокеты параметр не применяется.

Имя алгоритма не проверяется на этапе конфигурации: если алгоритм неизвестен или недоступен, при установке исходящего соединения в лог записывается сообщение уровня **alert**, а для соединения действует системный алгоритм по умолчанию.

Встроенные переменные

`$memcached_key`

Задаёт ключ для получения ответа из сервера memcached.

Metric

Модуль `ngx_http_metric_module` позволяет создавать вычисляемые в реальном времени произвольные метрики. Значения таких метрик сохраняются в разделяемой памяти и отображаются в реальном времени в ветке API `/status/http/metric_zones/`. Поддерживаются различные типы агрегации данных (счетчики, гистограммы, скользящие средние и др.) с группировкой по произвольным ключам.

Схема ответа приведена в справочнике API зон HTTP-метрик.

Примечание

При перезагрузке конфигурации зона, созданная директивой `metric_zone` или `metric_complex_zone`, сохраняет накопленные значения только при неизменной структуре хранимых данных. Зона начинает работу пустой, если от предыдущей конфигурации отличаются размер зоны, количество метрик или режим, если изменился параметр `count` режима **average mean**, если параметр `window` режима **average mean** переключен в **off** или обратно или если изменилось количество корзин **histogram**; причина при этом записывается в *error_log* на уровне **info**. Остальные параметры, например `factor` режима **average exp**, интервал `window` режима **average mean** или границы корзин **histogram**, можно менять без потери накопленных значений.

Пример конфигурации

Подсчет запросов к API:

```
http {
    metric_zone api_requests:1m count;

    server {
        listen 80;

        location /api/ {
            allow 127.0.0.1;
            deny all;
            api /status/;

            metric api_requests $http_user_agent on=request;
        }
    }
}
```

Если с такой конфигурацией выполнить запрос на `/api/`:

```
$ curl 127.0.0.1/api/ --user-agent "Firefox"
```

В реальном времени происходит обновление метрики `api_requests`:

```
{
  "http": {
    "metric_zones": {
      "api_requests": {
        "discarded": 0,
        "metrics": {
          "Firefox": 1
        }
      }
    }
  }
}
```

Директивы

metric

Синтаксис	<code>metric название ключ=значение [on=request response end];</code>
По умолчанию	—
Контекст	http, server, location

Подсчитывает значение метрики с указанным *названием* зоны разделяемой памяти.

Параметры:

- *ключ* — произвольная строка (часто переменная), по которой группируются значения. Максимальная длина — 255 байт; более длинные ключи обрезаются. В выводе API Angie добавляет ... к любому ключу длиной 255 байт, даже если до сохранения его длина составляла ровно 255 байт. Если ключ сам содержит один или несколько символов =, в качестве *значения* воспринимается только подстрока после последнего =; все, что стоит до него, становится частью ключа;
- *значение* — число (может быть переменной), обрабатываемое выбранным режимом. Если пропущено, считается 0. Если параметр невозможно преобразовать в число, считается 1;
- *on* — необязательный параметр, указывающий в какой момент происходит подсчет метрики:
 - Если *on=request*, подсчет происходит при получении запроса;
 - Если *on=response*, подсчет происходит при подготовке ответа;
 - Если *on=end* (по умолчанию), подсчет происходит после отправки ответа.

Примечание

В случае внутреннего перенаправления, метрики на стадии *on=request* посчитаются в исходном *location*. При этом метрики *on=response* и *on=end* будут подсчитаны уже в новом *location*.

Директив *metric* может быть несколько. Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы *metric*. Поэтому одна директива *metric* в блоке *location* отменяет все правила, унаследованные от *server* или *http*, на всех трех стадиях *on*, а не только на своей.

Пример использования:

```
metric requests $http_user_agent=$request_time;
```

Примечание

Метрики с пустым ключом или неправильной парой **ключ=значение** не учитываются. Пропущенное *значение* учитывается как 0:

```
metric foo $bar; # Вместо $bar=0
```

Что полезно, например, для режима **count**, который игнорирует числовое значение и реагирует на факт обновления метрики.

Примечание

Важно помнить, что переменные вычисляются в разных фазах. Например, невозможно использовать **\$bytes_sent** (количество отправленных байт клиенту) при **on=request** (при получении запроса).

metric_complex_zone

Синтаксис	metric_complex_zone	название:размер	[expire=on off]
	[discard_key=название] { ... }		
По умолчанию	—		
Контекст	http		

Определяет *составную метрику* — набор метрик с независимыми режимами. Каждая строка в теле блока задает *название подметрики*, *режим* и необязательные *параметры* режима. Размер зоны должен составлять не менее восьми системных страниц памяти: 32К при размере страницы 4К.

Пример использования:

```
metric_complex_zone requests:1m expire=on discard_key="old" {
    # название подметрики    режим работы    параметры
    min_time                  min;
    avg_time                  average exp    factor=60;
    max_time                  max;
    total                     count;
}
```

В API дереве шаблон такой составной метрики выглядит следующим образом:

```
{
  "discarded": 3,
  "metrics": {
    "ключ1": {
      "min_time": 20,
      "avg_time": 50,
      "max_time": 80,
      "total": 2
    },
    "old": {
      "min_time": 3,
      "avg_time": 40,
      "max_time": 152,
      "total": 80
    }
  }
}
```

```
}
}
}
```

metric_zone

Синтаксис	<code>metric_zone <i>название:размер</i> [expire=on off] [discard_key=<i>название</i>] <i>режим</i> [<i>параметры</i>];</code>
По умолчанию	—
Контекст	http

Создает зону разделяемой памяти указанного *размера* с именем *название*, в которой хранятся метрики. Имя зоны является узлом в ветке `/status/http/metric_zones/`. Размер зоны должен составлять не менее восьми системных страниц памяти: 32К при размере страницы 4К.

Параметры:

- `expire=<on|off>` — поведение при переполнении зоны:
 - Если `on`, самые старые метрики по времени обновления отбрасываются, освобождая память под поступающие;
 - Если `off` (по умолчанию) — отбрасываются поступающие метрики, сохраняя информацию об установленных.
- `discard_key=<название>` — задает метрику с ключом *название*, в которой сохраняются значения отброшенных метрик. По умолчанию такая метрика не создается. Зарезервированный ключ нельзя обновлять вручную.
- *режим* — алгоритм обработки значений (см. раздел Режимы работы);
- *параметры* — дополнительная настройка выбранного режима (например, `factor` для `average exp`).

Пример использования:

```
metric_zone request_time:1m max;
```

В API дереве шаблон зоны разделяемой памяти выглядит следующим образом:

```
{
  "discarded": 0,
  "metrics": {
    "ключ1": 123,
    "ключ2": 10.5
  }
}
```

Примечание

В зоне размером 1 мегабайт при размере ключа 39 байт и с одним режимом метрики может разместиться около 8 тысяч записей с уникальным ключом.

Режимы работы

Список доступных режимов работы метрик:

- `count` — счетчик обращений;
- `gauge` — стрелка (инкремент/декремент);
- `last` — последнее поступившее значение;
- `min` — минимальное значение;
- `max` — максимальное значение;
- `average exp` — скользящее среднее (параметр `factor`);
- `average mean` — среднее за окно (параметры `window` и `count`);
- `histogram` — распределение по "бакетам" (перечень пороговых значений).

В примерах ниже используется конечная точка `/api/` из примера конфигурации. Результат каждого примера доступен по адресу `/api/http/metric_zones/<зона>/metrics/`.

count

Каждое обновление метрики увеличивает счетчик на 1 независимо от переданного значения.

Значение по умолчанию — 0.

Примеры:

```
metric_zone count:1m count;

# В качестве составной метрики:
#
# metric_complex_zone count:1m {
#     some_metric_name count;
# }

server {
    listen 80;

    location /metric/ {
        metric count KEY;
    }

    location ~ ^/metric/set/(.+)$ {
        metric count KEY=$1;
    }
}
```

Обновление метрики:

```
$ curl 127.0.0.1/metric/
$ curl 127.0.0.1/metric/set/1
$ curl 127.0.0.1/metric/set/23
$ curl 127.0.0.1/metric/set/-32
```

Ожидаемое значение метрики в API:

```
{
  "KEY": 4
}
```

gauge

Счетчик увеличивает или уменьшает свое значение в зависимости от знака переданного числа. При положительном — увеличивает счетчик. При отрицательном — уменьшает. Переданное значение 0 не изменяет счетчик.

Значение по умолчанию — 0.

Примеры:

```
metric_zone gauge:1m gauge;

# В качестве составной метрики:
#
# metric_complex_zone gauge:1m {
#     some_metric_name gauge;
# }

server {
    listen 80;

    location /metric/ {
        metric gauge KEY;
    }

    location ~ ^/metric/set/(.+)$ {
        metric gauge KEY=$1;
    }
}
```

Обновление метрики:

```
$ curl 127.0.0.1/metric/
```

Ожидаемое значение метрики в API:

```
{
  "KEY": 0
}
```

Обновление метрики:

```
$ curl 127.0.0.1/metric/set/5
$ curl 127.0.0.1/metric/set/-5
$ curl 127.0.0.1/metric/set/8
```

Ожидаемое значение метрики в API:

```
{
  "KEY": 8
}
```

last

Хранит последнее полученное значение без какой-либо агрегации. Если *значение* опущено, используется 0.

Примеры:

```
metric_zone last:1m last;

# В качестве составной метрики:
#
# metric_complex_zone last:1m {
#     some_metric_name last;
# }

server {
    listen 80;

    location /metric/ {
        metric last KEY;
    }

    location ~ ^/metric/set/(.+) $ {
        metric last KEY=$1;
    }
}
```

Обновление метрики:

```
$ curl 127.0.0.1/metric/
```

Ожидаемое значение метрики в API:

```
{
  "KEY": 0
}
```

Обновление метрики:

```
$ curl 127.0.0.1/metric/set/8000
$ curl 127.0.0.1/metric/set/37
$ curl 127.0.0.1/metric/set/-3.5
```

Ожидаемое значение метрики в API:

```
{
  "KEY": -3.5
}
```

min

Сохраняет минимальное из двух значений — уже сохраненного и нового.

Примеры:

```
metric_zone min:1m min;

# В качестве составной метрики:
#
# metric_complex_zone min:1m {
#     some_metric_name min;
# }

server {
```

```
listen 80;

location /metric/ {
    metric min KEY;
}

location ~ ^/metric/set/(.+)$ {
    metric min KEY=$1;
}

}
```

Обновление метрики:

```
$ curl 127.0.0.1/metric/set/42.999
$ curl 127.0.0.1/metric/set/-512
$ curl 127.0.0.1/metric/set/1
$ curl 127.0.0.1/metric/
```

Ожидаемое значение метрики в API:

```
{
  "KEY": -512
}
```

max

Сохраняет наибольшее из двух значений — уже сохраненного и нового.

Примеры:

```
metric_zone max:1m max;

# В качестве составной метрики:
#
# metric_complex_zone max:1m {
#     some_metric_name max;
# }

server {
    listen 80;

    location /metric/ {
        metric max KEY;
    }

    location ~ ^/metric/set/(.+)$ {
        metric max KEY=$1;
    }

}
```

Обновление метрики:

```
$ curl 127.0.0.1/metric/set/42.999
$ curl 127.0.0.1/metric/set/-512
$ curl 127.0.0.1/metric/set/1
$ curl 127.0.0.1/metric/
```

Ожидаемое значение метрики в API:

```
{
  "KEY": 42.999
}
```

average exp

Вычисляет среднее значение по алгоритму экспоненциального сглаживания.

Принимает необязательный параметр **factor=<число>** — коэффициент, по которому установленное значение учитывается при расчете среднего. Допустимы целые числа от 0 до 99. По умолчанию — 90.

Чем больше коэффициент, тем сильнее новые значения влияют на среднее. Если указать 90, то будет взято 90% от нового значения и лишь 10% от предыдущего.

Примеры:

```
metric_zone avg_exp:1m average exp factor=60;

# В качестве составной метрики:
#
# metric_complex_zone avg_exp:1m {
#   some_metric_name average exp factor=60;
# }

server {
    listen 80;

    location ~ ^/metric/set/(.+) $ {
        metric avg_exp KEY=$1;
    }
}
```

Обновление метрики:

```
$ curl 127.0.0.1/metric/set/100
$ curl 127.0.0.1/metric/set/200
$ curl 127.0.0.1/metric/set/0
$ curl 127.0.0.1/metric/set/8
$ curl 127.0.0.1/metric/set/30
```

Ожидаемое значение метрики в API:

```
{
  "KEY": 30.16
}
```

average mean

Вычисляет среднее арифметическое. Принимает необязательные параметры **window=<off|время>** и **count=<число>**, задающие соответственно интервал времени и объем выборки для усреднения. По умолчанию: **window=off** (учитывается вся выборка) и **count=10**.

Примечание

Например, `window=5s` будет учитывать только события последних 5 секунд. Параметр `window` не может принимать значение 0. Параметр `count=число` управляет размером выборки (закэшированных значений) для более плавного подсчета среднего.

Примеры:

```
metric_zone avg_mean:1m average mean window=5s count=8;

# В качестве составной метрики:
#
# metric_complex_zone avg_mean:1m {
#     some_metric_name average mean window=5s count=8;
# }

server {
    listen 80;

    location ~ ^/metric/set/(.+)$ {
        metric avg_mean KEY=$1;
    }
}
```

Обновление метрики:

```
$ curl 127.0.0.1/metric/set/0.1
$ curl 127.0.0.1/metric/set/0.1
$ curl 127.0.0.1/metric/set/0.4
$ curl 127.0.0.1/metric/set/10
$ curl 127.0.0.1/metric/set/1
$ curl 127.0.0.1/metric/set/1
```

Ожидаемое значение метрики в API:

```
{
  "KEY": 2.1
}
```

Если подождать 5 секунд, с момента последнего обновления метрики, то ожидаемое значение метрики в API:

```
{
  "KEY": 0
}
```

histogram

Создает набор "бакетов", увеличивая соответствующий счетчик, если новое значение не превосходит порога бакета. Формат параметров — список числовых порогов. Полезен для анализа распределения, например времени ответа.

В качестве обязательных параметров передаются *числа* — предельные значения бакетов, обычно перечисляемые в порядке возрастания.

Примечание

Значение бакета `inf` или `+Inf` можно использовать для захвата всех значений, превышающих максимальный бакет.

Примеры:

```
metric_zone hist:1m histogram 0.1 0.2 0.5 1 2 inf;

# В качестве составной метрики:
#
# metric_complex_zone hist:1m {
#     some_metric_name histogram 0.1 0.2 0.5 1 2 inf;
# }

server {
    listen 80;

    location ~ ~/metric/set/(.+) $ {
        metric hist KEY=$1;
    }
}
```

Обновление метрики:

```
$ curl 127.0.0.1/metric/set/0.25
```

Ожидаемое значение метрики в API:

```
{
  "KEY": {
    "0.1": 0,
    "0.2": 0,
    "0.5": 1,
    "1": 1,
    "2": 1,
    "inf": 1
  }
}
```

Обновление метрики:

```
$ curl 127.0.0.1/metric/set/2
```

Ожидаемое значение метрики в API:

```
{
  "KEY": {
    "0.1": 0,
    "0.2": 0,
    "0.5": 1,
    "1": 1,
    "2": 2,
    "inf": 2
  }
}
```

Обновление метрики:

```
$ curl 127.0.0.1/metric/set/1000
```

Ожидаемое значение метрики в API:

```
{
  "KEY": {
    "0.1": 0,
    "0.2": 0,
    "0.5": 1,
    "1": 1,
    "2": 2,
    "inf": 3
  }
}
```

Встроенные переменные

Для каждой метрики создаются переменные:

- `$metric_<name>`
- `$metric_<name>_key`
- `$metric_<name>_value`

Для составной метрики добавляется переменная:

- `$metric_<name>_value_<metric>`

`$metric_<name>`

Аналогично директиве `metric`, можно использовать сеттер переменной `$metric_<name>` для подсчета метрики. Подсчет метрики будет происходить в фазе `Rewrite`, что позволяет производить обработку метрики, например, из модуля `njs`.

Сеттер принимает ключ с необязательным суффиксом `=значение`. Без суффикса используется значение 0. Ключ и значение могут состоять из текста, переменных или их комбинаций.

Пример использования:

```
http {
    metric_zone counter:1m count;

    # В этот момент добавилась переменная $metric_counter

    server {
        listen 80;

        location /metric/ {
            set $metric_counter $http_user_agent; # Вместо $http_user_agent=0
        }

        location /api/ {
            allow 127.0.0.1;
            deny all;
            api /status/http/metric_zones/counter/;
        }
    }
}
```

Подсчет метрики с помощью модуля `njs`:


```
http {
    js_import metrics.js;

    resolver 127.0.0.53;

    metric_complex_zone requests:1m {
        min_time      min;
        max_time      max;
        total         count;
    }

    location /metric/ {
        js_content metrics.js_request;
        js_fetch_trusted_certificate /path/to/ISRG_Root_X1.pem;
    }

    location /api/ {
        allow 127.0.0.1;
        deny all;
        api /static/http/metric_zones/requests/;
    }
}
```

Файл metrics.js:

```
async function js_request(r) {
    let start_time = Date.now();

    let results = await Promise.all([ngx.fetch('https://google.com/'),
                                     ngx.fetch('https://google.ru/')]);

    // Используем сеттер переменной $metric_requests
    r.variables.metric_requests = `google=${Date.now() - start_time}`;
}

export default {js_request};
```

После нескольких запросов на location /metric/ значения метрики могут быть следующими:

```
{
  "discarded": 0,
  "metrics": {
    "google": {
      "min_time": 70,
      "max_time": 432,
      "total": 6
    }
  }
}
```

i Примечание

После установки переменной можно получить ее значение. Оно будет равно ключу вместе с вновь подсчитанным значением метрики, в формате **ключ=значение**, а не буквальной строке, которая была присвоена.

Также значение, хранящееся в переменной `$metric_<name>_key`, изменится на указанный ключ.

`$metric_<name>_key` и `$metric_<name>_value`

Переменные `$metric_<name>_key` и `$metric_<name>_value` задают соответственно ключ и значение. Обновление метрики происходит в момент установки значения `$metric_<name>_value`, если ключ в `$metric_<name>_key` уже задан.

Примечание

Для составной метрики значения подметрик в переменной `$metric_<name>_value` объединены при помощи разделяющего символа `" , "`.

Пример использования:

```
http {
    metric_zone gauge:1m gauge;

    # В этот момент добавилась переменная $metric_gauge
    # А еще переменные $metric_gauge_key и $metric_gauge_value

    metric_complex_zone complex:1m {
        hist histogram 1 2 3;
        avg average exp;
    }

    # В этот момент добавилась переменная $metric_complex
    # А еще переменные $metric_complex_key и $metric_complex_value

    server {
        listen 80;

        location /gauge/ {
            set $metric_gauge_key "foo";
            set $metric_gauge_value 1;

            # Либо set $metric_gauge foo=1;

            return 200 "Updated with '$metric_gauge'\nValue='$metric_gauge_value'\n";
        }

        location /complex/ {
            set $metric_complex_key "foo";
            set $metric_complex_value 3;

            # Либо set $metric_complex foo=3;

            return 200 "Updated with '$metric_complex'\nValue='$metric_complex_value'\n";
        }
    }
}
```

Для такой конфигурации после запроса к `/gauge/` ожидается следующий ответ:

```
$ curl 127.0.0.1/gauge/
Updated with 'foo=1'
Value='1'
```

Для `/complex/`:

```
$ curl 127.0.0.1/complex/
Updated with 'foo=0 0 1, 3'
Value='0 0 1, 3'
```

Примечание

Если в качестве значения для переменной `$metric_<name>_value` указать пустую строку, значение будет распознано как 0. Если строка состоит из символов, которые невозможно преобразовать в число, она будет распознана как 1.

Подсчет метрики произойдет только после того, как заданы значения переменных `$metric_<name>_key` и `$metric_<name>_value`.

В таком случае значение, сохраненное в `$metric_<name>`, станет равным ключу вместе с вновь подсчитанным значением метрики, в формате **ключ=значение**.

Значение, которое хранится в `$metric_<name>_key`, является последним указанным через переменные ключом метрики.

Значение, которое хранится в `$metric_<name>_value`, является последним подсчитанным значением метрики с ключом, установленным в `$metric_<name>_key`.

`$metric_<name>_value_<metric>`

Для составной метрики значение конкретной подметрики можно получить при помощи переменной `$metric_<name>_value_<metric>`, указав имя подметрики в качестве `<metric>`.

Пример использования:

```
http {
    metric_complex_zone foo:1m {
        count count;
        min    min;
        avg    average exp;
    }

    # В этот момент добавилась переменная $metric_foo
    # А еще переменные $metric_foo_key и $metric_foo_value
    # А также $metric_foo_value_count, $metric_foo_value_min и $metric_foo_value_avg

    server {
        listen 80;

        location /foo/ {
            set $metric_foo_key bar;
            set $metric_foo_value 9;

            # Либо set $metric_foo bar=9;

            return 200 "Updated with '$metric_foo'\nValues='$metric_foo_value'\nCount=
↪'$metric_foo_value_count'\n";
        }
    }
}
```

Для такой конфигурации после запроса к `/foo/` ожидается следующий ответ:

```
$ curl 127.0.0.1/foo/
Updated with 'bar=1, 9, 9'
```

```
Values='1, 9, 9'
Count='1'
```

Дополнительные примеры

Мониторинг HTTP-методов

```
metric_zone http_methods:1m count;

server {
    listen 80;

    location / {
        metric http_methods $request_method;
    }

    location /metrics/ {
        allow 127.0.0.1;
        deny all;
        api /status/http/metric_zones/http_methods/metrics/;
    }
}
```

Ответ:

```
{
  "GET": 65,
  "POST": 20,
  "PUT": 10,
  "DELETE": 5
}
```

Распределение времени ответа upstream

```
metric_zone upstream_time:10m expire=on histogram
    0.05 0.1 0.3 0.5 1 2 5 10 inf;

server {
    listen 80;

    location /backend/ {
        proxy_pass http://backend;
        metric upstream_time $upstream_addr=$upstream_response_time on=end;
    }

    location /metrics/ {
        allow 127.0.0.1;
        deny all;
        api /status/http/metric_zones/upstream_time/;
    }
}
```

Ответ:

```
{
  "discarded": 0,
  "metrics": {
```

```

    "backend1:8080": {
      "0.05": 12,
      "0.1": 28,
      "0.3": 56,
      "0.5": 78,
      "1": 92,
      "2": 97,
      "5": 99,
      "10": 100,
      "inf": 100
    }
  }
}

```

Активные подключения

```

metric_zone active_connections:2m gauge;

server {
    listen 80;
    server_name site1.com;

    location / {
        # Увеличиваем при подключении
        metric active_connections site1=1 on=request;

        # Уменьшаем при завершении
        metric active_connections site1=-1 on=end;
    }
}

server {
    listen 80;
    server_name site2.com;

    location / {
        metric active_connections site2=1 on=request;
        metric active_connections site2=-1 on=end;
    }
}

server {
    listen 8080;

    location /connections/ {
        allow 127.0.0.1;
        deny all;
        api /status/http/metric_zones/active_connections/metrics;
    }
}

```

Ответ:

```

{
  "site1": 42,
  "site2": 17
}

```

Поддержка Prometheus

Angie имеет встроенный модуль Prometheus для отображения метрик в формате Prometheus, поддерживающий произвольные метрики.

В качестве примера интеграции рассмотрим следующую конфигурацию:

```
http {
    # Создание метрики "upload"
    metric_complex_zone upload:1m discard_key="other" {
        stats    histogram 64 256 1024 4096 16384 +Inf;
        sum      gauge;
        count    count;
        avg_size average exp;
    }

    # Описание шаблона Prometheus для метрики "upload"
    prometheus_template upload_metric {
        'stats{le="$1"}' $p8s_value
                        path=~~/http/metric_zones/upload/metrics/angie/stats/(.+)
                        type=histogram;

        'stats_sum'      $p8s_value
                        path=/http/metric_zones/upload/metrics/angie/sum;
        'stats_count'    $p8s_value
                        path=/http/metric_zones/upload/metrics/angie/count;

        'avg_size'       $p8s_value
                        path=/http/metric_zones/upload/metrics/angie/avg_size;
    }

    server {
        listen 80;

        # Обновление метрики
        location ~ ~/upload/(.*)$ {
            api /status/http/metric_zones/upload/metrics/angie/;
            metric upload angie=$1 on=request;
        }

        # Таргет для сбора метрик
        location /prometheus/upload_metric/ {
            prometheus upload_metric;
        }
    }
}
```

После нескольких запросов на /upload/...:

```
$ curl 127.0.0.1/upload/16384
$ curl 127.0.0.1/upload/64448
$ curl 127.0.0.1/upload/64
$ curl 127.0.0.1/upload/1028
$ curl 127.0.0.1/upload/1028
```

Значения метрики будут следующими:

```
{
  "stats": {
```

```

        "64": 1,
        "256": 1,
        "1024": 1,
        "4096": 3,
        "16384": 4,
        "+Inf": 5
    },

    "sum": 82952,
    "count": 5,
    "avg_size": 1077.9376
}

```

В формате Prometheus метрика доступна на `/prometheus/upload_metric/`:

```

# Angie Prometheus template "upload_metric"
# TYPE stats histogram
stats{le="64"} 1
stats{le="256"} 1
stats{le="1024"} 1
stats{le="4096"} 3
stats{le="16384"} 4
stats{le="+Inf"} 5
stats_sum 82952
stats_count 5
avg_size 1077.9376

```

Mirror

Позволяет зеркалировать исходный запрос при помощи создания фоновых зеркалирующих подзапросов. Ответы на зеркалирующие подзапросы игнорируются.

Пример конфигурации

```

location / {
    mirror /mirror;
    proxy_pass http://backend;
}

location = /mirror {
    internal;
    proxy_pass http://test_backend$request_uri;
}

```

Директивы

mirror

Синтаксис	<code>mirror uri off;</code>
По умолчанию	<code>mirror off;</code>
Контекст	<code>http, server, location</code>

Задаёт URI, на который будет зеркалироваться исходный запрос. На одном уровне конфигурации может быть задано несколько зеркал.

mirror_request_body

Синтаксис	<code>mirror_request_body on off;</code>
По умолчанию	<code>mirror_request_body on;</code>
Контекст	<code>http, server, location</code>

Определяет, зеркалировать ли тело запроса клиента. Если включено, то тело запроса клиента будет прочитано перед созданием зеркалирующих подзапросов. В этом случае небуферизованное проксирование тела запроса клиента, задаваемое директивами *proxy_request_buffering*, *fastcgi_request_buffering*, *scgi_request_buffering* и *uwsgi_request_buffering*, будет отключено.

```
location / {
    mirror /mirror;
    mirror_request_body off;
    proxy_pass http://backend;
}

location = /mirror {
    internal;
    proxy_pass http://log_backend;
    proxy_pass_request_body off;
    proxy_set_header Content-Length "";
    proxy_set_header X-Original-URI $request_uri;
}
```

MP4

Обеспечивает серверную поддержку псевдо-стриминга для файлов в формате MP4. Такие файлы обычно имеют расширения `.mp4`, `.m4v` и `.m4a`.

Псевдо-стриминг работает в паре с совместимым медиаплеером. Плеер посылает серверу HTTP-запрос с указанием точки времени старта в аргументе `start` строки запроса (время задается в секундах), а сервер в ответ посылает поток, у которого начальная позиция соответствует запрошенному времени, например:

```
http://example.com/elephants_dream.mp4?start=238.88
```

Это позволяет в любой момент времени выполнить произвольное позиционирование, а также начать воспроизведение с середины временной шкалы.

В форматах, основанных на H.264, метаданные, необходимые для поддержки позиционирования, хранятся в так называемом "моов-атоме". Это часть файла, которая содержит индексную информацию для всего файла.

До начала воспроизведения плееру необходимо прочитать метаданные. Для этого он отправляет специальный запрос с аргументом `start=0`. Многие кодирующие программы добавляют метаданные в конец файла. Это неоптимально для псевдо-стриминга, поскольку плееру потребуется загрузить файл целиком прежде чем начать воспроизведение. Если метаданные находятся в начале файла, Angie достаточно начать отправлять в ответ содержимое файла. Если же метаданные находятся в конце файла, потребуется прочитать весь файл и подготовить новый поток, в котором метаданные предшествуют медийным данным. Это требует дополнительного процессорного времени, памяти и дискового ввода/вывода, поэтому лучше заранее **подготовить** исходный файл для псевдо-стриминга, нежели делать это для каждого запроса.

Модуль также поддерживает аргумент `end` HTTP-запроса, задающий время окончания воспроизведения потока. Аргумент `end` задается совместно с аргументом `start` или самостоятельно:


```
http://example.com/elephants_dream.mp4?start=238.88&end=555.55
```

Для запроса с ненулевыми аргументами **start** или **end** Angie считывает из файла метаданные, готовит поток с запрошенным диапазоном и отправляет его клиенту. Это тоже требует дополнительных ресурсов, как указано выше.

Если аргумент **start** указывает на видеокادر, не являющийся ключевым, то начало такого видео может воспроизводиться с ошибками. В этом случае к запрашиваемому видео могут быть добавлены ближайший к точке **start** ключевой кадр и все промежуточные кадры между ними. При воспроизведении эти кадры будут скрыты при помощи edit-листа.

Если запрос, обрабатываемый этим модулем, не содержит аргументов **start** и **end**, дополнительные ресурсы не тратятся, а файл отсылается непосредственно как статический ресурс. Некоторые плееры также поддерживают запросы с указанием диапазона запрашиваемых байт (byte-range requests), для них этот модуль не требуется.

Схожая поддержка псевдо-стриминга для FLV-файлов обеспечивается модулем FLV.

Пример конфигурации

```
location /video/ {
    mp4;
    mp4_buffer_size      1m;
    mp4_max_buffer_size  5m;
}
```

Директивы

mp4

Синтаксис	mp4;
По умолчанию	—
Контекст	location

Включает в содержащем location обработку этим модулем.

mp4_buffer_size

Синтаксис	mp4_buffer_size размер;
По умолчанию	mp4_buffer_size 512K;
Контекст	http, server, location

Задаёт начальный размер буфера, используемого при обработке MP4-файлов.

mp4_limit_rate

Синтаксис	mp4_limit_rate on off коэффициент;
По умолчанию	mp4_limit_rate off;
Контекст	http, server, location

Ограничивает скорость передачи запрошенного MP4-файла клиенту. Предельное значение вычисляется умножением заданного *коэффициента* на средний битрейт файла.

- Специальное значение `off` отключает ограничение скорости.
- Специальное значение `on` соответствует *коэффициенту 1.1*.
- Начало действия ограничения регулируется значением `mp4_limit_rate_after`.

Запросы ограничиваются индивидуально: если клиент открыл два соединения, общая скорость удваивается. В связи с этим обратите внимание на `limit_conn` и сопутствующие директивы.

`mp4_limit_rate_after`

Синтаксис	<code>mp4_limit_rate_after время;</code>
По умолчанию	<code>mp4_limit_rate_after 60s;</code>
Контекст	<code>http, server, location</code>

Задаёт (в пересчёте на время воспроизведения) объём медиаданных, после передачи которого скорость будет ограничена директивой `mp4_limit_rate`.

`mp4_max_buffer_size`

Синтаксис	<code>mp4_max_buffer_size размер;</code>
По умолчанию	<code>mp4_max_buffer_size 10M;</code>
Контекст	<code>http, server, location</code>

В ходе обработки метаданных может понадобиться буфер большего размера. Его размер не может превышать указанного, иначе Angie вернёт серверную ошибку 500 (Internal Server Error) и запишет в лог следующее сообщение:

```
"/some/movie/file.mp4" mp4 moov atom is too large: 12583268, you may want to increase
mp4_max_buffer_size
```

`mp4_start_key_frame`

Синтаксис	<code>mp4_start_key_frame on off;</code>
По умолчанию	<code>mp4_start_key_frame off;</code>
Контекст	<code>http, server, location</code>

Включает режим, в котором видео всегда начинается с ключевого видеокadra. Если аргумент `start` не указывает на ключевой кадр, то первоначальные кадры будут скрыты при помощи `mp4 edit-lista`. Edit-листы поддерживаются большинством плееров и браузеров включая Chrome, Safari, QuickTime и ffmpeg, частично поддерживаются в Firefox.

Prometheus

Собирает статистику Angie ADC, исходя из определенных в конфигурации шаблонов, и возвращает сформированные на основании этих шаблонов метрики в [формате Prometheus](#).

Предупреждение

Чтобы собирать статистику, включите в нужных контекстах зону разделяемой памяти с помощью:

- директивы `zone` в `http_upstream` или `stream_upstream`;
- директивы `status_zone`;
- параметра `status_zone` в директиве `resolver`.

Пример конфигурации

Три метрики для сбора статистики запросов к серверным зонам разделяемой памяти, объединенные в шаблон `custom` и опубликованные по пути `/p8s`:

```
http {

    prometheus_template custom {
        'angie_http_server_zones_requests_total{zone="$1"}' $p8s_value
        path=~^/http/server_zones/([~/]+)/requests/total$
        type=counter;

        'angie_http_server_zones_requests_processing{zone="$1"}' $p8s_value
        path=~^/http/server_zones/([~/]+)/requests/processing$
        type=gauge;

        'angie_http_server_zones_requests_discarded{zone="$1"}' $p8s_value
        path=~^/http/server_zones/([~/]+)/requests/discarded$
        type=counter;
    }

    # ...

    server {

        listen 80;

        location =/p8s {
            prometheus custom;
        }

        # ...

    }

}
```

В состав Angie ADC входит вспомогательный файл `prometheus_all.conf`, куда включен набор общепотребимых метрик, сведенных в шаблон `all`:

Содержимое файла

```
prometheus_template all {

angie_connections_accepted $p8s_value
    path=/connections/accepted
    type=counter
    'help=The total number of accepted client connections.';

angie_connections_dropped $p8s_value
    path=/connections/dropped
    type=counter
    'help=The total number of dropped client connections.';

}
```

```

angie_connections_active $p8s_value
    path=/connections/active
    type=gauge
    'help=The current number of active client connections.';

angie_connections_idle $p8s_value
    path=/connections/idle
    type=gauge
    'help=The current number of idle client connections.';

'angie_slabs_pages_used{zone="$1"}' $p8s_value
    path=~slabs/([~/]+)/pages/used$
    type=gauge
    'help=The number of currently used memory pages in a slab zone.';

'angie_slabs_pages_free{zone="$1"}' $p8s_value
    path=~slabs/([~/]+)/pages/free$
    type=gauge
    'help=The number of currently free memory pages in a slab zone.';

'angie_slabs_pages_slots_used{zone="$1",size="$2"}' $p8s_value
    path=~slabs/([~/]+)/slots/([~/]+)/used$
    type=gauge
    'help=The number of currently used memory slots of a specific size in a slab zone.
→';

'angie_slabs_pages_slots_free{zone="$1",size="$2"}' $p8s_value
    path=~slabs/([~/]+)/slots/([~/]+)/free$
    type=gauge
    'help=The number of currently free memory slots of a specific size in a slab zone.
→';

'angie_slabs_pages_slots_reqs{zone="$1",size="$2"}' $p8s_value
    path=~slabs/([~/]+)/slots/([~/]+)/reqs$
    type=counter
    'help=The total number of attempts to allocate a memory slot of a specific size,
→in a slab zone.';

'angie_slabs_pages_slots_fails{zone="$1",size="$2"}' $p8s_value
    path=~slabs/([~/]+)/slots/([~/]+)/fails$
    type=counter
    'help=The number of unsuccessful attempts to allocate a memory slot of a specific,
→size in a slab zone.';

'angie_resolvers_queries{zone="$1",type="$2"}' $p8s_value
    path=~resolvers/([~/]+)/queries/([~/]+)/$
    type=counter
    'help=The number of queries of a specific type to resolve in a resolver zone.';

'angie_resolvers_sent{zone="$1",type="$2"}' $p8s_value
    path=~resolvers/([~/]+)/sent/([~/]+)/$
    type=counter
    'help=The number of sent DNS queries of a specific type to resolve in a resolver,
→';

```

```

↪zone.';

'angie_resolvers_responses{zone="$1",status="$2"}' $p8s_value
  path=~~/resolvers/([~/]+)/responses/([~/]+)$
  type=counter
  'help=The number of resolution results with a specific status in a resolver zone.
↪';

'angie_http_server_zones_ssl_handshaked{zone="$1"}' $p8s_value
  path=~~/http/server_zones/([~/]+)/ssl/handshaked$
  type=counter
  'help=The total number of successful SSL handshakes in an HTTP server zone.';

'angie_http_server_zones_ssl_reuses{zone="$1"}' $p8s_value
  path=~~/http/server_zones/([~/]+)/ssl/reuses$
  type=counter
  'help=The total number of session reuses during SSL handshakes in an HTTP server
↪zone.';

'angie_http_server_zones_ssl_timedout{zone="$1"}' $p8s_value
  path=~~/http/server_zones/([~/]+)/ssl/timedout$
  type=counter
  'help=The total number of timed-out SSL handshakes in an HTTP server zone.';

'angie_http_server_zones_ssl_failed{zone="$1"}' $p8s_value
  path=~~/http/server_zones/([~/]+)/ssl/failed$
  type=counter
  'help=The total number of failed SSL handshakes in an HTTP server zone.';

'angie_http_server_zones_requests_total{zone="$1"}' $p8s_value
  path=~~/http/server_zones/([~/]+)/requests/total$
  type=counter
  'help=The total number of client requests received in an HTTP server zone.';

'angie_http_server_zones_requests_processing{zone="$1"}' $p8s_value
  path=~~/http/server_zones/([~/]+)/requests/processing$
  type=gauge
  'help=The number of client requests currently being processed in an HTTP server
↪zone.';

'angie_http_server_zones_requests_discarded{zone="$1"}' $p8s_value
  path=~~/http/server_zones/([~/]+)/requests/discarded$
  type=counter
  'help=The total number of client requests completed in an HTTP server zone
↪without sending a response.';

'angie_http_server_zones_responses{zone="$1",code="$2"}' $p8s_value
  path=~~/http/server_zones/([~/]+)/responses/([~/]+)$
  type=counter
  'help=The number of responses with a specific status in an HTTP server zone.';

'angie_http_server_zones_data_received{zone="$1"}' $p8s_value
  path=~~/http/server_zones/([~/]+)/data/received$

```

```

type=counter
'help=The total number of bytes received from clients in an HTTP server zone.';

'angie_http_server_zones_data_sent{zone="$1"}' $p8s_value
path=~^/http/server_zones/([~/]+)/data/sent$
type=counter
'help=The total number of bytes sent to clients in an HTTP server zone.';

'angie_http_location_zones_requests_total{zone="$1"}' $p8s_value
path=~^/http/location_zones/([~/]+)/requests/total$
type=counter
'help=The total number of client requests in an HTTP location zone.';

'angie_http_location_zones_requests_discarded{zone="$1"}' $p8s_value
path=~^/http/location_zones/([~/]+)/requests/discarded$
type=counter
'help=The total number of client requests completed in an HTTP location zone,
↳ without sending a response.';

'angie_http_location_zones_responses{zone="$1",code="$2"}' $p8s_value
path=~^/http/location_zones/([~/]+)/responses/([~/]+)/$
type=counter
'help=The number of responses with a specific status in an HTTP location zone.';

'angie_http_location_zones_data_received{zone="$1"}' $p8s_value
path=~^/http/location_zones/([~/]+)/data/received$
type=counter
'help=The total number of bytes received from clients in an HTTP location zone.';

'angie_http_location_zones_data_sent{zone="$1"}' $p8s_value
path=~^/http/location_zones/([~/]+)/data/sent$
type=counter
'help=The total number of bytes sent to clients in an HTTP location zone.';

'angie_http_upstreams_peers_backup{upstream="$1",peer="$2"}' $p8st_all_ups_backup
path=~^/http/upstreams/([~/]+)/peers/([~/]+)/backup$
type=gauge
'help=The HTTP upstream peer backup group level.';

'angie_http_upstreams_peers_state{upstream="$1",peer="$2"}' $p8st_all_ups_state
path=~^/http/upstreams/([~/]+)/peers/([~/]+)/state$
type=gauge
'help=The current state of an upstream peer in "HTTP": 1 - up, 2 - down, 3 -
↳ unavailable, 4 - recovering, 5 - unhealthy, 6 - checking, 7 - draining, 8 - busy,
↳ or 9 - idle.';

'angie_http_upstreams_peers_selected_current{upstream="$1",peer="$2"}' $p8s_value
path=~^/http/upstreams/([~/]+)/peers/([~/]+)/selected/current$
type=gauge
'help=The number of requests currently being processed by an upstream peer in
↳ "HTTP".';

```

```
'angie_http_upstreams_peers_selected_total{upstream="$1",peer="$2"}' $p8s_value
  path=~~/http/upstreams/([~/]+)/peers/([~/]+)/selected/total$
  type=counter
  'help=The total number of attempts to use an upstream peer in "HTTP".';

'angie_http_upstreams_peers_responses{upstream="$1",peer="$2",code="$3"}' $p8s_value
  path=~~/http/upstreams/([~/]+)/peers/([~/]+)/responses/([~/]+)/$
  type=counter
  'help=The number of responses with a specific status received from an upstream
↪peer in "HTTP".';

'angie_http_upstreams_peers_data_sent{upstream="$1",peer="$2"}' $p8s_value
  path=~~/http/upstreams/([~/]+)/peers/([~/]+)/data/sent$
  type=counter
  'help=The total number of bytes sent to an upstream peer in "HTTP".';

'angie_http_upstreams_peers_data_received{upstream="$1",peer="$2"}' $p8s_value
  path=~~/http/upstreams/([~/]+)/peers/([~/]+)/data/received$
  type=counter
  'help=The total number of bytes received from an upstream peer in "HTTP".';

'angie_http_upstreams_peers_health_fails{upstream="$1",peer="$2"}' $p8s_value
  path=~~/http/upstreams/([~/]+)/peers/([~/]+)/health/fails$
  type=counter
  'help=The total number of unsuccessful attempts to communicate with an upstream
↪peer in "HTTP".';

'angie_http_upstreams_peers_health_unavailable{upstream="$1",peer="$2"}' $p8s_value
  path=~~/http/upstreams/([~/]+)/peers/([~/]+)/health/unavailable$
  type=counter
  'help=The number of times when an upstream peer in "HTTP" became "unavailable"
↪due to reaching the max_fails limit.';

'angie_http_upstreams_peers_health_downtime{upstream="$1",peer="$2"}' $p8s_value
  path=~~/http/upstreams/([~/]+)/peers/([~/]+)/health/downtime$
  type=counter
  'help=The total time (in milliseconds) that an upstream peer in "HTTP" was
↪"unavailable".';

'angie_http_upstreams_peers_health_header_time{upstream="$1",peer="$2"}' $p8s_value
  path=~~/http/upstreams/([~/]+)/peers/([~/]+)/health/header_time$
  type=gauge
  'help=Average time (in milliseconds) to receive the response headers from an
↪upstream peer in "HTTP".';

'angie_http_upstreams_peers_health_response_time{upstream="$1",peer="$2"}' $p8s_value
  path=~~/http/upstreams/([~/]+)/peers/([~/]+)/health/response_time$
  type=gauge
  'help=Average time (in milliseconds) to receive the complete response from an
↪upstream peer in "HTTP".';

'angie_http_upstreams_peers_health_probes_count{upstream="$1",peer="$2"}' $p8s_value
  path=~~/http/upstreams/([~/]+)/peers/([~/]+)/health/probes/count$
```

```

type=counter
'help=The total number of probes for this peer.';

'angie_http_upstreams_peers_health_probes_fails{upstream="$1",peer="$2"}' $p8s_value
path=~http/upstreams/([~/]+)/peers/([~/]+)/health/probes/fails$
type=counter
'help=The total number of failed probes for this peer.';

'angie_http_upstreams_peers_balance_feedback{upstream="$1",peer="$2"}' $p8s_value
path=~http/upstreams/([~/]+)/peers/([~/]+)/feedback$
type=gauge
'help=The feedback balancer indicator, average value.';

'angie_http_upstreams_peers_balance_bandwidth{upstream="$1",peer="$2"}' $p8s_value
path=~http/upstreams/([~/]+)/peers/([~/]+)/bandwidth$
type=gauge
'help=The least_bandwidth balancer indicator, average value.';

'angie_http_upstreams_keepalive{upstream="$1"}' $p8s_value
path=~http/upstreams/([~/]+)/keepalive$
type=gauge
'help=The number of currently cached keepalive connections for an HTTP upstream.';

'angie_http_upstreams_backup_switch_active{upstream="$1"}' $p8s_value
path=~http/upstreams/([~/]+)/backup_switch/active$
type=gauge
'help=The currently active HTTP upstream servers backup group level.';

'angie_http_upstreams_queue_queued{upstream="$1"}' $p8s_value
path=~http/upstreams/([~/]+)/queue/queued$
type=counter
'help=The total number of queued requests for an HTTP upstream.';

'angie_http_upstreams_queue_waiting{upstream="$1"}' $p8s_value
path=~http/upstreams/([~/]+)/queue/waiting$
type=gauge
'help=The number of requests currently waiting in an HTTP upstream queue.';

'angie_http_upstreams_queue_dropped{upstream="$1"}' $p8s_value
path=~http/upstreams/([~/]+)/queue/dropped$
type=counter
'help=The total number of requests dropped from an HTTP upstream queue because
↳the client had prematurely closed the connection.';

'angie_http_upstreams_queue_timedout{upstream="$1"}' $p8s_value
path=~http/upstreams/([~/]+)/queue/timedout$
type=counter
'help=The total number of requests timed out from an HTTP upstream queue.';

'angie_http_upstreams_queue_overflows{upstream="$1"}' $p8s_value
path=~http/upstreams/([~/]+)/queue/overflows$
type=counter
'help=The total number of requests rejected by an HTTP upstream queue because the
↳size limit had been reached.';

```



```
'angie_http_caches_responses{zone="$1",status="$2"}' $p8s_value
  path=~~/http/caches/([~/]+)/([~/]+)/responses$
  type=counter
  'help=The total number of responses processed in an HTTP cache zone with a
↳specific cache status.';

'angie_http_caches_bytes{zone="$1",status="$2"}' $p8s_value
  path=~~/http/caches/([~/]+)/([~/]+)/bytes$
  type=counter
  'help=The total number of bytes processed in an HTTP cache zone with a specific
↳cache status.';

'angie_http_caches_responses_written{zone="$1",status="$2"}' $p8s_value
  path=~~/http/caches/([~/]+)/([~/]+)/responses_written$
  type=counter
  'help=The total number of responses written to an HTTP cache zone with a specific
↳cache status.';

'angie_http_caches_bytes_written{zone="$1",status="$2"}' $p8s_value
  path=~~/http/caches/([~/]+)/([~/]+)/bytes_written$
  type=counter
  'help=The total number of bytes written to an HTTP cache zone with a specific
↳cache status.';

'angie_http_caches_size{zone="$1"}' $p8s_value
  path=~~/http/caches/([~/]+)/size$
  type=gauge
  'help=The current size (in bytes) of cached responses in an HTTP cache zone.';

'angie_http_caches_shards_size{zone="$1",path="$2"}' $p8s_value
  path=~~/http/caches/([~/]+)/shards/([~/]+)/size$
  type=gauge
  'help=The current size (in bytes) of cached responses in a shard path of an HTTP
↳cache zone.';

'angie_http_limit_conns{zone="$1",status="$2"}' $p8s_value
  path=~~/http/limit_conns/([~/]+)/([~/]+)/$
  type=counter
  'help=The number of requests processed by an HTTP limit_conn zone with a specific
↳result.';

'angie_http_limit_reqs{zone="$1",status="$2"}' $p8s_value
  path=~~/http/limit_reqs/([~/]+)/([~/]+)/$
  type=counter
  'help=The number of requests processed by an HTTP limit_reqs zone with a specific
↳result.';

'angie_http_acme_clients_state{client="$1"}' $p8st_acme_cert_state
  path=~~/http/acme_clients/([~/]+)/state$
  type=gauge
  'help=The current state of an ACME client: 1 - ready, 2 - requesting, 3 -
```

```

↪disabled, or 4 - failed.';

'angie_http_acme_certs_state{client="$1"}' $p8st_acme_cli_state
  path=~/http/acme_clients/([~/]+)/certificate$
  type=gauge
  'help=The current state of an ACME client certificate: 1 - valid, 2 - mismatch, 3↪
↪- expired, 4 - missing, or 5 - error.';

'angie_stream_server_zones_ssl_handshaked{zone="$1"}' $p8s_value
  path=~/stream/server_zones/([~/]+)/ssl/handshaked$
  type=counter
  'help=The total number of successful SSL handshakes in a stream server zone.';

'angie_stream_server_zones_ssl_reuses{zone="$1"}' $p8s_value
  path=~/stream/server_zones/([~/]+)/ssl/reuses$
  type=counter
  'help=The total number of session reuses during SSL handshakes in a stream server↪
↪zone.';

'angie_stream_server_zones_ssl_timedout{zone="$1"}' $p8s_value
  path=~/stream/server_zones/([~/]+)/ssl/timedout$
  type=counter
  'help=The total number of timed-out SSL handshakes in a stream server zone.';

'angie_stream_server_zones_ssl_failed{zone="$1"}' $p8s_value
  path=~/stream/server_zones/([~/]+)/ssl/failed$
  type=counter
  'help=The total number of failed SSL handshakes in a stream server zone.';

'angie_stream_server_zones_connections_total{zone="$1"}' $p8s_value
  path=~/stream/server_zones/([~/]+)/connections/total$
  type=counter
  'help=The total number of client connections received in a stream server zone.';

'angie_stream_server_zones_connections_processing{zone="$1"}' $p8s_value
  path=~/stream/server_zones/([~/]+)/connections/processing$
  type=gauge
  'help=The number of client connections currently being processed in a stream↪
↪server zone.';

'angie_stream_server_zones_connections_discarded{zone="$1"}' $p8s_value
  path=~/stream/server_zones/([~/]+)/connections/discarded$
  type=counter
  'help=The total number of client connections completed in a stream server zone↪
↪without establishing a session.';

'angie_stream_server_zones_connections_passed{zone="$1"}' $p8s_value
  path=~/stream/server_zones/([~/]+)/connections/passed$
  type=counter
  'help=The total number of client connections in a stream server zone passed for↪
↪handling to a different listening socket.';

'angie_stream_server_zones_sessions{zone="$1",status="$2"}' $p8s_value
  path=~/stream/server_zones/([~/]+)/sessions/([~//])$

```

```

    type=counter
    'help=The number of sessions finished with a specific status in a stream server
↪zone.';

'angie_stream_server_zones_data_received{zone="$1"}' $p8s_value
    path=~/.stream/server_zones/([~/]+)/data/received$
    type=counter
    'help=The total number of bytes received from clients in a stream server zone.';

'angie_stream_server_zones_data_sent{zone="$1"}' $p8s_value
    path=~/.stream/server_zones/([~/]+)/data/sent$
    type=counter
    'help=The total number of bytes sent to clients in a stream server zone.';

'angie_stream_upstreams_peers_backup{upstream="$1",peer="$2"}' $p8st_all_ups_backup
    path=~/.stream/upstreams/([~/]+)/peers/([~/]+)/backup$
    type=gauge
    'help=The "stream" upstream peer backup group level.';

'angie_stream_upstreams_peers_state{upstream="$1",peer="$2"}' $p8st_all_ups_state
    path=~/.stream/upstreams/([~/]+)/peers/([~/]+)/state$
    type=gauge
    'help=The current state of an upstream peer in "stream": 1 - up, 2 - down, 3 -
↪unavailable, 4 - recovering, 5 - unhealthy, 6 - checking, 7 - draining, 8 - busy,
↪or 9 - idle.';

'angie_stream_upstreams_peers_selected_current{upstream="$1",peer="$2"}' $p8s_value
    path=~/.stream/upstreams/([~/]+)/peers/([~/]+)/selected/current$
    type=gauge
    'help=The number of sessions currently being processed by an upstream peer in
↪"stream".';

'angie_stream_upstreams_peers_selected_total{upstream="$1",peer="$2"}' $p8s_value
    path=~/.stream/upstreams/([~/]+)/peers/([~/]+)/selected/total$
    type=counter
    'help=The total number of attempts to use an upstream peer in "stream".';

'angie_stream_upstreams_peers_data_sent{upstream="$1",peer="$2"}' $p8s_value
    path=~/.stream/upstreams/([~/]+)/peers/([~/]+)/data/sent$
    type=counter
    'help=The total number of bytes sent to an upstream peer in "stream".';

'angie_stream_upstreams_peers_data_received{upstream="$1",peer="$2"}' $p8s_value
    path=~/.stream/upstreams/([~/]+)/peers/([~/]+)/data/received$
    type=counter
    'help=The total number of bytes received from an upstream peer in "stream".';

'angie_stream_upstreams_peers_data_pkt_sent{upstream="$1",peer="$2"}' $p8s_value
    path=~/.stream/upstreams/([~/]+)/peers/([~/]+)/data/pkt_sent$
    type=counter
    'help=The total number of packets sent to an upstream peer in "stream".';

```

```
'angie_stream_upstreams_peers_data_pkt_received{upstream="$1",peer="$2"}' $p8s_value
  path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/data/pkt_received$
  type=counter
  'help=The total number of packets received from an upstream peer in "stream"';

'angie_stream_upstreams_peers_health_fails{upstream="$1",peer="$2"}' $p8s_value
  path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/health/fails$
  type=counter
  'help=The total number of unsuccessful attempts to communicate with an upstream
↪peer in "stream"';

'angie_stream_upstreams_peers_health_unavailable{upstream="$1",peer="$2"}' $p8s_value
  path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/health/unavailable$
  type=counter
  'help=The number of times when an upstream peer in "stream" became "unavailable"
↪due to reaching the max_fails limit.';

'angie_stream_upstreams_peers_health_downtime{upstream="$1",peer="$2"}' $p8s_value
  path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/health/downtime$
  type=counter
  'help=The total time (in milliseconds) that an upstream peer in "stream" was
↪"unavailable"';

'angie_stream_upstreams_peers_health_connect_time{upstream="$1",peer="$2"}' $p8s_value
  path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/health/connect_time$
  type=gauge
  'help=Average time (in milliseconds) to connect to an upstream peer in "stream"';

'angie_stream_upstreams_peers_health_first_byte_time{upstream="$1",peer="$2"}' $p8s_
↪value
  path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/health/first_byte_time$
  type=gauge
  'help=Average time (in milliseconds) to receive the first byte from an upstream
↪peer in "stream"';

'angie_stream_upstreams_peers_health_last_byte_time{upstream="$1",peer="$2"}' $p8s_
↪value
  path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/health/last_byte_time$
  type=gauge
  'help=Average time (in milliseconds) of the whole communication session with an
↪upstream peer in "stream"';

'angie_stream_upstreams_peers_health_probes_count{upstream="$1",peer="$2"}' $p8s_value
  path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/health/probes/count$
  type=counter
  'help=The total number of probes for this peer.';

'angie_stream_upstreams_peers_health_probes_fails{upstream="$1",peer="$2"}' $p8s_value
  path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/health/probes/fails$
  type=counter
  'help=The total number of failed probes for this peer.';

'angie_stream_upstreams_peers_balance_feedback{upstream="$1",peer="$2"}' $p8s_value
  path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/feedback$
  type=gauge
```

```
'help=The feedback balancer indicator, average value.';

'angie_stream_upstreams_peers_balance_bandwidth{upstream="$1",peer="$2"}' $p8s_value
path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/bandwidth$
type=gauge
'help=The least_bandwidth balancer indicator, average value.';

'angie_stream_upstreams_peers_balance_packets{upstream="$1",peer="$2"}' $p8s_value
path=~~/stream/upstreams/([~/]+)/peers/([~/]+)/packets$
type=gauge
'help=The least_packets balancer indicator, average value.';

'angie_stream_upstreams_backup_switch_active{upstream="$1"}' $p8s_value
path=~~/stream/upstreams/([~/]+)/backup_switch/active$
type=gauge
'help=The currently active "stream" upstream servers backup group level.';
}

map $p8s_value $p8st_all_ups_state {
    volatile;
    "up"          1;
    "down"        2;
    "unavailable" 3;
    "recovering"  4;
    "unhealthy"   5;
    "checking"    6;
    "draining"    7;
    "busy"        8;
    "idle"        9;
    default       0;
}

map $p8s_value $p8st_all_ups_backup {
    volatile;
    "false"       0;
    "true"        1;
    default       $p8s_value;
}

map $p8s_value $p8st_acme_cli_state {
    volatile;
    "ready"       1;
    "requesting"  2;
    "disabled"    3;
    "failed"      4;
}

map $p8s_value $p8st_acme_cert_state {
    volatile;
    "valid"       1;
    "mismatch"    2;
    "expired"     3;
    "missing"     4;
    "error"       5;
}
```

```
}
```

Его использование:

```
http {
    include prometheus_all.conf;

    # ...

    server {

        listen 80;

        location =/p8s {
            prometheus all;
        }

        # ...

    }
}
```

```
$ curl localhost/p8s

# Angie Prometheus template "all"
...
```

Директивы

prometheus

Синтаксис	<code>prometheus имя_шаблона;</code>
По умолчанию	—
Контекст	location

Указывает для контекста `location` шаблон-обработчик, заданный директивой `prometheus_template`. При запросе этот `location` вычисляет и возвращает метрики шаблона в формате Prometheus.

```
location =/p8s {
    prometheus custom;
}
```

```
$ curl localhost/p8s

# Angie Prometheus template "custom"
...
```

prometheus_template

Синтаксис	<code>prometheus_template имя_шаблона { ... }</code>
По умолчанию	—
Контекст	http

Определяет именованный шаблон метрик, собираемых и экспортируемых Angie, для использования с директивой *prometheus*.

Примечание

В состав Angie также входит готовый шаблон *all*, куда включен набор наиболее общеупотребимых метрик.

Может содержать произвольное число определений метрик, каждая из которых имеет следующую структуру: `<имя_метрики> <переменная> [path=<строка_сопоставления>] [type=<тип>] [help=<справка>]`.

имя_метрики	Задаёт имя метрики, под которым она будет добавлена в формате Prometheus в ответ на запрос. Может содержать необязательную часть с метками (...), например: <pre>http_requests_total{method="\$1",code="\$2"}</pre> В значениях меток можно использовать переменные Angie; если <i>строка_сопоставления</i> задана регулярным выражением, то можно также использовать определенные в этом выражении группы захвата. Такие переменные и группы вычисляются при получении значения метрики, которое задает <i>переменная</i>.
переменная	Задаёт имя переменной, которая будет вычислена и добавлена как значение метрики в ответ на запрос. Если переменной нет или результат вычисления пуст (""), метрика не добавляется.

Метрика вычисляется со значением, которое задает *переменная*; при успехе вычисления метрика добавляется в ответ, например:

```
'angie_time{version="$angie_version"}' $msec;
```

```
$ curl localhost/p8s
```

```
angie_time{version="1.1.1"} 1695119820.562
```

path=строка_соп	Сопоставляется со всеми конечными путями метрик в поддереве <i>/status API</i> -интерфейса Angie, позволяя добавить в ответ сразу несколько экземпляров метрики.
-----------------	--

При сопоставлении пути берутся с начальной косой чертой, но без конечной, например */angie/generation*; при этом регистр символов не учитывается. Есть два способа сопоставления:

path=точное_соот	Проверяется посимвольным сравнением.
path=~регулярное	Проверяется при помощи библиотеки PCRE; может задавать группы захвата для использования в метках поля <i>имя метрики</i> .

Если *строка_сопоставления* соответствует какому-либо пути, то значение метрики Angie по нему заносится в переменную `$p8s_value`, которую при заданном `path=` можно использовать в поле *переменная*.

Если *строка_сопоставления* заканчивается конечной косой чертой, значением метрики будет количество элементов соответствующего списка или объекта. Например:

```
'angie_http_server_zones_count' $p8s_value
  path=/http/server_zones/;
```

В случае регулярного выражения совпадающих путей может быть несколько; метрика добавляется в ответ для *каждого* совпадения. В сочетании с группами захвата это позволяет получить ряд метрик с одним именем и разными метками, например:

```
'angie_slabs_slots_free{zone="$1",size="$2"}' $p8s_value
  path=~^/slabs/([~/]+)/slots/([~/]+)/free$;
```

Это определение добавляет метрики для всех зон и всех размеров, которые есть сейчас в конфигурации:

```
angie_slabs_slots_free{zone="one",size="8"} 502
angie_slabs_slots_free{zone="one",size="16"} 249
angie_slabs_slots_free{zone="one",size="32"} 122
angie_slabs_slots_free{zone="one",size="128"} 22
angie_slabs_slots_free{zone="one",size="512"} 4
angie_slabs_slots_free{zone="two",size="8"} 311
...
```

Если совпадений нет (при любом способе сопоставления), то метрика не добавляется.

Примечание

Параметр `path=` доступен только при сборке Angie с модулем API.

<code>type=тип,</code> <code>help=справка</code>	Задают соответственно тип и справочную строку метрики в формате Prometheus , которые добавляются с ней в ответ без изменений и проверок.
---	--

Встроенные переменные

В модуле `http_prometheus` есть встроенная переменная, получающая значение при сопоставлении путей метрик из раздела `/status` API-интерфейса Angie с параметром *строка_сопоставления* метрик, заданных директивой `prometheus_template`.

`$p8s_value`

Если *строка_сопоставления* метрики, заданной в `prometheus_template`, соответствует какому-либо пути, то значение метрики Angie, находящееся по этому пути, заносится в переменную `$p8s_value`. Она предназначена для использования в поле *переменная* в определениях метрик, которые вычисляются на основе параметра `path=`.

Значения метрик Angie, заносимые в переменную `$p8s_value`, не всегда отвечают потребностям формата Prometheus. Тогда можно воспользоваться директивой `map`, например для преобразования строк в числа:

```
map $p8s_value $ups_state_n {
  up          0;
  unavailable 1;
```



```

    down      2;
    default   3;
}

prometheus_template main {
    'angie_http_upstreams_state{upstream="$1",peer="$2"}' $ups_state_n
    path=~^/http/upstreams/([~/]+)/peers/([~/]+)/state$;
}

```

Если у метрики Angie логическое значение, то есть **true** или **false**, переменная получает значение "1" или "0" соответственно; если значение метрики — **null**, то переменная будет равна "(null)". Для дат используется целочисленный формат эпохи UNIX.

Proxy

Позволяет передавать запросы другому (проксируемому) серверу.

Пример конфигурации

```

location / {
    proxy_pass      http://localhost:8000;
    proxy_set_header Host      $host;
    proxy_set_header X-Real-IP $remote_addr;

    proxy_cache      cache_zone;
    proxy_cache_valid 200 302 10m;
    proxy_cache_valid 404      1m;
}

```

Директивы

proxy_bind

Синтаксис	<code>proxy_bind <i>адрес</i> [transparent] off;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт локальный IP-адрес с необязательным портом, который будет использоваться в исходящих соединениях с проксируемым сервером. В значении параметра можно использовать переменные. Специальное значение **off** отменяет действие унаследованной с предыдущего уровня конфигурации директивы *proxy_bind*, позволяя системе самостоятельно выбирать локальный IP-адрес и порт.

По умолчанию это значение вычисляется один раз при инициализации запроса, до выбора проксируемого сервера. В Angie PRO вычисление происходит после каждого выбора проксируемого сервера (в том числе при повторных попытках), поэтому адрес привязки может зависеть от выбранного сервера; например, с помощью переменной `$upstream_current_addr`. Если вычисленное значение пусто, привязка не выполняется.

Параметр **transparent** позволяет задать нелокальный IP-адрес, который будет использоваться в исходящих соединениях с проксируемым сервером, например, реальный IP-адрес клиента:

```
proxy_bind $remote_addr transparent;
```

Для работы параметра обычно требуется запустить рабочие процессы Angie с привилегиями суперпользователя. В Linux это не требуется, так как если указан параметр *transparent*, то рабочие процессы наследуют *capability CAP_NET_RAW* из главного процесса.

i Примечание

Необходимо настроить таблицу маршрутизации ядра для перехвата сетевого трафика с проксируемого сервера.

proxy_buffer_size

Синтаксис	<code>proxy_buffer_size размер;</code>
По умолчанию	<code>proxy_buffer_size 4k 8k;</code>
Контекст	<code>http, server, location</code>

Задаёт размер буфера, в который будет читаться первая часть ответа, получаемого от проксируемого сервера. В этой части ответа обычно находится небольшой заголовок ответа. По умолчанию размер одного буфера равен размеру страницы памяти. В зависимости от платформы это или 4К, или 8К, однако его можно сделать меньше.

proxy_buffering

Синтаксис	<code>proxy_buffering on off;</code>
По умолчанию	<code>proxy_buffering on;</code>
Контекст	<code>http, server, location</code>

Разрешает или запрещает буферизацию ответов от проксируемого сервера.

on	Angie отправляет ответ клиенту параллельно с чтением ответа от проксируемого сервера; клиент не ждет ответа целиком. Ответ читается в буферы, заданные директивами <code>proxy_buffer_size</code> и <code>proxy_buffers</code> . Буфер передается на отправку, как только заполнится целиком, причем суммарный размер буферов, занятых отправкой, ограничен значением <code>proxy_busy_buffers_size</code> . Заполненный частично буфер ждет, пока не заполнится или пока не закончится ответ, поэтому данные обычно попадают к клиенту целыми буферами. Часть ответа записывается на диск во временный файл, когда все буферы заполнены, а клиент читает медленнее, чем отдает проксируемый сервер. Ответы, которые кэшируются или сохраняются на диске, всегда проходят через временный файл. Запись во временные файлы контролируется директивами <code>proxy_max_temp_file_size</code> и <code>proxy_temp_file_write_size</code> . При буферизации проксируемый сервер освобождается сразу, как только отдаст ответ целиком: соединение с ним не удерживается все то время, пока ответ читает медленный клиент.
off	Каждая порция данных, прочитанная от проксируемого сервера, сразу же передается клиенту, без какого-либо минимального размера: прочитанные 10 байт из буфера 4К отправляются немедленно. Angie использует один буфер, заданный директивой <code>proxy_buffer_size</code> , поэтому за один раз он читает не больше этого размера. Временный файл не используется, поэтому ответы не кэшируются и не сохраняются на диске, а <code>proxy_limit_rate</code> и <code>limit_rate</code> не действуют. Режим <code>off</code> подходит для ответов, которые должны доходить до клиента по мере их формирования: Server-Sent Events, длинные опросы (long polling), потоковая передача.

Буферизация может быть также включена или выключена путем передачи значения "yes" или "no" в поле X-Accel-Buffering заголовка ответа. Эту возможность можно запретить с помощью

директивы *proxy_ignore_headers*.

proxy_buffers

Синтаксис	<code>proxy_buffers число размер;</code>
По умолчанию	<code>proxy_buffers 8 4k 8k;</code>
Контекст	<code>http, server, location</code>

Задаёт число и размер буферов для одного соединения, в которые будет читаться ответ, получаемый от проксируемого сервера.

По умолчанию размер одного буфера равен размеру страницы. В зависимости от платформы это или 4K, или 8K.

proxy_busy_buffers_size

Синтаксис	<code>proxy_busy_buffers_size размер;</code>
По умолчанию	<code>proxy_busy_buffers_size 8k 16k;</code>
Контекст	<code>http, server, location</code>

При включенной буферизации ответов проксируемого сервера, ограничивает суммарный размер буферов, которые могут быть заняты для отправки ответа клиенту, пока ответ еще не прочитан целиком. Оставшиеся буферы тем временем могут использоваться для чтения ответа и, при необходимости, буферизации части ответа во временный файл.

По умолчанию размер ограничен величиной двух буферов, заданных директивами *proxy_buffer_size* и *proxy_buffers*.

proxy_cache

Синтаксис	<code>proxy_cache зона off [path=путь];</code>
По умолчанию	<code>proxy_cache off;</code>
Контекст	<code>http, server, location</code>

Задаёт зону разделяемой памяти для кэширования. Одну и ту же зону можно использовать в нескольких местах конфигурации. В значении параметра можно использовать переменные.

off	запрещает кэширование, унаследованное с предыдущего уровня конфигурации.
------------	--

В Angie PRO можно указать несколько директив *proxy_cache_path*, использующих одно и то же значение **keys_zone**, чтобы включить *шардинг кэша*. При этом следует задать параметр **path** директивы *proxy_cache*, использующей это значение **keys_zone**:

path=путь	Значение вычисляется в момент <i>кэширования</i> ответа от бэкенда и предполагает использование переменных, в том числе содержащих информацию из ответа. Если ответ берется из кэша, то <i>путь</i> не вычисляется заново; таким образом, кэшированный ответ сохраняет изначальный <i>путь</i> , пока не будет удален из кэша.
------------------	---

Это позволяет выбирать нужный путь кэша, применяя директивы *map* или скрипты к ответам от бэкенда. Пример для Content-Type:

```
proxy_cache_path /cache/one keys_zone=zone:10m;
proxy_cache_path /cache/two keys_zone=zone;

map $upstream_http_content_type $cache {
    ~~text/ one;
    default two;
}

server {
    ...
    location / {
        proxy_pass http://backend;
        proxy_cache zone path=/cache/$cache;
        proxy_cache_valid 200 10m;
    }
}
```

Здесь есть два пути кэша и отображение переменной, чтобы их различать. Если Content-Type начинается с *text/*, будет выбран первый путь, иначе — второй.

Примечание

При использовании *proxy_cache* обычно необходимо также задать директиву *proxy_cache valid*, чтобы явно указать время действия кэшированных ответов. Если она не задана, Angie не использует значения по умолчанию, а определяет время хранения ответа в кэше на основе HTTP-заголовков ответа от сервера в следующем порядке приоритета:

1. Заголовок X-Accel-Expires (наивысший приоритет).
2. Заголовок Cache-Control с параметрами *max-age* или *s-maxage*.
3. Заголовок Expires.

Если ни один заголовок не содержит допустимого значения или их нет вовсе, ответ не будет кэшироваться, так как определить срок его действия нельзя.

proxy_cache_background_update

Синтаксис	<code>proxy_cache_background_update on off;</code>
По умолчанию	<code>proxy_cache_background_update off;</code>
Контекст	<code>http, server, location</code>

Позволяет запустить фоновый подзапрос для обновления просроченного элемента кэша, в то время как клиенту возвращается устаревший кэшированный ответ.

Предупреждение

Использование устаревшего кэшированного ответа в момент его обновления должно быть разрешено.

proxy_cache_bypass

Синтаксис	<code>proxy_cache_bypass ...;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт условия, при которых ответ не будет браться из кэша. Если значение хотя бы одного из строковых параметров непустое и не равно "0", то ответ не берётся из кэша:

```
proxy_cache_bypass $cookie_nocache $arg_nocache$arg_comment;
proxy_cache_bypass $http_pragma $http_authorization;
```

В значении параметров можно использовать переменные.

Можно использовать совместно с директивой `proxy_no_cache`.

proxy_cache_convert_head

Синтаксис	<code>proxy_cache_convert_head on off;</code>
По умолчанию	<code>proxy_cache_convert_head on;</code>
Контекст	http, server, location

Разрешает или запрещает преобразование метода "HEAD" в "GET" для кэширования. Если преобразование выключено, то необходимо, чтобы ключ кэширования включал в себя `$request_method`.

proxy_cache_key

Синтаксис	<code>proxy_cache_key строка;</code>
По умолчанию	<code>proxy_cache_key \$scheme\$proxy_host\$request_uri;</code>
Контекст	http, server, location

Задаёт ключ для кэширования. В значении ключа можно использовать переменные. Например,

```
proxy_cache_key "$host$request_uri $cookie_user";
```

По умолчанию значение директивы близко к строке

```
proxy_cache_key $scheme$proxy_host$uri$is_args$args;
```

proxy_cache_lock

Синтаксис	<code>proxy_cache_lock on off;</code>
По умолчанию	<code>proxy_cache_lock off;</code>
Контекст	http, server, location

Если включено, одновременно только одному запросу будет позволено заполнить новый элемент кэша, идентифицируемый согласно директиве `proxy_cache_key`, передав запрос на проксируемый сервер. Остальные запросы этого же элемента будут либо ожидать появления ответа в кэше, либо освобождения блокировки этого элемента, в течение времени, заданного директивой `proxy_cache_lock_timeout`.

proxy_cache_lock_age

Синтаксис	proxy_cache_lock_age время;
По умолчанию	proxy_cache_lock_age 5s;
Контекст	http, server, location

Если последний запрос, переданный на проксируемый сервер для заполнения нового элемента кэша, не завершился за указанное время, на проксируемый сервер может быть передан еще один запрос.

proxy_cache_lock_timeout

Синтаксис	proxy_cache_lock_timeout время;
По умолчанию	proxy_cache_lock_timeout 5s;
Контекст	http, server, location

Задаёт таймаут для *proxy_cache_lock*. По истечении указанного времени запрос будет передан на проксируемый сервер, однако ответ не будет кэширован.

proxy_cache_max_range_offset

Синтаксис	proxy_cache_max_range_offset <i>число</i> ;
По умолчанию	—
Контекст	http, server, location

Задаёт смещение в байтах для запросов с указанием диапазона запрашиваемых байт (byte-range requests). Если диапазон находится за указанным смещением, range-запрос будет передан на проксируемый сервер и ответ не будет кэширован.

proxy_cache_methods

Синтаксис	proxy_cache_methods GET HEAD POST ...;
По умолчанию	proxy_cache_methods GET HEAD;
Контекст	http, server, location

Если метод запроса клиента указан в этой директиве, то ответ будет кэширован. Методы "GET" и "HEAD" всегда добавляются в список, но тем не менее рекомендуется перечислять их явно. См. также директиву *proxy_no_cache*.

proxy_cache_min_uses

Синтаксис	proxy_cache_min_uses <i>число</i> ;
По умолчанию	proxy_cache_min_uses 1;
Контекст	http, server, location

Задаёт число запросов, после которого ответ будет кэширован.

⚠ Предупреждение

Метаданные кэша хранятся в разделяемой памяти. Ручное удаление файлов кэша не сбрасывает счетчики и может привести к непредсказуемому поведению. Для полного сброса остановите сервер, удалите директорию кэша и запустите снова.

ℹ Примечание

Сторонние модули очистки кэша (например, Cache Purge) удаляют только файлы, но не сбрасывают счетчик `proxy_cache_min_uses`. Директива предназначена для защиты кэша от загрязнения редкими запросами, и сброс счетчика при очистке может негативно повлиять на производительность.

proxy_cache_path

Синтаксис	<code>proxy_cache_path <i>путь</i> [levels=<i>уровни</i>] [use_temp_path=on off] [keys_zone=<i>имя:размер</i>[:file=<i>файл</i>] [inactive=<i>время</i>] [max_size=<i>размер</i>] [min_free=<i>размер</i>] [manager_files=<i>число</i>] [manager_sleep=<i>время</i>] [manager_threshold=<i>время</i>] [loader_files=<i>число</i>] [loader_sleep=<i>время</i>] [loader_threshold=<i>время</i>];</code>
По умолчанию	—
Контекст	http

Задаёт *путь* и другие параметры кэша. Данные кэша хранятся в файлах. Именем файла в кэше является результат функции MD5 от ключа кэширования.

levels	задаёт уровни иерархии кэша: можно задать от 1 до 3 уровней, на каждом уровне допускаются значения 1 или 2.
---------------	---

Например, при использовании

```
proxy_cache_path /data/angie/cache levels=1:2 keys_zone=one:10m;
```

имена файлов в кэше будут такого вида:

```
/data/angie/cache/c/29/b7f54b2df7773722d382f4809d65029c
```

Кэшируемый ответ сначала записывается во временный файл, а потом этот файл переименовывается. Временные файлы и кэш могут располагаться на разных файловых системах. Однако нужно учитывать, что в этом случае вместо дешевой операции переименовывания в пределах одной файловой системы файл копируется с одной файловой системы на другую. Поэтому лучше, если кэш будет находиться на той же файловой системе, что и каталог с временными файлами.

<code>use_temp_path=on</code>	определяет каталог, который будет использоваться для временных файлов
<code> off</code>	
<code>on</code>	Если параметр не задан или установлен в значение <code>on</code> , будет использоваться каталог, задаваемый директивой <code>proxy_temp_path</code> для данного <code>location</code>
<code>off</code>	временные файлы будут располагаться непосредственно в каталоге кэша
<code>keys_zone</code>	Задаёт имя и размер зоны разделяемой памяти для хранения всех активных ключей и информации о данных. Метаданные кэша хранятся в разделяемой памяти. Размер зоны задается в байтах и должен составлять не менее двух системных страниц памяти. Также допустимы суффиксы <code>k</code> и <code>m</code>; суффикс <code>g</code>, в отличие от <code>max_size</code>, здесь не поддерживается. Зоны размером в 1 мегабайт достаточно для хранения около 8000 ключей. Когда с <code>keys_zone</code> задан необязательный <i>файл</i>, Angie сбрасывает содержимое этой зоны на диск при завершении работы основного процесса и пытается восстановить ее по тому же адресу памяти при следующем запуске или после обновления бинарных файлов, чтобы обеспечить более надежную сохранность данных и сократить время загрузки кэша. Если восстановление области невозможно из-за изменения ее размера, несовместимости версий бинарных файлов или по другим причинам, Angie запишет в журнал предупреждение (<code>failed to restore zone at address</code>) и не будет использовать механизм восстановления зоны. Вместо этого несовместимый файл будет переименован в <code>.old</code>; вы можете либо удалить его, либо восстановить его имя и вернуть Angie к конфигурации и версии, в которых этот файл был изначально создан.
⚠ Предупреждение Убедитесь, что путь к <i>файлу</i> указан правильно и имеет необходимые права доступа для использования Angie, а также защищен от несанкционированного доступа; относительные пути основаны на префиксе.	
<code>inactive</code>	Если к данным кэша не обращаются в течение времени, заданного этим параметром, они удаляются независимо от их свежести; каждое обращение сбрасывает этот таймер. Это отличается от вытеснения по размеру, выполняемого менеджером кэша (см. ниже): удаление по неактивности происходит, даже когда кэш не достиг <code>max_size</code>, и не зависит от сроков кэширования, заданных директивой <code>proxy_cache_valid</code>. По умолчанию <code>inactive</code> равен 10 минутам.

i Примечание

В Angie PRO можно задавать несколько директив `proxy_cache_path` с одним и тем же значением `keys_zone`. Размер зоны разделяемой памяти можно указывать только в первой из них. Выбор между директивами будет производиться на основе параметра `path` соответствующей директивы `proxy_cache`.

Специальный процесс **менеджера кэша** следит за максимальным размером кэша, а также за минимальным объемом свободного места на файловой системе с кэшем, и удаляет наименее востребованные данные при превышении максимального размера кэша или недостаточном объеме свободного места. Удаление данных происходит итерациями.

max_size	максимальное пороговое значение размера кэша Значение задается в байтах; допустимы суффиксы k , m и g . По умолчанию размер кэша не ограничен.
min_free	минимальное пороговое значение объема свободного места на файловой системе с кэшем Задается в тех же единицах, что и max_size . По умолчанию 0 — отслеживание свободного места отключено. На платформах, где статистика файловой системы недоступна, параметр игнорируется, а Angie записывает в журнал предупреждение min_free is not supported on this platform, ignored .
manager_files	максимальное количество удаляемых элементов кэша за одну итерацию По умолчанию: 100.
manager_threshold	ограничивает время работы одной итерации По умолчанию: 200 миллисекунд.
manager_sleep	время, в течение которого выдерживается пауза между итерациями По умолчанию: 50 миллисекунд.

Через минуту после запуска Angie активируется специальный процесс **загрузчика кэша**. Он сканирует файловую систему в поисках ранее закешированных данных и загружает эту информацию в область кэша. Этот процесс выполняется итеративно; каждая итерация обрабатывает ограниченное количество элементов, заданное параметром **loader_files**, следит за тем, чтобы не превышать **loader_threshold**, затем делает короткую паузу, заданную **loader_sleep**, перед переходом к следующей партии. Итерации продолжаются до тех пор, пока загрузчик не обработает все существующие записи кэша на диске:

loader_files	максимальное количество элементов кэша к загрузке в одну итерацию По умолчанию: 100
loader_threshold	ограничивает время работы одной итерации По умолчанию: 200 миллисекунд
loader_sleep	время, в течение которого выдерживается пауза между итерациями По умолчанию: 50 миллисекунд

Примечание

Указание *файла* для параметра **keys_zone** не влияет на работу загрузчика кэша.

proxy_cache_revalidate

Синтаксис	proxy_cache_revalidate on off;
По умолчанию	proxy_cache_revalidate off;
Контекст	http, server, location

Разрешает ревалидацию просроченных элементов кэша при помощи условных запросов с полями заголовка **If-Modified-Since** и **If-None-Match**.

proxy_cache_use_stale

Синтаксис	<code>proxy_cache_use_stale error timeout invalid_header updating http_500 http_502 http_503 http_504 http_403 http_404 http_429 non_idempotent off ...;</code>
По умолчанию	<code>proxy_cache_use_stale off;</code>
Контекст	<code>http, server, location</code>

Определяет, в каких случаях можно использовать устаревший кэшированный ответ. Параметры директивы совпадают с параметрами директивы `proxy_next_upstream`.

error	Позволяет использовать устаревший кэшированный ответ при невозможности выбора проксируемого сервера для обработки запроса.
updating	Дополнительный параметр, разрешает использовать устаревший кэшированный ответ, если на данный момент он уже обновляется. Это позволяет минимизировать число обращений к проксируемым серверам при обновлении кэшированных данных.
non_idempotent	Позволяет использовать устаревший кэшированный ответ, если запрос с неидемпотентным методом уже был отправлен проксируемому серверу, а при дальнейшей работе с этим сервером произошла любая ошибка; из неидемпотентных методов кэшировать можно только POST, и он должен быть разрешен директивой <code>proxy_cache_methods</code> .

Использование устаревшего кэшированного ответа может также быть разрешено непосредственно в заголовке ответа на определенное количество секунд после того, как ответ устарел:

- Расширение **stale-while-revalidate** поля заголовка **Cache-Control** разрешает использовать устаревший кэшированный ответ, если на данный момент он уже обновляется.
- Расширение **stale-if-error** поля заголовка **Cache-Control** разрешает использовать устаревший кэшированный ответ в случае ошибки.

Примечание

Такой способ менее приоритетен, чем задание параметров директивы.

Чтобы минимизировать число обращений к проксируемым серверам при заполнении нового элемента кэша, можно воспользоваться директивой `proxy_cache_lock`.

proxy_cache_valid

Синтаксис	<code>proxy_cache_valid [код ...] время;</code>
По умолчанию	—
Контекст	<code>http, server, location</code>

Задаёт время кэширования для разных кодов ответа. Например, директивы

```
proxy_cache_valid 200 302 10m;
proxy_cache_valid 404 1m;
```

задают время кэширования 10 минут для ответов с кодами 200 и 302 и 1 минуту для ответов с кодом 404.

Если указано только время кэширования,

```
proxy_cache_valid 5m;
```

то кэшируются только ответы 200, 301 и 302.

Кроме того, можно кэшировать любые ответы с помощью параметра `any`:

```
proxy_cache_valid 200 302 10m;
proxy_cache_valid 301      1h;
proxy_cache_valid any      1m;
```

Примечание

Параметры кэширования могут также быть заданы непосредственно в заголовке ответа. Такой способ приоритетнее, чем задание времени кэширования с помощью директивы.

- Поле заголовка **X-Accel-Expires** задает время кэширования ответа в секундах. Значение *0* запрещает кэшировать ответ. Если значение начинается с префикса *@*, оно задает абсолютное время в секундах с начала эпохи, до которого ответ может быть кэширован.
- Если в заголовке нет поля **X-Accel-Expires**, параметры кэширования определяются по полям заголовка **Expires** или **Cache-Control**.
- Ответ, в заголовке которого есть поле **Set-Cookie**, не будет кэшироваться.
- Ответ, в заголовке которого есть поле **Vary** со специальным значением *"*"*, не будет кэшироваться. Ответ, в заголовке которого есть поле **Vary** с другим значением, будет кэширован с учетом соответствующих полей заголовка запроса.

Обработка одного или более из этих полей заголовка может быть отключена при помощи директивы *proxy_ignore_headers*.

proxy_connect_timeout

Синтаксис	<code>proxy_connect_timeout время;</code>
По умолчанию	<code>proxy_connect_timeout 60s;</code>
Контекст	<code>http, server, location</code>

Задаёт таймаут для установления соединения с проксированным сервером. Необходимо иметь в виду, что этот таймаут обычно не может превышать 75 секунд.

Примечание

Этот таймаут не управляет неактивными соединениями: время, в течение которого уже открытое соединение сохраняется для повторного использования, задается директивой *keepalive_timeout* в блоке `upstream`, соединения которого кэшируются директивой *keepalive*.

proxy_connection_drop

Синтаксис	<code>proxy_connection_drop время on off;</code>
По умолчанию	<code>proxy_connection_drop off;</code>
Контекст	<code>http, server, location</code>

Настраивает завершение всех активных соединений с проксируемым сервером, если он был удален из группы или помечен как постоянно недоступный в результате процесса `eresolve` или команды `API DELETE`. Неактивные соединения в кэше не затрагиваются.

Соединение завершается, когда обрабатывается следующее событие чтения или записи для клиента или проксируемого сервера.

Установка *времени* включает таймаут до завершения соединения; при выборе значения `on` соединения завершаются немедленно.

proxy_cookie_domain

Синтаксис	<code>proxy_cookie_domain off;</code> <code>proxy_cookie_domain домен замена;</code>
По умолчанию	<code>proxy_cookie_domain off;</code>
Контекст	<code>http, server, location</code>

Задаёт текст, который нужно изменить в атрибуте `domain` полей **Set-Cookie** заголовка ответа проксируемого сервера. Предположим, проксируемый сервер вернул поле заголовка **Set-Cookie** с атрибутом `"domain=localhost"`. Директива

```
proxy_cookie_domain localhost example.org;
```

перепишет данный атрибут в виде `"domain=example.org"`.

Точка в начале строк *домен* и *замена*, а равно как и в атрибуте *domain* игнорируется. Регистр значения не имеет.

В строках *домен* и *замена* можно использовать переменные:

```
proxy_cookie_domain www.$host $host;
```

Директиву также можно задать при помощи регулярных выражений. При этом *домен* должен начинаться с символа `"~"`. Регулярное выражение может содержать именованные и позиционные группы захвата, а *замена* ссылаться на них:

```
proxy_cookie_domain ~\.(?P<sl_domain>[-0-9a-z]+\.[a-z]+)$ $sl_domain;
```

На одном уровне может быть указано несколько директив *proxy_cookie_domain*:

```
proxy_cookie_domain localhost example.org;
proxy_cookie_domain ~\.[a-z]+\.[a-z]+)$ $1;
```

Если к cookie могут быть применены несколько директив, будет выбрана первая из них.

Параметр `off` отменяет действие унаследованных с предыдущего уровня конфигурации директив *proxy_cookie_domain*.

proxy_cookie_flags

Синтаксис	<code>proxy_cookie_flags off cookie [флаг ...];</code>
По умолчанию	<code>proxy_cookie_flags off;</code>
Контекст	<code>http, server, location</code>

Задаёт один или несколько флагов для cookie. В качестве cookie можно использовать переменные. В качестве флага можно использовать переменные.

Параметры *secure*, *httponly*, *samesite=strict*, *samesite=lax*, *samesite=none* добавляют соответствующие флаги.

Параметры *nosecure*, *nohttponly*, *nosamesite* удаляют соответствующие флаги.

Cookie также можно задать при помощи регулярных выражений. При этом cookie должен начинаться с символа "~".

На одном уровне конфигурации может быть указано несколько директив *proxy_cookie_flags*:

```
proxy_cookie_flags one httponly;
proxy_cookie_flags ~ nsecure samesite=strict;
```

Если к cookie могут быть применены несколько директив, будет выбрана первая из них. В данном примере флаг *httponly* добавляется к cookie *one*, для остальных cookie добавляется флаг *samesite=strict* и удаляется флаг *secure*.

Параметр *off* отменяет действие всех директив *proxy_cookie_flags* на данном уровне.

proxy_cookie_path

Синтаксис	<code>proxy_cookie_path off;</code> <code>proxy_cookie_path <i>путь</i> <i>замена</i>;</code>
По умолчанию	<code>proxy_cookie_path off;</code>
Контекст	http, server, location

Задаёт текст, который нужно изменить в атрибуте *path* полей **Set-Cookie** заголовка ответа проксируемого сервера. Предположим, проксируемый сервер вернул поле заголовка **Set-Cookie** с атрибутом "path=/two/some/uri/". Директива

```
proxy_cookie_path /two/ /;
```

перепишет данный атрибут в виде "path=/some/uri/".

В строках *путь* и *замена* можно использовать переменные:

```
proxy_cookie_path $uri /some$uri;
```

Директиву также можно задать при помощи регулярных выражений. При этом *путь* должен начинаться либо с символа "~", если при сравнении следует учитывать регистр символов, либо с символов "~*", если регистр символов учитывать не нужно. Регулярное выражение может содержать именованные и позиционные группы захвата, а *замена* ссылаться на них:

```
proxy_cookie_path ~*/user/([~/]+) /u/$1;
```

На одном уровне может быть указано несколько директив *proxy_cookie_path*:

```
proxy_cookie_path /one/ /;
proxy_cookie_path / /two/;
```

Если к cookie могут быть применены несколько директив, будет выбрана первая из них.

Параметр *off* отменяет действие унаследованных с предыдущего уровня конфигурации директив *proxy_cookie_path*.

proxy_force_ranges

Синтаксис	proxy_force_ranges on off;
По умолчанию	proxy_force_ranges off;
Контекст	http, server, location

Включает поддержку диапазонов запрашиваемых байт (byte-range) для кэшированных и неэкэшированных ответов проксируемого сервера вне зависимости от наличия поля **Accept-Ranges** в заголовках этих ответов.

proxy_headers_hash_bucket_size

Синтаксис	proxy_headers_hash_bucket_size размер;
По умолчанию	proxy_headers_hash_bucket_size 64;
Контекст	http, server, location

Задаёт размер корзины для хэш-таблиц, используемых директивами *proxy_hide_header* и *proxy_set_header*. Подробнее настройка хэш-таблиц обсуждается отдельно.

proxy_headers_hash_max_size

Синтаксис	proxy_headers_hash_max_size размер;
По умолчанию	proxy_headers_hash_max_size 512;
Контекст	http, server, location

Задаёт максимальный размер хэш-таблиц, используемых директивами *proxy_hide_header* и *proxy_set_header*. Подробнее настройка хэш-таблиц обсуждается отдельно.

proxy_hide_header

Синтаксис	proxy_hide_header поле;
По умолчанию	—
Контекст	http, server, location

По умолчанию Angie не передает клиенту поля заголовка **Date**, **Server**, **X-Pad** и **X-Accel-...** из ответа проксированного сервера. Директива *proxy_hide_header* задает дополнительные поля, которые не будут передаваться. Если же передачу полей нужно разрешить, можно воспользоваться директивой *proxy_pass_header*.

proxy_http3_hq

Синтаксис	proxy_http3_hq on off;
По умолчанию	proxy_http3_hq off;
Контекст	http, server

Отключает или включает особый режим согласования **hq-interop**, используемый в функциональных тестах QUIC, на которые полагается Angie.

Предупреждение

Включайте этот режим только для запуска специализированных тестов, которым в явной форме необходим данный режим.

proxy_http3_max_concurrent_streams

Синтаксис	proxy_http3_max_concurrent_streams <i>число</i> ;
По умолчанию	proxy_http3_max_concurrent_streams 128;
Контекст	http, server

Инициализирует настройки HTTP/3 и QUIC, а также задает максимальное число параллельных HTTP/3-потоков в соединении. Требуется включения постоянных соединений.

proxy_http3_max_table_capacity

Синтаксис	proxy_http3_max_table_capacity <i>число</i> ;
Значение по умолчанию	proxy_http3_max_table_capacity 4096;
Контекст	http, server, location

Определяет емкость [динамической таблицы](#) для прокси-соединений.

Примечание

Похожая директива [http3_max_table_capacity](#) задает это значение для серверных соединений. Чтобы избежать ошибок, использование динамической таблицы отключается при включенном кэшировании в режиме проксирования.

proxy_http3_stream_buffer_size

Синтаксис	proxy_http3_stream_buffer_size размер;
По умолчанию	proxy_http3_stream_buffer_size 64k;
Контекст	http, server

Задает размер буфера, используемого для чтения и записи QUIC-потоков.

proxy_http_version

Синтаксис	proxy_http_version 1.0 1.1 2 3;
По умолчанию	proxy_http_version 1.1;
Контекст	http, server, location, if в location, limit_except

Задает версию протокола HTTP для проксирования. Для работы постоянных соединений рекомендуется версия 1.1 или выше.

proxy_ignore_client_abort

Синтаксис	<code>proxy_ignore_client_abort on off;</code>
По умолчанию	<code>proxy_ignore_client_abort off;</code>
Контекст	<code>http, server, location</code>

Определяет, закрывать ли соединение с проксируемым сервером в случае, если клиент закрыл соединение, не дождавшись ответа.

proxy_ignore_headers

Синтаксис	<code>proxy_ignore_headers поле ...;</code>
По умолчанию	—
Контекст	<code>http, server, location</code>

Запрещает обработку некоторых полей заголовка из ответа проксированного сервера. В директиве можно указать поля `X-Accel-Redirect`, `X-Accel-Expires`, `X-Accel-Limit-Rate`, `X-Accel-Buffering`, `X-Accel-Charset`, `Expires`, `Cache-Control`, `Set-Cookie` и `Vary`.

Если не запрещено, обработка этих полей заголовка заключается в следующем:

- `X-Accel-Expires`, `Expires`, `Cache-Control`, `Set-Cookie` и `Vary` задают параметры кэширования ответа;
- `X-Accel-Redirect` производит внутреннее перенаправление на указанный URI;
- `X-Accel-Limit-Rate` задает ограничение скорости передачи ответа клиенту;
- `X-Accel-Buffering` включает или выключает буферизацию ответа;
- `X-Accel-Charset` задает желаемую кодировку ответа.

proxy_intercept_errors

Синтаксис	<code>proxy_intercept_errors on off;</code>
По умолчанию	<code>proxy_intercept_errors off;</code>
Контекст	<code>http, server, location</code>

Определяет, передавать ли клиенту проксированные ответы с кодом больше либо равным 300, или же перехватывать их и перенаправлять на обработку Angie с помощью директивы `error_page`.

proxy_interface

Синтаксис	<code>proxy_interface и.м.я;</code>
По умолчанию	—
Контекст	<code>http, server, location</code>

Задаёт сетевой интерфейс, который будет использоваться в исходящих соединениях с проксируемым сервером; сокет привязывается к нему параметром сокета `SO_BINDTODEVICE`.

proxy_limit_rate

Синтаксис	proxy_limit_rate <i>скорость</i> ;
По умолчанию	proxy_limit_rate 0;
Контекст	http, server, location

Ограничивает скорость чтения ответа от проксируемого сервера. *Скорость* задается в байтах в секунду; можно использовать переменные.

0	отключает ограничение скорости
---	--------------------------------

Примечание

Ограничение устанавливается на запрос, поэтому, если Angie одновременно откроет два соединения к проксируемому серверу, суммарная скорость будет вдвое выше заданного ограничения. Ограничение работает только в случае, если включена буферизация ответов проксируемого сервера.

proxy_max_temp_file_size

Синтаксис	proxy_max_temp_file_size размер;
По умолчанию	proxy_max_temp_file_size 1024m;
Контекст	http, server, location

Если включена буферизация ответов проксируемого сервера, и ответ не помещается целиком в буферы, заданные директивами *proxy_buffer_size* и *proxy_buffers*, часть ответа может быть записана во временный файл. Эта директива задает максимальный размер временного файла. Размер данных, сбрасываемых во временный файл за один раз, задается директивой *proxy_temp_file_write_size*.

0	отключает возможность буферизации ответов во временные файлы
---	--

Примечание

Данное ограничение не распространяется на ответы, которые будут кэшированы или сохранены на диске.

proxy_method

Синтаксис	proxy_method <i>метод</i> ;
По умолчанию	—
Контекст	http, server, location

Задаёт HTTP-метод, который будет использоваться в передаваемых на проксируемый сервер запросах вместо метода из клиентского запроса. В значении параметра можно использовать переменные.

proxy_next_upstream

Синтаксис	proxy_next_upstream error timeout invalid_header http_500 http_502 http_503 http_504 http_403 http_404 http_429 non_idempotent off ...;
По умолчанию	proxy_next_upstream error timeout;
Контекст	http, server, location

Определяет, в каких случаях запрос будет передан следующему в группе upstream серверу:

error	произошла ошибка соединения с сервером, передачи ему запроса или чтения заголовка ответа сервера;
timeout	произошел таймаут во время соединения с сервером, передачи ему запроса или чтения заголовка ответа сервера;
invalid_header	сервер вернул пустой или неверный ответ;
http_500	сервер вернул ответ с кодом 500;
http_502	сервер вернул ответ с кодом 502;
http_503	сервер вернул ответ с кодом 503;
http_504	сервер вернул ответ с кодом 504;
http_403	сервер вернул ответ с кодом 403;
http_404	сервер вернул ответ с кодом 404;
http_429	сервер вернул ответ с кодом 429;
non_idempotent	обычно запросы с неидемпотентным методом (<i>POST</i> , <i>LOCK</i> , <i>PATCH</i>) не передаются на другой сервер, если запрос серверу группы уже был отправлен; включение параметра явно разрешает повторять подобные запросы;
off	запрещает передачу запроса следующему серверу.

Если все upstream-серверы недоступны или возвращают ответ со статусом, считающимся ошибкой для `proxy_next_upstream`, Angie возвращает собственную стандартную страницу ошибки вместо тела ответа от последнего upstream-сервера.

Примечание

Необходимо понимать, что передача запроса следующему серверу возможна только при условии, что клиенту еще ничего не передавалось. То есть, если ошибка или таймаут возникли в середине передачи ответа клиенту, то действие директивы на такой запрос не распространяется.

Директива также определяет, что считается неудачной попыткой работы с сервером.

error	всегда считаются неудачными попытками, даже если они не указаны в директиве
timeout	
invalid_header	
http_500	
http_502	считаются неудачными попытками, только если они указаны в директиве
http_503	
http_504	
http_429	
http_403	никогда не считаются неудачными попытками
http_404	

Передача запроса следующему серверу может быть ограничена по количеству попыток и по времени.

proxy_next_upstream_timeout

Синтаксис	proxy_next_upstream_timeout время;
По умолчанию	proxy_next_upstream_timeout 0;
Контекст	http, server, location

Ограничивает время, в течение которого возможна передача запроса следующему серверу.

0	отключает это ограничение
---	---------------------------

proxy_next_upstream_tries

Синтаксис	proxy_next_upstream_tries число;
По умолчанию	proxy_next_upstream_tries 0;
Контекст	http, server, location

Ограничивает число допустимых попыток для передачи запроса следующему серверу.

0	отключает это ограничение
---	---------------------------

proxy_no_cache

Синтаксис	proxy_no_cache строка ...;
По умолчанию	—
Контекст	http, server, location

Задаёт условия, при которых ответ не будет сохраняться в кэш. Если значение хотя бы одного из строковых параметров непустое и не равно "0", то ответ не будет сохранен:

```
proxy_no_cache $cookie_nocache $arg_nocache$arg_comment;
proxy_no_cache $http_pragma $http_authorization;
```

В значении параметров можно использовать переменные.

Можно использовать совместно с директивой *proxy_cache_bypass*.

proxy_pass

Синтаксис	proxy_pass uri;
По умолчанию	—
Контекст	location, if в location, limit_except

Задаёт протокол и адрес проксируемого сервера, а также необязательный URI, на который должен отображаться location. В качестве протокола можно указать **http** или **https**. Адрес может быть указан в виде доменного имени или IP-адреса, и необязательного порта:

```
proxy_pass http://localhost:8000/uri/;
```

или в виде пути UNIX-сокета, который указывается после слова `unix` и заключается в двоеточия:

```
proxy_pass http://unix:/tmp/backend.socket:/uri/;
```

Если доменному имени соответствует несколько адресов, то все они будут использоваться по очереди (round-robin). Кроме того, в качестве адреса можно указать группу серверов.

В значении параметра можно использовать переменные. В этом случае, если адрес указан в виде доменного имени, имя ищется среди описанных групп серверов и если не найдено, то определяется с помощью `resolver`'а.

Примечание

Доменное имя в `proxy_pass`, которое не содержит переменных и не совпадает с именем описанной группы серверов, неявно создает группу серверов из одного сервера, эквивалентную:

```
upstream example.com {
    server example.com;
}
```

Имена таких групп серверов являются общими в пределах всей конфигурации `http`, независимо от блока `server`, в котором используется `proxy_pass`. Поскольку у неявно созданного `server` нет параметра `resolve`, имя разрешается один раз, на этапе загрузки конфигурации, и полученные адреса используются вплоть до следующей перезагрузки, даже если задан `resolver`.

В результате `proxy_pass`, переменная которого принимает то же самое имя, повторно использует эту существующую группу серверов с адресами, полученными при загрузке, вместо того чтобы разрешать имя при каждом запросе через `resolver`. Чтобы повторно разрешать имя во время работы и отслеживать изменения DNS, опишите группу серверов явно и добавьте параметр `resolve` к ее `server`.

URI запроса передается на сервер так:

- Если директива `proxy_pass` указана с **URI**, то при передаче запроса серверу часть нормализованного URI запроса, соответствующая `location`, заменяется на URI, указанный в директиве:

```
location /name/ {
    proxy_pass http://127.0.0.1/remote/;
}
```

- Если директива `proxy_pass` указана **без URI**, то при обработке первоначального запроса на сервер передается URI запроса в том же виде, в каком его прислал клиент, а при обработке измененного URI - нормализованный URI запроса целиком:

```
location /some/path/ {
    proxy_pass http://127.0.0.1;
}
```

В ряде случаев часть URI запроса, подлежащую замене, выделить невозможно:

- Если `location` задан регулярным выражением, а также в именованных `location`.

В этих случаях `proxy_pass` следует указывать без URI.

- Если внутри проксируемого `location` с помощью директивы `rewrite` изменяется URI, и именно с этой конфигурацией будет обрабатываться запрос (`break`):

```
location /name/ {
    rewrite /name/([^/]+) /users?name=$1 break;
```

```
proxy_pass http://127.0.0.1;
}
```

В этом случае URI, указанный в директиве, игнорируется, и на сервер передается измененный URI запроса целиком.

- При использовании переменных в *proxy_pass*:

```
location /name/ {
    proxy_pass http://127.0.0.1$request_uri;
}
```

В этом случае если в директиве указан URI, он передается на сервер как есть, заменяя URI первоначального запроса.

Проксирование WebSocket требует особой настройки.

Примечание

Если *proxy_pass* стоит в *location* с косой чертой в конце префикса (например, *location /name/*), и при этом в директиве *auto_redirect* указано *default*, запросы без косой черты в конце будут перенаправляться (*/name -> /name/*).

proxy_pass_header

Синтаксис	<i>proxy_pass_header поле ...;</i>
По умолчанию	—
Контекст	http, server, location

Разрешает передавать от проксируемого сервера клиенту запрещенные для передачи поля заголовка.

proxy_pass_request_body

Синтаксис	<i>proxy_pass_request_body on off;</i>
По умолчанию	<i>proxy_pass_request_body on;</i>
Контекст	http, server, location

Позволяет запретить передачу исходного тела запроса на проксируемый сервер.

```
location /x-accel-redirect-here/ {
    proxy_method GET;
    proxy_pass_request_body off;
    proxy_set_header Content-Length "";

    proxy_pass ...;
}
```

См. также директивы *proxy_set_header* и *proxy_pass_request_headers*.

proxy_pass_request_headers

Синтаксис	<code>proxy_pass_request_headers on off;</code>
По умолчанию	<code>proxy_pass_request_headers on;</code>
Контекст	<code>http, server, location</code>

Позволяет запретить передачу полей заголовка исходного запроса на проксируемый сервер.

```
location /x-accel-redirect-here/ {
    proxy_method GET;
    proxy_pass_request_headers off;
    proxy_pass_request_body off;

    proxy_pass ...;
}
```

См. также директивы `proxy_set_header` и `proxy_pass_request_body`.

proxy_pass_trailers

Синтаксис	<code>proxy_pass_trailers on off;</code>
По умолчанию	<code>proxy_pass_trailers off;</code>
Контекст	<code>http, server, location</code>

Разрешает передачу полей трейлеров от проксируемого сервера клиенту.

Секция трейлеров в HTTP/1.1 **включается явно**.

```
location / {
    proxy_http_version 1.1;
    proxy_set_header Connection "te";
    proxy_set_header TE "trailers";
    proxy_pass_trailers on;

    proxy_pass ...;
}
```

proxy_quic_active_connection_id_limit

Синтаксис	<code>proxy_quic_active_connection_id_limit <i>число</i>;</code>
По умолчанию	<code>proxy_quic_active_connection_id_limit 2;</code>
Контекст	<code>http, server</code>

Задаёт значение транспортного параметра QUIC `active_connection_id_limit`. Это максимальное число активных идентификаторов соединений, поддерживаемых для одного сервера.

proxy_quic_gso

Синтаксис	proxy_quic_gso on off;
По умолчанию	proxy_quic_gso off;
Контекст	http, server

Отключает или включает отправку данных в оптимизированном пакетном режиме QUIC, использующем программное снижение нагрузки путем сегментации ([generic segmentation offload](#)).

proxy_quic_host_key

Синтаксис	proxy_quic_host_key <i>файл</i> ;
По умолчанию	—
Контекст	http, server

Задаёт *файл* с секретным ключом, применяемым в QUIC при шифровании токенов [Stateless Reset](#) и [Address Validation](#). По умолчанию случайный ключ создается при каждом перезапуске. Токены, созданные при помощи старых ключей, не принимаются.

proxy_read_timeout

Синтаксис	proxy_read_timeout время;
По умолчанию	proxy_read_timeout 60s;
Контекст	http, server, location

Задаёт таймаут при чтении ответа проксируемого сервера. Таймаут устанавливается не на всю передачу ответа, а только между двумя операциями чтения. Если по истечении этого времени проксируемый сервер ничего не передаст, соединение закрывается.

proxy_redirect

Синтаксис	proxy_redirect default; proxy_redirect off; proxy_redirect <i>перенаправление замена</i> ;
По умолчанию	proxy_redirect default;
Контекст	http, server, location

Задаёт текст, который нужно изменить в полях заголовка "Location" и "Refresh" в ответе проксируемого сервера.

Предположим, проксируемый сервер вернул поле заголовка:

```
Location: http://localhost:8000/two/some/uri/
```

Директива

```
proxy_redirect http://localhost:8000/two/ http://frontend/one/;
```

перепишет эту строку в виде:

```
Location: http://frontend/one/some/uri/
```

В заменяемой строке можно не указывать имя сервера:

```
proxy_redirect http://localhost:8000/two/ /;
```

тогда будут подставлены основное имя сервера и порт, если он отличен от 80.

Стандартная замена, задаваемая параметром `default`, использует параметры директив `location` и `proxy_pass`. Поэтому две нижеприведенные конфигурации одинаковы:

```
location /one/ {
    proxy_pass      http://upstream:port/two/;
    proxy_redirect default;
```

```
location /one/ {
    proxy_pass      http://upstream:port/two/;
    proxy_redirect http://upstream:port/two/ /one/;
```

Предупреждение

Параметр `default` недопустим, если в `proxy_pass` используются переменные.

В строке *замена* можно использовать переменные:

```
proxy_redirect http://localhost:8000/ http://$host:$server_port/;
```

В строке *перенаправление* тоже можно использовать переменные:

```
proxy_redirect http://$proxy_host:8000/ /;
```

Директиву также можно задать при помощи регулярных выражений. При этом *перенаправление* должно начинаться либо с символа `"~"`, если при сравнении следует учитывать регистр символов, либо с символов `"~*"`, если регистр символов учитывать не нужно. Регулярное выражение может содержать именованные и позиционные группы захвата, а *замена* ссылаться на них:

```
proxy_redirect ~^(http://[~:~*]+):~*\d+(/~*~*)$ $1$2;
proxy_redirect ~*/user/([~*~*]+)/([~*~*])$ http://$1.example.com/$2;
```

На одном уровне может быть указано несколько директив `proxy_redirect`:

```
proxy_redirect default;
proxy_redirect http://localhost:8000/ /;
proxy_redirect http://www.example.com/ /;
```

Если к полям заголовка в ответе проксируемого сервера могут быть применены несколько директив, будет выбрана первая из них.

Параметр `off` отменяет действие унаследованных с предыдущего уровня конфигурации директив `proxy_redirect`.

С помощью этой директивы можно также добавлять имя хоста к относительным перенаправлениям, выдаваемым проксируемым сервером:

```
proxy_redirect / /;
```


proxy_request_buffering

Синтаксис	<code>proxy_request_buffering on off;</code>
По умолчанию	<code>proxy_request_buffering on;</code>
Контекст	<code>http, server, location</code>

Разрешает или запрещает использовать буферизацию тела запроса клиента.

on	тело запроса полностью читается от клиента перед отправкой запроса на проксируемый сервер.
off	тело запроса отправляется на проксируемый сервер сразу же по мере его поступления. В этом случае запрос не может быть передан следующему серверу, если Angie уже начал отправку тела запроса.

Если для отправки тела исходного запроса используется HTTP/1.1 и передача данных частями (chunked transfer encoding), то тело запроса буферизуется независимо от значения директивы, если для проксирования не включен ни один из протоколов HTTP/1.1, HTTP/2 или HTTP/3.

proxy_send_lowat

Синтаксис	<code>proxy_send_lowat размер;</code>
По умолчанию	<code>proxy_send_lowat 0;</code>
Контекст	<code>http, server, location</code>

При установке директивы в ненулевое значение Angie будет пытаться минимизировать число операций отправки на исходящих соединениях с проксируемым сервером либо при помощи флага `NOTE_LOWAT` метода `kqueue`, либо при помощи параметра сокета `SO_SNDLOWAT`, с указанным размером.

Примечание

Эта директива игнорируется на Linux, Solaris и Windows.

proxy_send_timeout

Синтаксис	<code>proxy_send_timeout время;</code>
По умолчанию	<code>proxy_send_timeout 60s;</code>
Контекст	<code>http, server, location</code>

Задаёт таймаут при передаче запроса проксируемому серверу. Таймаут устанавливается не на всю передачу запроса, а только между двумя операциями записи. Если по истечении этого времени проксируемый сервер не примет новых данных, соединение закрывается.

proxy_set_body

Синтаксис	<code>proxy_set_body значение;</code>
По умолчанию	—
Контекст	http, server, location

Позволяет переопределить тело запроса, передаваемое на проксируемый сервер. В качестве значения можно использовать переменные.

proxy_set_header

Синтаксис	<code>proxy_set_header поле значение;</code>
По умолчанию	<code>proxy_set_header Host \$proxy_host;</code>
Контекст	http, server, location

Позволяет переопределять или добавлять поля заголовка запроса, передаваемые проксируемому серверу. В качестве значения можно использовать переменные. Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы `proxy_set_header`. По умолчанию переопределяются только два поля:

```
proxy_set_header Host $proxy_host;
proxy_set_header Connection close;
```

Если включено кэширование, поля заголовка `If-Modified-Since`, `If-Unmodified-Since`, `If-None-Match`, `If-Match`, `Range` и `If-Range` исходного запроса не передаются на проксируемый сервер.

Неизмененное поле заголовка запроса "Host" можно передать так:

```
proxy_set_header Host $http_host;
```

Однако, если это поле отсутствует в заголовке запроса клиента, то ничего передаваться не будет. В этом случае лучше воспользоваться переменной `$host` - ее значение равно имени сервера в поле "Host" заголовка запроса, или же основному имени сервера, если поля нет:

```
proxy_set_header Host $host;
```

Кроме того, можно передать имя сервера вместе с портом проксируемого сервера:

```
proxy_set_header Host $host:$proxy_port;
```

Если значение поля заголовка — пустая строка, то поле вообще не будет передаваться проксируемому серверу:

```
proxy_set_header Accept-Encoding "";
```

proxy_socket_keepalive

Синтаксис	<code>proxy_socket_keepalive on off;</code>
По умолчанию	<code>proxy_socket_keepalive off;</code>
Контекст	http, server, location

Конфигурирует поведение "TCP keepalive" для исходящих соединений к проксируемому серверу.

""	По умолчанию для сокета действуют настройки операционной системы.
on	для сокета включается параметр <i>SO_KEEPALIVE</i>

proxy_ssl_certificate

Синтаксис	<code>proxy_ssl_certificate <i>файл</i> [<i>файл</i>];</code>
По умолчанию	—
Контекст	http, server, location

Задаёт файл с сертификатом в формате PEM для аутентификации на проксируемом HTTPS-сервере. В имени файла можно использовать переменные.

При включенном *proxy_ssl_ntls* директива принимает два аргумента вместо одного:

```
location /proxy {
    proxy_ssl_ntls on;

    proxy_ssl_certificate      sign.crt enc.crt;
    proxy_ssl_certificate_key  sign.key enc.key;

    proxy_ssl_ciphers "ECC-SM2-WITH-SM4-SM3:ECDSA-SM2-WITH-SM4-SM3:RSA";

    proxy_pass https://backend:443;
}
```

proxy_ssl_certificate_cache

Синтаксис	<code>proxy_ssl_certificate_cache off;</code> <code>proxy_ssl_certificate_cache max=<i>N</i> [<i>inactive=time</i>] [<i>valid=time</i>];</code>
Значение по умолчанию	<code>proxy_ssl_certificate_cache off;</code>
Контекст	http, server, location

Определяет кэш для хранения SSL-сертификатов и секретных ключей, заданных через переменные.

Директива поддерживает следующие параметры:

- **max** — устанавливает максимальное количество элементов в кэше. При переполнении кэша удаляются наименее недавно использованные (LRU) элементы.
- **inactive** — определяет время, после которого элемент будет удален, если к нему не было обращений в течение этого времени, даже если он еще находится в пределах своего периода **valid**; каждое обращение сбрасывает этот таймер. Значение по умолчанию — 10 секунд.
- **valid** — определяет время, в течение которого элемент кэша считается действительным и может использоваться повторно. Значение по умолчанию — 60 секунд. По истечении этого времени сертификаты перезагружаются или проходят повторную проверку.
- **off** — отключает кэш.

Пример:

```
proxy_ssl_certificate      $proxy_ssl_server_name.crt;
proxy_ssl_certificate_key  $proxy_ssl_server_name.key;
proxy_ssl_certificate_cache max=1000 inactive=20s valid=1m;
```

proxy_ssl_certificate_key

Синтаксис	proxy_ssl_certificate_key <i>файл</i> [<i>файл</i>];
По умолчанию	—
Контекст	http, server, location

Задаёт файл с секретным ключом в формате PEM для аутентификации на проксируемом HTTPS-сервере.

Вместо файла можно указать значение `engine:`name`:id`, которое загружает ключ с указанным `id` из OpenSSL engine с заданным именем.

Вместо файла можно указать значение `store:scheme:id`, которое используется для загрузки ключа с указанным `id` и URI-схемой `scheme`, зарегистрированной в OpenSSL provider, например `pkcs11`.

В имени файла можно использовать переменные.

При включенном `proxy_ssl_nTLS` директива принимает два аргумента вместо одного:

```
location /proxy {
    proxy_ssl_nTLS on;

    proxy_ssl_certificate      sign.crt enc.crt;
    proxy_ssl_certificate_key  sign.key enc.key;

    proxy_ssl_ciphers "ECC-SM2-WITH-SM4-SM3:ECDHE-SM2-WITH-SM4-SM3:RSA";

    proxy_pass https://backend:443;
}
```

proxy_ssl_ciphers

Синтаксис	proxy_ssl_ciphers <i>шифры</i> ;
По умолчанию	proxy_ssl_ciphers DEFAULT;
Контекст	http, server, location

Описывает разрешенные шифры для запросов к проксируемому HTTPS-серверу. Шифры задаются в формате, поддерживаемом библиотекой OpenSSL.

Список шифров зависит от установленной версии OpenSSL. Полный список можно посмотреть с помощью команды `openssl ciphers`.

Предупреждение

Директива `proxy_ssl_ciphers` не настраивает шифры для TLS 1.3 при использовании OpenSSL. Для настройки шифров TLS 1.3 в OpenSSL используйте директиву `proxy_ssl_conf_command`, добавленную для расширенной конфигурации SSL.

- В LibreSSL шифры TLS 1.3 можно настраивать с помощью `proxy_ssl_ciphers`.

- В BoringSSL шифры TLS 1.3 настроить невозможно.

proxy_ssl_conf_command

Синтаксис	<code>proxy_ssl_conf_command имя значение;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт произвольные конфигурационные команды OpenSSL при установлении соединения с проксируемым HTTPS-сервером.

Примечание

Директива поддерживается при использовании OpenSSL 1.0.2 и выше. Чтобы настроить шифры TLS 1.3 в OpenSSL, используйте команду `ciphersuites`.

На одном уровне может быть указано несколько директив `proxy_ssl_conf_command`. Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы `proxy_ssl_conf_command`.

Предупреждение

Следует учитывать, что изменение настроек OpenSSL напрямую может привести к неожиданному поведению.

proxy_ssl_crl

Синтаксис	<code>proxy_ssl_crl файл;</code>
По умолчанию	—
Контекст	http, server, location

Указывает файл с отозванными сертификатами (CRL) в формате PEM, используемыми при проверке сертификата проксируемого HTTPS-сервера.

proxy_ssl_keylog_file

Синтаксис	<code>proxy_ssl_keylog_file файл;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт файл для журналирования SSL/TLS-ключей соединений с проксируемыми HTTPS-серверами в контексте, использующий формат `SSLKEYLOGFILE`, который применяется, например, в Wireshark для расшифровки перехваченного TLS-трафика.

Директиву не следует использовать с LibreSSL.

⚠ Предупреждение

Файл содержит конфиденциальный ключевой материал TLS, с помощью которого можно расшифровать перехваченный трафик соответствующих соединений. Используйте директиву только для отладки или анализа протокола, ограничьте доступ к файлу и удалите его, когда он больше не нужен.

proxy_ssl_name

Синтаксис	<code>proxy_ssl_name имя;</code>
По умолчанию	<code>proxy_ssl_name \$proxy_host;</code>
Контекст	http, server, location

Позволяет переопределить имя сервера, используемое при проверке сертификата проксируемого HTTPS-сервера, а также для передачи его через SNI при установлении соединения с проксируемым HTTPS-сервером.

Имя сервера также можно задать с помощью переменных.

По умолчанию используется имя хоста из URL'a, заданного директивой `proxy_pass`.

proxy_ssl_ntls

Синтаксис	<code>proxy_ssl_ntls on off;</code>
По умолчанию	<code>proxy_ssl_ntls off;</code>
Контекст	http, server

Включает клиентскую поддержку NTLS при использовании TLS библиотеки `Tongtsuo`.

```
location /proxy {
    proxy_ssl_ntls on;

    proxy_ssl_certificate      sign.crt enc.crt;
    proxy_ssl_certificate_key  sign.key enc.key;

    proxy_ssl_ciphers "ECC-SM2-WITH-SM4-SM3:ECDHE-SM2-WITH-SM4-SM3:RSA";

    proxy_pass https://backend:443;
}
```

proxy_ssl_password_file

Синтаксис	<code>proxy_ssl_password_file файл;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт файл с паролями от секретных ключей, где каждый пароль указан на отдельной строке. Пароли применяются по очереди в момент загрузки ключа.

proxy_ssl_protocols

Синтаксис	<code>proxy_ssl_protocols [SSLv2] [SSLv3] [TLSv1] [TLSv1.1] [TLSv1.2] [TLSv1.3];</code>
По умолчанию	<code>proxy_ssl_protocols TLSv1.2 TLSv1.3;</code>
Контекст	<code>http, server, location</code>

Разрешает указанные протоколы для запросов к проксируемому HTTPS-серверу.

proxy_ssl_server_name

Синтаксис	<code>proxy_ssl_server_name on off;</code>
По умолчанию	<code>proxy_ssl_server_name off;</code>
Контекст	<code>http, server, location</code>

Разрешает или запрещает передачу имени сервера, заданного директивой `proxy_ssl_name`, через расширение [Server Name Indication](#) протокола TLS (SNI, RFC 6066) при установлении соединения с проксируемым HTTPS-сервером.

proxy_ssl_session_reuse

Синтаксис	<code>proxy_ssl_session_reuse on off;</code>
По умолчанию	<code>proxy_ssl_session_reuse on;</code>
Контекст	<code>http, server, location</code>

Определяет, использовать ли повторно SSL-сессии при работе с проксируемым сервером. Если в логах появляются ошибки "`SSL3_GET_FINISHED:digest check failed`", то можно попробовать включить повторное использование сессий.

proxy_ssl_trusted_certificate

Синтаксис	<code>proxy_ssl_trusted_certificate <i>файл</i>;</code>
По умолчанию	—
Контекст	<code>http, server, location</code>

Задаёт файл с доверенными сертификатами СА в формате PEM, используемыми при проверке сертификата проксируемого HTTPS-сервера.

proxy_ssl_verify

Синтаксис	<code>proxy_ssl_verify on off;</code>
По умолчанию	<code>proxy_ssl_verify off;</code>
Контекст	<code>http, server, location</code>

Разрешает или запрещает проверку сертификата проксируемого HTTPS-сервера.

proxy_ssl_verify_depth

Синтаксис	<code>proxy_ssl_verify_depth <i>число</i>;</code>
По умолчанию	<code>proxy_ssl_verify_depth 1;</code>
Контекст	<code>http, server, location</code>

Устанавливает глубину проверки в цепочке сертификатов проксируемого HTTPS-сервера.

proxy_store

Синтаксис	<code>proxy_store on off <i>строка</i>;</code>
По умолчанию	<code>proxy_store off;</code>
Контекст	<code>http, server, location</code>

Разрешает сохранение на диск файлов.

<code>on</code>	сохраняет файлы в соответствии с путями, указанными в директивах <i>alias</i> или <i>root</i>
<code>off</code>	запрещает сохранение файлов

Имя файла можно задать явно с помощью строки с переменными:

```
proxy_store /data/www$original_uri;
```

Время изменения файлов выставляется согласно полученному полю **Last-Modified** в заголовке ответа. Ответ сначала записывается во временный файл, а потом этот файл переименовывается. Временный файл и постоянное место хранения ответа могут располагаться на разных файловых системах. Однако нужно учитывать, что в этом случае вместо дешевой операции переименовывания в пределах одной файловой системы файл копируется с одной файловой системы на другую. Поэтому лучше, если сохраняемые файлы будут находиться на той же файловой системе, что и каталог с временными файлами, задаваемый директивой *proxy_temp_path* для данного *location*.

Директиву можно использовать для создания локальных копий статических неизменяемых файлов, например, так:

```
location /images/ {
    root          /data/www;
    error_page    404 = /fetch$uri;
}

location /fetch/ {
    internal;

    proxy_pass    http://backend/;
    proxy_store    on;
    proxy_store_access user:rw group:rw all:r;
    proxy_temp_path /data/temp;

    alias          /data/www/;
}
```

или так:


```
location /images/ {
    root                /data/www;
    error_page          404 = @fetch;
}

location @fetch {
    internal;

    proxy_pass          http://backend;
    proxy_store          on;
    proxy_store_access  user:rw group:rw all:r;
    proxy_temp_path     /data/temp;

    root                /data/www;
}
```

proxy_store_access

Синтаксис	<code>proxy_store_access пользователи:права ...;</code>
По умолчанию	<code>proxy_store_access user:rw;</code>
Контекст	http, server, location

Задаёт права доступа для создаваемых файлов и каталогов, например,

```
proxy_store_access user:rw group:rw all:r;
```

Если заданы какие-либо права для *group* или *all*, то права для *user* указывать необязательно:

```
proxy_store_access group:rw all:r;
```

proxy_tcp_congestion

Синтаксис	<code>proxy_tcp_congestion алгоритм;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт алгоритм управления перегрузкой TCP (параметр сокета TCP_CONGESTION) для исходящих соединений с проксируемым сервером; к соединениям через UNIX-сокеты параметр не применяется.

Имя алгоритма не проверяется на этапе конфигурации: если алгоритм неизвестен или недоступен, при установке исходящего соединения в лог записывается сообщение уровня **alert**, а для соединения действует системный алгоритм по умолчанию.

proxy_temp_file_write_size

Синтаксис	<code>proxy_temp_file_write_size размер;</code>
По умолчанию	<code>proxy_temp_file_write_size 8k 16k;</code>
Контекст	http, server, location

Ограничивает размер данных, сбрасываемых во временный файл за один раз, при включенной буферизации ответов проксируемого сервера во временные файлы. По умолчанию размер ограни-

чен двумя буферами, заданными директивами *proxy_buffer_size* и *proxy_buffers*. Максимальный размер временного файла задается директивой *proxy_max_temp_file_size*.

proxy_temp_path

Синтаксис	<code>proxy_temp_path путь [уровень1 [уровень2 [уровень3]]]`;</code>
По умолчанию	<code>proxy_temp_path proxy_temp;</code> (путь зависит от параметра сборки <code>--http-proxy-temp-path</code>)
Контекст	http, server, location

Задаёт имя каталога для хранения временных файлов с данными, полученными от проксируемых серверов. В каталоге может использоваться иерархия подкаталогов до трех уровней. Например, при такой конфигурации

```
proxy_temp_path /spool/angie/proxy_temp 1 2;
```

временный файл будет следующего вида:

```
/spool/angie/proxy_temp/7/45/00000123457
```

См. также параметр *use_temp_path* директивы *proxy_cache_path*.

Встроенные переменные

В модуле *http_proxy* есть встроенные переменные, которые можно использовать для формирования заголовков с помощью директивы *proxy_set_header*:

`$proxy_add_x_forwarded_for`

поле заголовка запроса клиента **X-Forwarded-For** и добавленная к нему через запятую переменная *\$remote_addr*. Если же поля **X-Forwarded-For** в заголовке запроса клиента нет, то переменная *\$proxy_add_x_forwarded_for* равна переменной *\$remote_addr*.

`$proxy_host`

имя и порт проксируемого сервера, как указано в директиве *proxy_pass*;

`$proxy_port`

порт проксируемого сервера, как указано в директиве *proxy_pass*, или стандартный порт протокола;

Random Index

Обслуживает запросы, оканчивающиеся косой чертой (/), и выдает случайный файл в качестве индексного файла каталога. Модуль выполняется до модуля *http_index*.

Пример конфигурации

```
location / {
    random_index on;
}
```

Директивы

random_index

Синтаксис	<code>random_index on off;</code>
По умолчанию	<code>random_index off;</code>
Контекст	<code>location</code>

Разрешает или запрещает в содержащем эту директиву `location` обработку этим модулем.

RealIP

Позволяет менять адрес и необязательный порт клиента на переданные в указанном поле заголовка.

Пример конфигурации

```
set_real_ip_from 192.168.1.0/24;
set_real_ip_from 192.168.2.1;
set_real_ip_from 2001:0db8::/32;
real_ip_header X-Forwarded-For;
real_ip_recursive on;
```

Директивы

real_ip_header

Синтаксис	<code>real_ip_header none X-Real-IP X-Forwarded-For proxy_protocol;</code>
По умолчанию	<code>real_ip_header X-Real-IP;</code>
Контекст	<code>http, server, location</code>

Задаёт поле заголовка запроса, значение которого будет использоваться для замены адреса клиента.

Значение поля заголовка запроса, содержащее необязательный порт, также используется для замены порта клиента. Адрес и порт должны быть указаны согласно [RFC 3986](#).

Параметр `proxy_protocol` меняет адрес клиента на указанный в заголовке PROXY-протокола. Протокол PROXY должен быть предварительно включен при помощи установки параметра `proxy_protocol` в директиве `listen`.

real_ip_recursive

Синтаксис	<code>real_ip_recursive on off;</code>
По умолчанию	<code>real_ip_recursive off;</code>
Контекст	<code>http, server, location</code>

При выключенном рекурсивном поиске исходный адрес клиента, совпадающий с одним из доверенных адресов, заменяется на последний адрес, переданный в поле заголовка запроса, заданного директивой `real_ip_header`. При включенном рекурсивном поиске исходный адрес клиента, совпадающий с одним из доверенных адресов, заменяется на последний не доверенный адрес, переданный в заданном поле заголовка запроса.

set_real_ip_from

Синтаксис	<code>set_real_ip_from <i>адрес</i> <i>CIDR</i> unix::;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт доверенные адреса, которые передают верный адрес для замены. Если указано специальное значение `unix::`, доверенными будут считаться все UNIX-сокеты. Доверенные адреса могут быть также заданы при помощи имени хоста.

Встроенные переменные

`$realip_remote_addr`

хранит исходный адрес клиента

`$realip_remote_port`

хранит исходный порт клиента

Referer

Позволяет блокировать доступ к сайту для запросов с неверными значениями поля **Referer** в заголовке. Следует иметь в виду, что подделать запрос с нужным значением поля **Referer** не составляет большого труда, поэтому цель использования данного модуля заключается не в стопроцентном блокировании подобных запросов, а в блокировании массового потока запросов, сделанных обычными браузерами. Нужно также учитывать, что обычные браузеры могут не передавать поле **Referer** даже для верных запросов.

Пример конфигурации

```
valid_referers none blocked server_names
               *.example.com example.* www.example.org/galleries/
               ~\.google\.;

if ($invalid_referer) {
    return 403;
}
```

Директивы

referer_hash_bucket_size

Синтаксис	<code>referer_hash_bucket_size размер;</code>
По умолчанию	<code>referer_hash_bucket_size 64;</code>
Контекст	server, location

Задаёт размер корзины хэш-таблиц со значениями **Referer**. Подробнее настройка хэш-таблиц описывается отдельно.

referer_hash_max_size

Синтаксис	<code>referer_hash_max_size</code> размер;
По умолчанию	<code>referer_hash_max_size 2048;</code>
Контекст	<code>server, location</code>

Задаёт максимальный размер хэш-таблиц со значениями **Referer**. Подробнее настройка хэш-таблиц обсуждается отдельно.

valid_referers

Синтаксис	<code>valid_referers none blocked server_names строка ...;</code>
По умолчанию	—
Контекст	<code>server, location</code>

Задаёт значения поля **Referer** заголовка запроса, при которых встроенная переменная `$invalid_referer` будет иметь пустую строку в качестве значения. В противном случае значение переменной равно "1". Поиск совпадения производится без учёта регистра символов.

Параметры могут быть следующие:

none	поле Referer в заголовке запроса отсутствует;
blocked	поле Referer в заголовке запроса присутствует, но его значение удалено межсетевым экраном (firewall) или прокси-сервером; к таким значениям относятся строки, не начинающиеся на "http://" или "https://";
server_names	в поле Referer заголовка запроса указано одно из имен сервера;
произвольная строка	задаёт имя сервера и необязательное начало URI. В начале или конце имени сервера может быть "*". При проверке порт сервера в поле Referer игнорируется;
регулярное выражение	в начале должен быть символ "~". Необходимо учитывать, что на совпадение с выражением будет проверяться текст, начинающийся после "http://" или "https://".

Пример:

```
valid_referers none blocked server_names
               *.example.com example.* www.example.org/galleries/
               ~\.google\.;
```

Встроенные переменные

`$invalid_referer`

Пустая строка, если значение поля **Referer** заголовка запроса считается правильным, иначе "1".

Rewrite

Позволяет изменять URI запроса с помощью регулярных выражений PCRE, делать перенаправления и выбирать конфигурацию по условию.

Директивы `break`, `goto`, `if`, `return`, `rewrite` и `set` обрабатываются в следующем порядке:

- последовательно выполняются директивы этого модуля, описанные на уровне `server`;

- в цикле:
 - ищется `location` по URI запроса;
 - последовательно выполняются директивы этого модуля, описанные в найденном `location`;
 - цикл повторяется, если URI запроса изменялся, но не более 10 раз.

Директивы

`break`

Синтаксис	<code>break;</code>
По умолчанию	—
Контекст	<code>server</code> , <code>location</code> , <code>if</code>

Завершает обработку текущего набора директив модуля `http_rewrite`.

Если директива указана внутри `location`, дальнейшая обработка запроса продолжается в этом `location`.

Пример:

```
if ($slow) {
    limit_rate 10k;
    break;
}
```

`goto`

Синтаксис	<code>goto @имя;</code>
По умолчанию	—
Контекст	<code>server</code> , <code>location</code> , <code>if</code>

Завершает обработку текущего набора директив модуля `http_rewrite` и выполняет безусловное внутреннее перенаправление в именованный `location`, заданный параметром `имя`. В отличие от `rewrite`, не изменяет `$uri`. Параметром должно быть имя с префиксом `@`; в противном случае конфигурация отклоняется.

Перенаправление в именованный `location` учитывается в том же лимите из 10 изменений URI, что и при изменении URI запроса.

Например, запросы без cookie `session` можно направить в отдельный бэкенд аутентификации:

```
if ($cookie_session = "") {
    goto @auth_backend;
}

location / {
    proxy_pass http://backend;
}

location @auth_backend {
    proxy_pass http://127.0.0.1:8000;
    proxy_set_header Host $host;
}
```

if

Синтаксис	<code>if (условие) { ... }</code>
По умолчанию	—
Контекст	server, location

Проверяется указанное условие. Если оно истинно, то выполняются указанные в фигурных скобках директивы этого модуля и запросу назначается конфигурация, указанная внутри директивы `if`. Конфигурации внутри директив `if` наследуются с предыдущего уровня конфигурации.

Предупреждение

Хотя внутри блока `if` можно применять директивы других модулей, делать это не рекомендуется, так как это может привести к непредвиденному поведению.

В качестве условия могут быть заданы:

- имя переменной; ложными значениями переменной являются пустая строка или `"0"`;
- сравнение переменной со строкой с помощью операторов `"="` и `"!="`;
- соответствие переменной регулярному выражению с учетом регистра символов — `"~"` и без него — `"~*"`. В регулярных выражениях можно использовать группы захвата, которые затем доступны в виде переменных `$1..$9`. Также можно использовать отрицательные операторы `"!~"` и `"!~*"`. Если в регулярном выражении встречаются символы `"}"` или `";"`, то все выражение следует заключить в одинарные или двойные кавычки.
- проверка существования файла с помощью операторов `"-f"` и `"!-f"`;
- проверка существования каталога с помощью операторов `"-d"` и `"!-d"`;
- проверка существования файла, каталога или символической ссылки с помощью операторов `"-e"` и `"!-e"`;
- проверка исполняемости файла с помощью операторов `"-x"` и `"!-x"`.

Примеры:

```
if ($http_user_agent ~ MSIE) {
    rewrite ^(.*)$ /msie/$1 break;
}

if ($http_cookie ~* "id=(\[~;]+)(?:;|$)" ) {
    set $id $1;
}

if ($request_method = POST) {
    return 405;
}

if ($slow) {
    limit_rate 10k;
}

if ($invalid_referer) {
    return 403;
}
```

Примечание

Значение встроенной переменной `$invalid_referer` задается директивой `valid_referers`.

return

Синтаксис	<code>return код [текст];</code> <code>return код URL;</code> <code>return URL;</code>
По умолчанию	—
Контекст	server, location, if

Завершает обработку и возвращает клиенту указанный код. Нестандартный код 444 закрывает соединение без передачи заголовка ответа.

Можно задать либо URL перенаправления (для кодов 301, 302, 303, 307 и 308) либо текст тела ответа (для остальных кодов). В тексте тела ответа и URL перенаправления можно использовать переменные. Как частный случай, URL перенаправления может быть задан как URI, локальный для данного сервера, при этом полный URL перенаправления формируется согласно схеме запроса (`$scheme`) и директивам `server_name_in_redirect` и `port_in_redirect`.

Кроме того, в качестве единственного параметра можно указать URL для временного перенаправления с кодом 302. Такой параметр должен начинаться со строк `http://`, `https://` или `"$scheme"`. В URL можно использовать переменные.

См. также директиву `error_page`.

rewrite

Синтаксис	<code>rewrite regex замена [флаг];</code>
По умолчанию	—
Контекст	server, location, if

Если указанное регулярное выражение `regex` соответствует URI запроса, URI изменяется в соответствии со строкой `замены`. Директивы `rewrite` выполняются последовательно, в порядке их следования в конфигурационном файле. С помощью `флага` можно прекратить дальнейшую обработку директив. Если строка `замены` начинается с `http://`, `https://` или `$scheme`, то обработка завершается и клиенту возвращается перенаправление.

В строке `замены` можно использовать переменные, в том числе позиционные переменные от `$1` до `$9`.

Необязательный параметр `флаг` может быть задан одним из следующих значений:

last	завершает обработку текущего набора директив модуля <code>http_rewrite</code> , после чего ищется новый <code>location</code> , соответствующий измененному URI;
break	завершает обработку текущего набора директив модуля <code>http_rewrite</code> аналогично директиве <code>break</code> ;
redirect	возвращает временное перенаправление с кодом 302; используется, если строка <code>замены</code> не начинается с <code>http://</code> , <code>https://</code> или <code>"\$scheme"</code> ;
permanent	возвращает постоянное перенаправление с кодом 301.

Полный URL перенаправлений формируется согласно схеме запроса (`$scheme`) и директив `server_name_in_redirect` и `port_in_redirect`.

Пример:

```
server {
#   ...
    rewrite ^(/download/.*)/media/(.*)\..*$ $1/mp3/$2.mp3 last;
    rewrite ^(/download/.*)/audio/(.*)\..*$ $1/mp3/$2.ra last;
    return 403;
#   ...
}
```

Если же эти директивы поместить в location `/download/`, то нужно заменить флаг `last` на `break`, иначе Angie сделает 10 циклов и вернет ошибку 500:

```
location /download/ {
    rewrite ^(/download/.*)/media/(.*)\..*$ $1/mp3/$2.mp3 break;
    rewrite ^(/download/.*)/audio/(.*)\..*$ $1/mp3/$2.ra break;
    return 403;
}
```

Если в строке замены указаны новые аргументы запроса, то предыдущие аргументы запроса добавляются после них. Если такое поведение нежелательно, можно отказаться от этого добавления, указав в конце строки замены знак вопроса, например:

```
rewrite ^/users/(.*)$ /show?user=$1? last;
```

Если в регулярном выражении встречаются символы `"}` или `;"`, то все выражение следует заключить в одинарные или двойные кавычки.

rewrite_log

Синтаксис	<code>rewrite_log on off;</code>
По умолчанию	<code>rewrite_log off;</code>
Контекст	http, server, location, if

Разрешает или запрещает записывать в `error_log` на уровне `notice` результаты обработки директив модуля `http_rewrite`.

set

Синтаксис	<code>set \$переменная значение;</code>
По умолчанию	—
Контекст	server, location, if

Устанавливает значение указанной переменной. В качестве значения можно использовать переменные.

uninitialized_variable_warn

Синтаксис	<code>uninitialized_variable_warn on off;</code>
По умолчанию	<code>uninitialized_variable_warn on;</code>
Контекст	http, server, location, if

Определяет, нужно ли писать в лог предупреждения о неинициализированных переменных.

Внутреннее устройство

Директивы модуля `http_rewrite` компилируются на стадии конфигурации во внутренние инструкции, интерпретируемые во время обработки запроса. Интерпретатор представляет из себя простую стековую виртуальную машину.

Например, директивы

```
location /download/ {
    if ($forbidden) {
        return 403;
    }

    if ($slow) {
        limit_rate 10k;
    }

    rewrite ^/(download/.*)/media/(.*)\..*$ /$1/mp3/$2.mp3 break;
}
```

будут транслированы в такие инструкции:

```
переменная $forbidden
проверка на ноль
    возврат 403
    завершение всего кода
переменная $slow
проверка на ноль
проверка регулярного выражения
копирование "/"
копирование $1
копирование "/mp3/"
копирование $2
копирование ".mp3"
завершение регулярного выражения
завершение всего кода
```

Обратите внимание, что инструкций для директивы `limit_rate` нет, поскольку она не имеет отношения к модулю `http_rewrite`. Для блока `if` создается отдельная конфигурация. Если условие истинно, запрос получает эту конфигурацию, и в ней `limit_rate` равен 10k.

Директиву

```
rewrite ^/(download/.*)/media/(.*)\..*$ /$1/mp3/$2.mp3 break;
```

можно сделать на одну инструкцию меньше, если в регулярном выражении перенести первую косую черту внутрь скобок:

```
rewrite ^/(download/.*)/media/(.*)\..*$ $1/mp3/$2.mp3 break;
```

Тогда соответствующие инструкции будут выглядеть так:

```
проверка регулярного выражения
копирование $1
копирование "/mp3/"
копирование $2
копирование ".mp3"
```

завершение регулярного выражения
завершение всего кода

SCGI

Позволяет передавать запросы SCGI-серверу.

Пример конфигурации

```
location / {
    include    scgi_params;
    scgi_pass  localhost:9000;
}
```

Директивы

scgi_bind

Синтаксис	<code>scgi_bind <i>адрес</i> [transparent] off;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт локальный IP-адрес с необязательным портом, который будет использоваться в исходящих соединениях с SCGI-сервером. В значении параметра можно использовать переменные. Специальное значение `off` отменяет действие унаследованной с предыдущего уровня конфигурации директивы `scgi_bind`, позволяя системе самостоятельно выбирать локальный IP-адрес и порт.

По умолчанию это значение вычисляется один раз при инициализации запроса, до выбора SCGI-сервера. В Angie PRO вычисление происходит после каждого выбора SCGI-сервера (в том числе при повторных попытках), поэтому адрес привязки может зависеть от выбранного сервера; например, с помощью переменной `$upstream_current_addr`. Если вычисленное значение пусто, привязка не выполняется.

Параметр `transparent` позволяет задать нелокальный IP-адрес, который будет использоваться в исходящих соединениях с SCGI-сервером, например, реальный IP-адрес клиента:

```
scgi_bind $remote_addr transparent;
```

Для работы параметра обычно требуется запустить рабочие процессы Angie с привилегиями суперпользователя. В Linux это не требуется, так как если указан параметр `transparent`, то рабочие процессы наследуют *capability CAP_NET_RAW* из главного процесса.

Примечание

Необходимо настроить таблицу маршрутизации ядра для перехвата сетевого трафика с SCGI-сервера.

scgi_buffer_size

Синтаксис	<code>scgi_buffer_size размер;</code>
По умолчанию	<code>scgi_buffer_size 4k 8k;</code>
Контекст	http, server, location

Задаёт размер буфера, в который будет читаться первая часть ответа, получаемого от SCGI-сервера. В этой части ответа обычно находится небольшой заголовок ответа. По умолчанию размер одного буфера равен размеру страницы памяти. В зависимости от платформы это или 4К, или 8К, однако его можно сделать меньше.

scgi_buffering

Синтаксис	<code>scgi_buffering on off;</code>
По умолчанию	<code>scgi_buffering on;</code>
Контекст	<code>http, server, location</code>

Разрешает или запрещает использовать буферизацию ответов SCGI-сервера.

on	Angie отправляет ответ клиенту параллельно с чтением ответа от SCGI-сервера; клиент не ждёт ответа целиком. Ответ читается в буферы, заданные директивами <code>scgi_buffer_size</code> и <code>scgi_buffers</code> . Буфер передается на отправку, как только заполнится целиком, причем суммарный размер буферов, занятых отправкой, ограничен значением <code>scgi_busy_buffers_size</code> . Заполненный частично буфер ждёт, пока не заполнится или пока не закончится ответ, поэтому данные обычно попадают к клиенту целыми буферами. Часть ответа записывается на диск во временный файл, когда все буферы заполнены, а клиент читает медленнее, чем отдаёт SCGI-сервер. Ответы, которые кэшируются или сохраняются на диске, всегда проходят через временный файл. Запись во временные файлы контролируется директивами <code>scgi_max_temp_file_size</code> и <code>scgi_temp_file_write_size</code> . При буферизации SCGI-сервер освобождается сразу, как только отдаст ответ целиком: соединение с ним не удерживается все то время, пока ответ читает медленный клиент.
off	Каждая порция данных, прочитанная от SCGI-сервера, сразу же передается клиенту, без какого-либо минимального размера: прочитанные 10 байт из буфера 4К отправляются немедленно. Angie использует один буфер, заданный директивой <code>scgi_buffer_size</code> , поэтому за один раз он читает не больше этого размера. Временный файл не используется, поэтому ответы не кэшируются и не сохраняются на диске, а <code>scgi_limit_rate</code> и <code>limit_rate</code> не действуют. Режим off подходит для ответов, которые должны доходить до клиента по мере их формирования: Server-Sent Events, длинные опросы (long polling), потоковая передача.

Буферизация может быть также включена или выключена путем передачи значения "yes" или "no" в поле `X-Accel-Buffering` заголовка ответа. Эту возможность можно запретить с помощью директивы `scgi_ignore_headers`.

scgi_buffers

Синтаксис	<code>scgi_buffers число размер;</code>
По умолчанию	<code>scgi_buffers 8 4k 8k;</code>
Контекст	<code>http, server, location</code>

Задаёт число и размер буферов для одного соединения, в которые будет читаться ответ, получаемый от SCGI-сервера.

По умолчанию размер одного буфера равен размеру страницы. В зависимости от платформы это или 4К, или 8К.

scgi_busy_buffers_size

Синтаксис	scgi_busy_buffers_size размер;
По умолчанию	scgi_busy_buffers_size 8k 16k;
Контекст	http, server, location

При включенной буферизации ответов SCGI-сервера, ограничивает суммарный размер буферов, которые могут быть заняты для отправки ответа клиенту, пока ответ еще не прочитан целиком. Оставшиеся буферы тем временем могут использоваться для чтения ответа и, при необходимости, буферизации части ответа во временный файл.

По умолчанию размер ограничен величиной двух буферов, заданных директивами *scgi_buffer_size* и *scgi_buffers*.

scgi_cache

Синтаксис	scgi_cache зона off;
По умолчанию	scgi_cache off;
Контекст	http, server, location

Задаёт зону разделяемой памяти, используемой для кэширования. Одна и та же зона может использоваться в нескольких местах. В значении параметра можно использовать переменные.

off	запрещает кэширование, унаследованное с предыдущего уровня конфигурации.
-----	--

Примечание

При использовании *scgi_cache* обычно необходимо также задать директиву *scgi_cache_valid*, чтобы явно указать время действия кэшированных ответов. Если она не задана, Angie не использует значения по умолчанию, а определяет время хранения ответа в кэше на основе HTTP-заголовков ответа от сервера в следующем порядке приоритета:

1. Заголовок X-Accel-Expires (наивысший приоритет).
2. Заголовок Cache-Control с параметрами max-age или s-maxage.
3. Заголовок Expires.

Если ни один заголовок не содержит допустимого значения или их нет вовсе, ответ не будет кэшироваться, так как определить срок его действия нельзя.

scgi_cache_background_update

Синтаксис	scgi_cache_background_update on off;
По умолчанию	scgi_cache_background_update off;
Контекст	http, server, location

Позволяет запустить фоновый подзапрос для обновления просроченного элемента кэша, в то время как клиенту возвращается устаревший кэшированный ответ.

Предупреждение

Использование устаревшего кэшированного ответа в момент его обновления должно быть разрешено.

scgi_cache_bypass

Синтаксис	<code>scgi_cache_bypass ...;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт условия, при которых ответ не будет браться из кэша. Если значение хотя бы одного из строковых параметров непустое и не равно "0", то ответ не берётся из кэша:

```
scgi_cache_bypass $cookie_nocache $arg_nocache$arg_comment;  
scgi_cache_bypass $http_pragma $http_authorization;
```

В значении параметров можно использовать переменные.

Можно использовать совместно с директивой `scgi_no_cache`.

scgi_cache_key

Синтаксис	<code>scgi_cache_key строка;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт ключ для кэширования. В значении ключа можно использовать переменные. Например,

```
scgi_cache_key localhost:9000$request_uri;
```

scgi_cache_lock

Синтаксис	<code>scgi_cache_lock on off;</code>
По умолчанию	<code>scgi_cache_lock off;</code>
Контекст	http, server, location

Если включено, одновременно только одному запросу будет позволено заполнить новый элемент кэша, идентифицируемый согласно директиве `scgi_cache_key`, передав запрос на SCGI-сервер. Остальные запросы этого же элемента будут либо ожидать появления ответа в кэше, либо освобождения блокировки этого элемента, в течение времени, заданного директивой `scgi_cache_lock_timeout`.

scgi_cache_lock_age

Синтаксис	<code>scgi_cache_lock_age время;</code>
По умолчанию	<code>scgi_cache_lock_age 5s;</code>
Контекст	http, server, location

Если последний запрос, переданный на SCGI-сервер для заполнения нового элемента кэша, не завершился за указанное время, на SCGI-сервер может быть передан еще один запрос.

`scgi_cache_lock_timeout`

Синтаксис	<code>scgi_cache_lock_timeout</code> время;
По умолчанию	<code>scgi_cache_lock_timeout 5s</code> ;
Контекст	http, server, location

Задаёт таймаут для `scgi_cache_lock`. По истечении указанного времени запрос будет передан на SCGI-сервер, однако ответ не будет кэширован.

`scgi_cache_max_range_offset`

Синтаксис	<code>scgi_cache_max_range_offset</code> <i>число</i> ;
По умолчанию	—
Контекст	http, server, location

Задаёт смещение в байтах для запросов с указанием диапазона запрашиваемых байт (byte-range requests). Если диапазон находится за указанным смещением, range-запрос будет передан на SCGI-сервер и ответ не будет кэширован.

`scgi_cache_methods`

Синтаксис	<code>scgi_cache_methods</code> GET HEAD POST ...;
По умолчанию	<code>scgi_cache_methods</code> GET HEAD;
Контекст	http, server, location

Если метод запроса клиента указан в этой директиве, то ответ будет кэширован. Методы "GET" и "HEAD" всегда добавляются в список, но тем не менее рекомендуется перечислять их явно. См. также директиву `scgi_no_cache`.

`scgi_cache_min_uses`

Синтаксис	<code>scgi_cache_min_uses</code> <i>число</i> ;
По умолчанию	<code>scgi_cache_min_uses 1</code> ;
Контекст	http, server, location

Задаёт число запросов, после которого ответ будет кэширован.

Предупреждение

Метаданные кэша хранятся в разделяемой памяти. Ручное удаление файлов кэша не сбрасывает счетчики и может привести к непредсказуемому поведению. Для полного сброса остановите сервер, удалите директорию кэша и запустите снова.

i Примечание

Сторонние модули очистки кэша (например, Cache Purge) удаляют только файлы, но не сбрасывают счетчик `scgi_cache_min_uses`. Директива предназначена для защиты кэша от загрязнения редкими запросами, и сброс счетчика при очистке может негативно повлиять на производительность.

scgi_cache_path

Синтаксис	<code>scgi_cache_path <i>путь</i> [levels=<i>уровни</i>] [use_temp_path=on off] keys_zone=<i>имя:размер</i>[:file=<i>файл</i>] [inactive=<i>время</i>] [max_size=<i>размер</i>] [min_free=<i>размер</i>] [manager_files=<i>число</i>] [manager_sleep=<i>время</i>] [manager_threshold=<i>время</i>] [loader_files=<i>число</i>] [loader_sleep=<i>время</i>] [loader_threshold=<i>время</i>];</code>
По умолчанию	—
Контекст	http

Задаёт путь и другие параметры кэша. Данные кэша хранятся в файлах. Именем файла в кэше является результат функции MD5 от ключа кэширования.

levels	задаёт уровни иерархии кэша: можно задать от 1 до 3 уровней, на каждом уровне допускаются значения 1 или 2.
---------------	---

Например, при использовании

```
scgi_cache_path /data/angie/cache levels=1:2 keys_zone=one:10m;
```

имена файлов в кэше будут такого вида:

```
/data/angie/cache/c/29/b7f54b2df7773722d382f4809d65029c
```

Кэшируемый ответ сначала записывается во временный файл, а потом этот файл переименовывается. Временные файлы и кэш могут располагаться на разных файловых системах. Однако нужно учитывать, что в этом случае вместо дешевой операции переименовывания в пределах одной файловой системы файл копируется с одной файловой системы на другую. Поэтому лучше, если кэш будет находиться на той же файловой системе, что и каталог с временными файлами.

<code>use_temp_path=on</code>	определяет каталог, который будет использоваться для временных файлов
<code> off</code>	
<code>on</code>	Если параметр не задан или установлен в значение <code>on</code> , будет использоваться каталог, задаваемый директивой <code>scgi_temp_path</code> для данного <code>location</code>
<code>off</code>	временные файлы будут располагаться непосредственно в каталоге кэша
<code>keys_zone</code>	Задаёт имя и размер зоны разделяемой памяти для хранения всех активных ключей и информации о данных. Метаданные кэша хранятся в разделяемой памяти. Размер зоны задается в байтах и должен составлять не менее двух системных страниц памяти. Также допустимы суффиксы <code>k</code> и <code>m</code>; суффикс <code>g</code>, в отличие от <code>max_size</code>, здесь не поддерживается. Зоны размером в 1 мегабайт достаточно для хранения около 8000 ключей. Когда с <code>keys_zone</code> задан необязательный <i>файл</i>, Angie сбрасывает содержимое этой зоны на диск при завершении работы основного процесса и пытается восстановить ее по тому же адресу памяти при следующем запуске или после обновления бинарных файлов, чтобы обеспечить более надежную сохранность данных и сократить время загрузки кэша. Если восстановление области невозможно из-за изменения ее размера, несовместимости версий бинарных файлов или по другим причинам, Angie запишет в журнал предупреждение (<code>failed to restore zone at address</code>) и не будет использовать механизм восстановления зоны. Вместо этого несовместимый файл будет переименован в <code>.old</code>; вы можете либо удалить его, либо восстановить его имя и вернуть Angie к конфигурации и версии, в которых этот файл был изначально создан.
 Предупреждение Убедитесь, что путь к <i>файлу</i> указан правильно и имеет необходимые права доступа для использования Angie, а также защищен от несанкционированного доступа; относительные пути основаны на префиксе.	
<code>inactive</code>	Если к данным кэша не обращаются в течение времени, заданного этим параметром, они удаляются независимо от их свежести; каждое обращение сбрасывает этот таймер. Это отличается от вытеснения по размеру, выполняемого менеджером кэша (см. ниже): удаление по неактивности происходит, даже когда кэш не достиг <code>max_size</code>, и не зависит от сроков кэширования, заданных директивой <code>scgi_cache_valid</code>. По умолчанию <code>inactive</code> равен 10 минутам.

Специальный процесс **менеджера кэша** следит за максимальным размером кэша, а также за минимальным объемом свободного места на файловой системе с кэшем, и удаляет наименее востребованные данные при превышении максимального размера кэша или недостаточном объеме свободного места. Удаление данных происходит итерациями.

max_size	максимальное пороговое значение размера кэша Значение задается в байтах; допустимы суффиксы k , m и g . По умолчанию размер кэша не ограничен.
min_free	минимальное пороговое значение объема свободного места на файловой системе с кэшем Задается в тех же единицах, что и max_size . По умолчанию 0 — отслеживание свободного места отключено. На платформах, где статистика файловой системы недоступна, параметр игнорируется, а Angie записывает в журнал предупреждение min_free is not supported on this platform, ignored .
manager_files	максимальное количество удаляемых элементов кэша за одну итерацию По умолчанию: 100
manager_threshold	ограничивает время работы одной итерации По умолчанию: 200 миллисекунд
manager_sleep	время, в течение которого выдерживается пауза между итерациями По умолчанию: 50 миллисекунд

Через минуту после старта Angie активируется специальный процесс **загрузчика кэша**, который загружает в зону кэша информацию о ранее кэшированных данных, хранящихся на файловой системе. Загрузка также происходит итерациями.

loader_files	максимальное количество элементов кэша к загрузке в одну итерацию По умолчанию: 100
loader_threshold	ограничивает время работы одной итерации По умолчанию: 200 миллисекунд
loader_sleep	время, в течение которого выдерживается пауза между итерациями По умолчанию: 50 миллисекунд

scgi_cache_revalidate

Синтаксис	scgi_cache_revalidate on off;
По умолчанию	scgi_cache_revalidate off;
Контекст	http, server, location

Разрешает ревалидацию просроченных элементов кэша при помощи условных запросов с полями заголовка **If-Modified-Since** и **If-None-Match**.

scgi_cache_use_stale

Синтаксис	scgi_cache_use_stale error timeout invalid_header updating http_500 http_503 http_403 http_404 http_429 non_idempotent off ...;
По умолчанию	scgi_cache_use_stale off;
Контекст	http, server, location

Определяет, в каких случаях можно использовать устаревший кэшированный ответ. Параметры директивы совпадают с параметрами директивы *scgi_next_upstream*.

error	Позволяет использовать устаревший кэшированный ответ при невозможности выбора SCGI-сервера для обработки запроса.
updating	Дополнительный параметр, разрешает использовать устаревший кэшированный ответ, если на данный момент он уже обновляется. Это позволяет минимизировать число обращений к SCGI-серверам при обновлении кэшированных данных.
non_idempotent	Позволяет использовать устаревший кэшированный ответ, если запрос с неидемпотентным методом уже был отправлен SCGI-серверу, а при дальнейшей работе с этим сервером произошла любая ошибка; из неидемпотентных методов кэшировать можно только POST, и он должен быть разрешен директивой <code>scgi_cache_methods</code> .

Использование устаревшего кэшированного ответа может также быть разрешено непосредственно в заголовке ответа на определенное количество секунд после того, как ответ устарел:

- Расширение `stale-while-revalidate` поля заголовка **Cache-Control** разрешает использовать устаревший кэшированный ответ, если на данный момент он уже обновляется.
- Расширение `stale-if-error` поля заголовка **Cache-Control** разрешает использовать устаревший кэшированный ответ в случае ошибки.

Примечание

Такой способ менее приоритетен, чем задание параметров директивы.

Чтобы минимизировать число обращений к SCGI-серверам при заполнении нового элемента кэша, можно воспользоваться директивой `scgi_cache_lock`.

scgi_cache_valid

Синтаксис	<code>scgi_cache_valid [код ...] время;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт время кэширования для разных кодов ответа. Например, директивы

```
scgi_cache_valid 200 302 10m;
scgi_cache_valid 404 1m;
```

задают время кэширования 10 минут для ответов с кодами 200 и 302 и 1 минуту для ответов с кодом 404.

Если указано только время кэширования,

```
scgi_cache_valid 5m;
```

то кэшируются только ответы 200, 301 и 302.

Кроме того, можно кэшировать любые ответы с помощью параметра `any`:

```
scgi_cache_valid 200 302 10m;
scgi_cache_valid 301 1h;
scgi_cache_valid any 1m;
```

Примечание

Параметры кэширования могут также быть заданы непосредственно в заголовке ответа. Такой способ приоритетнее, чем задание времени кэширования с помощью директивы.

- Поле заголовка **X-Accel-Expires** задает время кэширования ответа в секундах. Значение *0* запрещает кэшировать ответ. Если значение начинается с префикса *@*, оно задает абсолютное время в секундах с начала эпохи, до которого ответ может быть кэширован.
- Если в заголовке нет поля **X-Accel-Expires**, параметры кэширования определяются по полям заголовка **Expires** или **Cache-Control**.
- Ответ, в заголовке которого есть поле **Set-Cookie**, не будет кэшироваться.
- Ответ, в заголовке которого есть поле **Vary** со специальным значением *"*"*, не будет кэшироваться. Ответ, в заголовке которого есть поле **Vary** с другим значением, будет кэширован с учетом соответствующих полей заголовка запроса.

Обработка одного или более из этих полей заголовка может быть отключена при помощи директивы *scgi_ignore_headers*.

scgi_connect_timeout

Синтаксис	<code>scgi_connect_timeout время;</code>
По умолчанию	<code>scgi_connect_timeout 60s;</code>
Контекст	<code>http, server, location</code>

Задаёт таймаут для установления соединения с SCGI-сервером. Необходимо иметь в виду, что этот таймаут обычно не может превышать 75 секунд.

scgi_connection_drop

Синтаксис	<code>scgi_connection_drop время on off;</code>
По умолчанию	<code>scgi_connection_drop off;</code>
Контекст	<code>http, server, location</code>

Настраивает завершение всех активных соединений с проксируемым сервером, если он был удален из группы или помечен как постоянно недоступный в результате процесса *resolver* или команды **API DELETE**. Неактивные соединения в кэше не затрагиваются.

Соединение завершается, когда обрабатывается следующее событие чтения или записи для клиента или проксируемого сервера.

Установка *времени* включает таймаут до завершения соединения; при выборе значения *on* соединения завершаются немедленно.

scgi_force_ranges

Синтаксис	<code>scgi_force_ranges off;</code>
По умолчанию	<code>scgi_force_ranges off;</code>
Контекст	<code>http, server, location</code>

Включает поддержку диапазонов запрашиваемых байт (byte-range) для кэшированных и некешированных ответов SCGI-сервера вне зависимости от наличия поля **Accept-Ranges** в заголовках этих ответов.

scgi_hide_header

Синтаксис	<code>scgi_hide_header поле;</code>
По умолчанию	—
Контекст	http, server, location

По умолчанию Angie не передает клиенту поля заголовка **Status** и **X-Accel-...** из ответа SCGI-сервера. Директива `scgi_hide_header` задает дополнительные поля, которые не будут передаваться. Если же передачу полей нужно разрешить, можно воспользоваться директивой `scgi_pass_header`.

scgi_ignore_client_abort

Синтаксис	<code>scgi_ignore_client_abort on off;</code>
По умолчанию	<code>scgi_ignore_client_abort off;</code>
Контекст	http, server, location

Определяет, закрывать ли соединение с SCGI-сервером в случае, если клиент закрыл соединение, не дождавшись ответа.

scgi_ignore_headers

Синтаксис	<code>scgi_ignore_headers поле ...;</code>
По умолчанию	—
Контекст	http, server, location

Запрещает обработку некоторых полей заголовка из ответа SCGI-сервера. В директиве можно указать поля **X-Accel-Redirect**, **X-Accel-Expires**, **X-Accel-Limit-Rate**, **X-Accel-Buffering**, **X-Accel-Charset**, **Expires**, **Cache-Control**, **Set-Cookie** и **Vary**.

Если не запрещено, обработка этих полей заголовка заключается в следующем:

- **X-Accel-Expires**, **Expires**, **Cache-Control**, **Set-Cookie** и **Vary** задают параметры кэширования ответа;
- **X-Accel-Redirect** производит внутреннее перенаправление на указанный URI;
- **X-Accel-Limit-Rate** задает ограничение скорости передачи ответа клиенту;
- **X-Accel-Buffering** включает или выключает буферизацию ответа;
- **X-Accel-Charset** задает желаемую кодировку ответа.

scgi_intercept_errors

Синтаксис	<code>scgi_intercept_errors on off;</code>
По умолчанию	<code>scgi_intercept_errors off;</code>
Контекст	http, server, location

Определяет, передавать ли клиенту ответы SCGI-сервера с кодом больше либо равным 300, или же перехватывать их и перенаправлять на обработку Angie с помощью директивы *error_page*.

scgi_interface

Синтаксис	<code>scgi_interface имя;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт сетевой интерфейс, который будет использоваться в исходящих соединениях с SCGI-сервером; сокет привязывается к нему параметром сокета `SO_BINDTODEVICE`.

scgi_limit_rate

Синтаксис	<code>scgi_limit_rate скорость;</code>
По умолчанию	<code>scgi_limit_rate 0;</code>
Контекст	http, server, location

Ограничивает скорость чтения ответа от SCGI-сервера. *Скорость* задается в байтах в секунду; можно использовать переменные.

0	отключает ограничение скорости
---	--------------------------------

Примечание

Ограничение устанавливается на запрос, поэтому, если Angie одновременно откроет два соединения к SCGI-серверу, суммарная скорость будет вдвое выше заданного ограничения. Ограничение работает только в случае, если включена буферизация ответов SCGI-сервера.

scgi_max_temp_file_size

Синтаксис	<code>scgi_max_temp_file_size размер;</code>
По умолчанию	<code>scgi_max_temp_file_size 1024m;</code>
Контекст	http, server, location

Если включена буферизация ответов SCGI-сервера, и ответ не помещается целиком в буферы, заданные директивами *scgi_buffer_size* и *scgi_buffers*, часть ответа может быть записана во временный файл. Эта директива задает максимальный размер временного файла. Размер данных, сбрасываемых во временный файл за один раз, задается директивой *scgi_temp_file_write_size*.

0	отключает возможность буферизации ответов во временные файлы
---	--

Примечание

Данное ограничение не распространяется на ответы, которые будут кэшированы или сохранены на диске.

scgi_next_upstream

Синтаксис	scgi_next_upstream error timeout invalid_header http_500 http_503 http_403 http_404 http_429 non_idempotent off ...;
По умолчанию	scgi_next_upstream error timeout;
Контекст	http, server, location

Определяет, в каких случаях запрос будет передан следующему серверу:

error	произошла ошибка соединения с сервером, передачи ему запроса или чтения заголовка ответа сервера;
timeout	произошел таймаут во время соединения с сервером, передачи ему запроса или чтения заголовка ответа сервера;
invalid_header	сервер вернул пустой или неверный ответ;
http_500	сервер вернул ответ с кодом 500;
http_503	сервер вернул ответ с кодом 503;
http_403	сервер вернул ответ с кодом 403;
http_404	сервер вернул ответ с кодом 404;
http_429	сервер вернул ответ с кодом 429;
non_idempotent	обычно запросы с неидемпотентным методом (<i>POST</i> , <i>LOCK</i> , <i>PATCH</i>) не передаются на другой сервер, если запрос серверу группы уже был отправлен; включение параметра явно разрешает повторять подобные запросы;
off	запрещает передачу запроса следующему серверу.

Примечание

Необходимо понимать, что передача запроса следующему серверу возможна только при условии, что клиенту еще ничего не передавалось. То есть, если ошибка или таймаут возникли в середине передачи ответа клиенту, то действие директивы на такой запрос не распространяется.

Директива также определяет, что считается неудачной попыткой работы с сервером.

error	всегда считаются неудачными попытками, даже если они не указаны в директиве
timeout	
invalid_header	
http_500	считаются неудачными попытками, только если они указаны в директиве
http_503	
http_429	
http_403	никогда не считаются неудачными попытками
http_404	

Передача запроса следующему серверу может быть ограничена по количеству попыток и по времени.

scgi_next_upstream_timeout

Синтаксис	scgi_next_upstream_timeout время;
По умолчанию	scgi_next_upstream_timeout 0;
Контекст	http, server, location

Ограничивает время, в течение которого возможна передача запроса следующему серверу.

0	отключает это ограничение
---	---------------------------

scgi_next_upstream_tries

Синтаксис	<code>scgi_next_upstream_tries <i>число</i>;</code>
По умолчанию	<code>scgi_next_upstream_tries 0;</code>
Контекст	http, server, location

Ограничивает число допустимых попыток для передачи запроса следующему серверу.

0	отключает это ограничение
---	---------------------------

scgi_no_cache

Синтаксис	<code>scgi_no_cache <i>строка</i> ...;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт условия, при которых ответ не будет сохраняться в кэш. Если значение хотя бы одного из строковых параметров непустое и не равно "0", то ответ не будет сохранен:

```
scgi_no_cache $cookie_nocache $arg_nocache$arg_comment;
scgi_no_cache $http_pragma $http_authorization;
```

В значении параметров можно использовать переменные.

Можно использовать совместно с директивой `scgi_cache_bypass`.

scgi_param

Синтаксис	<code>scgi_param <i>параметр значение</i> [if_not_empty];</code>
По умолчанию	—
Контекст	http, server, location

Задаёт параметр, который будет передаваться SCGI-серверу. В качестве значения можно использовать переменные. Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы `SCGI-param`.

Стандартные переменные окружения CGI должны передаваться как заголовки SCGI, см. файл `scgi_params` из дистрибутива:

```
location / {
    include scgi_params;
    # ...
}
```


В стандартном файле `scgi_params` параметр `REQUEST_METHOD` задается как `$upstream_request_method`.

Если директива указана с `if_not_empty`, то такой параметр с пустым значением передаваться на сервер не будет:

```
scgi_param HTTPS $https if_not_empty;
```

scgi_pass

Синтаксис	<code>scgi_pass uri;</code>
По умолчанию	—
Контекст	location, if в location

Задаёт адрес SCGI-сервера. Адрес может быть указан в виде доменного имени или IP-адреса, и порта:

```
scgi_pass localhost:9000;
```

или в виде пути UNIX-сокета:

```
scgi_pass unix:/tmp/scgi.socket;
```

Если доменному имени соответствует несколько адресов, то все они будут использоваться по очереди (round-robin). Кроме того, в качестве адреса можно указать группу серверов.

В значении параметра можно использовать переменные. В этом случае, если адрес указан в виде доменного имени, имя ищется среди описанных групп серверов и если не найдено, то определяется с помощью *resolver*'а.

Примечание

Если `scgi_pass` стоит в `location` с косой чертой в конце префикса (например, `location /name/`), и при этом в директиве `auto_redirect` указано `default`, запросы без косой черты в конце будут перенаправляться (`/name -> /name/`).

scgi_pass_header

Синтаксис	<code>scgi_pass_header поле ...;</code>
По умолчанию	—
Контекст	http, server, location

Разрешает передавать от SCGI-сервера клиенту запрещенные для передачи поля заголовка.

scgi_pass_request_body

Синтаксис	<code>scgi_pass_request_body on off;</code>
По умолчанию	<code>scgi_pass_request_body on;</code>
Контекст	http, server, location

Позволяет запретить передачу исходного тела запроса на SCGI-сервер. См. также директиву *scgi_pass_request_headers*.

scgi_pass_request_headers

Синтаксис	<code>scgi_pass_request_headers on off;</code>
По умолчанию	<code>scgi_pass_request_headers on;</code>
Контекст	http, server, location

Позволяет запретить передачу полей заголовка исходного запроса на SCGI-сервер. См. также директиву *scgi_pass_request_body*.

scgi_read_timeout

Синтаксис	<code>scgi_read_timeout время;</code>
По умолчанию	<code>scgi_read_timeout 60s;</code>
Контекст	http, server, location

Задаёт таймаут при чтении ответа SCGI-сервера. Таймаут устанавливается не на всю передачу ответа, а только между двумя операциями чтения. Если по истечении этого времени SCGI-сервер ничего не передаст, соединение закрывается.

scgi_request_buffering

Синтаксис	<code>scgi_request_buffering on off;</code>
По умолчанию	<code>scgi_request_buffering on;</code>
Контекст	http, server, location

Разрешает или запрещает использовать буферизацию тела запроса клиента.

on	тело запроса полностью читается от клиента перед отправкой запроса на SCGI-сервер.
off	тело запроса отправляется на SCGI-сервер сразу же по мере его поступления. В этом случае запрос не может быть передан следующему серверу, если Angie уже начал отправку тела запроса.

Если для отправки тела исходного запроса используется HTTP/1.1 и передача данных частями (chunked transfer encoding), то тело запроса буферизуется независимо от значения директивы.

scgi_send_timeout

Синтаксис	<code>scgi_send_timeout время;</code>
По умолчанию	<code>scgi_send_timeout 60s;</code>
Контекст	http, server, location

Задаёт таймаут при передаче запроса SCGI-серверу. Таймаут устанавливается не на всю передачу запроса, а только между двумя операциями записи. Если по истечении этого времени SCGI-сервер не примет новых данных, соединение закрывается.

scgi_socket_keepalive

Синтаксис	<code>scgi_socket_keepalive on off;</code>
По умолчанию	<code>scgi_socket_keepalive off;</code>
Контекст	<code>http, server, location</code>

Конфигурирует поведение "TCP keepalive" для исходящих соединений к SCGI-серверу.

" "	По умолчанию для сокета действуют настройки операционной системы.
on	для сокета включается параметр <code>SO_KEEPALIVE</code>

scgi_store

Синтаксис	<code>scgi_store on off строка;</code>
По умолчанию	<code>scgi_store off;</code>
Контекст	<code>http, server, location</code>

Разрешает сохранение на диск файлов.

on	сохраняет файлы в соответствии с путями, указанными в директивах <i>alias</i> или <i>root</i>
off	запрещает сохранение файлов

Имя файла можно задать явно с помощью строки с переменными:

```
scgi_store /data/www$original_uri;
```

Время изменения файлов выставляется согласно полученному полю `Last-Modified` в заголовке ответа. Ответ сначала записывается во временный файл, а потом этот файл переименовывается. Временный файл и постоянное место хранения ответа могут располагаться на разных файловых системах. Однако нужно учитывать, что в этом случае вместо дешевой операции переименовывания в пределах одной файловой системы файл копируется с одной файловой системы на другую. Поэтому лучше, если сохраняемые файлы будут находиться на той же файловой системе, что и каталог с временными файлами, задаваемый директивой `scgi_temp_path` для данного *location*.

Директиву можно использовать для создания локальных копий статических неизменяемых файлов:

```
location /images/ {
    root          /data/www;
    error_page    404 = /fetch$uri;
}

location /fetch/ {
    internal;

    scgi_pass     backend:9000;
    # ...

    scgi_store    on;
    scgi_store_access user:rw group:rw all:r;
    scgi_temp_path /data/temp;
```

```
alias /data/www/;
}
```

scgi_store_access

Синтаксис	scgi_store_access <i>пользователи:права ...</i> ;
По умолчанию	scgi_store_access user:rw;
Контекст	http, server, location

Задаёт права доступа для создаваемых файлов и каталогов, например,

```
scgi_store_access user:rw group:rw all:r;
```

Если заданы какие-либо права для *group* или *all*, то права для *user* указывать необязательно:

```
scgi_store_access group:rw all:r;
```

scgi_tcp_congestion

Синтаксис	scgi_tcp_congestion <i>алгоритм</i> ;
По умолчанию	—
Контекст	http, server, location

Задаёт алгоритм управления перегрузкой TCP (параметр сокета TCP_CONGESTION) для исходящих соединений с SCGI-сервером; к соединениям через UNIX-сокеты параметр не применяется.

Имя алгоритма не проверяется на этапе конфигурации: если алгоритм неизвестен или недоступен, при установке исходящего соединения в лог записывается сообщение уровня **alert**, а для соединения действует системный алгоритм по умолчанию.

scgi_temp_file_write_size

Синтаксис	scgi_temp_file_write_size размер;
По умолчанию	scgi_temp_file_write_size 8k 16k;
Контекст	http, server, location

Ограничивает размер данных, сбрасываемых во временный файл за один раз, при включенной буферизации ответов SCGI-сервера во временные файлы. По умолчанию размер ограничен двумя буферами, заданными директивами *scgi_buffer_size* и *scgi_buffers*. Максимальный размер временного файла задается директивой *scgi_max_temp_file_size*.

scgi_temp_path

Синтаксис	scgi_temp_path <i>путь</i> [<i>уровень1</i> [<i>уровень2</i> [<i>уровень3</i>]]]`;
По умолчанию	scgi_temp_path scgi_temp; (путь зависит от параметра сборки --http-scgi-temp-path)
Контекст	http, server, location

Задаёт имя каталога для хранения временных файлов с данными, полученными от SCGI-серверов. В каталоге может использоваться иерархия подкаталогов до трех уровней. Например, при такой конфигурации

```
scgi_temp_path /spool/angie/scgi_temp 1 2;
```

временный файл будет следующего вида:

```
/spool/angie/scgi_temp/7/45/00000123457
```

См. также параметр `use_temp_path` директивы `scgi_cache_path`.

Secure Link

Позволяет проверять аутентичность запрашиваемых ссылок, защищать ресурсы от несанкционированного доступа, а также ограничивать срок действия ссылок.

Правильность запрашиваемой ссылки проверяется сравнением переданного в запросе значения контрольной суммы со значением, вычисляемым для запроса. Если ссылка имеет ограниченный срок действия и он истек, ссылка считается устаревшей. Результат этих проверок делается доступным в переменной `$secure_link`.

Модуль реализует два альтернативных режима работы. В первом режиме, который включается директивой `secure_link_secret`, можно проверить аутентичность запрашиваемых ссылок и защитить их от несанкционированного доступа. Второй режим включается директивами `secure_link` и `secure_link_md5`, и позволяет также ограничить срок действия ссылок.

Директивы

secure_link

Синтаксис	<code>secure_link</code> <i>выражение</i> ;
По умолчанию	—
Контекст	http, server, location

Задаёт строку с переменными, из которой будет выделено значение контрольной суммы и время окончания срока действия ссылки.

Используемые в выражении переменные обычно связаны с запросом; см. пример ниже.

Выделенное из строки значение контрольной суммы сравнивается со значением MD5-хэша, вычисляемым для выражения, заданного директивой `secure_link_md5`.

Если контрольные суммы не совпадают, значением переменной `$secure_link` становится пустая строка. Если контрольные суммы совпадают, проверяется время действия ссылки.

Если срок действия ссылки задан и истек, переменная `$secure_link` получает значение 0. В противном случае она получает значение 1. Значение MD5-хэша передается в запросе закодированным в `base64url`.

Если ссылка имеет ограниченный срок действия, время его окончания задается в секундах с начала эпохи (1 января 1970 года 00:00:00 GMT). Значение указывается в выражении после MD5-хэша и отделяется от него запятой. Время окончания срока действия ссылки, переданное в запросе, делается доступным в переменной `$secure_link_expires` для использования в директиве `secure_link_md5`. Если время окончания не задано, ссылка имеет неограниченный срок действия.

secure_link_md5

Синтаксис	<code>secure_link_md5</code> <i>выражение</i> ;
По умолчанию	—
Контекст	http, server, location

Задаёт выражение, для которого считается значение MD5-хэша, сравниваемое с переданным в запросе. В выражении можно использовать переменные.

Выражение должно содержать защищаемую часть ссылки (ресурс) и секретную составляющую. Если ссылка имеет ограниченный срок действия, выражение также должно содержать `$secure_link_expires`.

Для предотвращения несанкционированного доступа выражение может содержать информацию о клиенте, например, его адрес и версию браузера.

Пример:

```
location /s/ {
    secure_link $arg_md5,$arg_expires;
    secure_link_md5 "$secure_link_expires$uri$remote_addr secret";

    if ($secure_link = "") {
        return 403;
    }

    if ($secure_link = "0") {
        return 410;
    }

    # ...
}
```

Ссылка `/s/link?md5=_e4Nc3iduzkWRm01TBbNYw&expires=2147483647` ограничивает доступ к `/s/link` для клиента с IP-адресом 127.0.0.1. Ссылка также имеет ограниченный срок действия до 19 января 2038 года (GMT).

Значение аргумента запроса md5 на UNIX можно получить так:

```
$ echo -n '2147483647/s/link127.0.0.1 secret' | \
  openssl md5 -binary | openssl base64 | tr +/ -_ | tr -d =
```

secure_link_secret

Синтаксис	<code>secure_link_secret</code> <i>слово</i> ;
По умолчанию	—
Контекст	location

Задаёт секретное слово для проверки аутентичности запрашиваемых ссылок.

Полный URI запрашиваемой ссылки выглядит так:

```
/префикс/хэш/ссылка
```

где хэш — MD5-хэш в шестнадцатеричном виде, вычисленный для конкатенации ссылки и секретного слова, а префикс — произвольная строка без косых черт.

Если запрашиваемая ссылка проходит проверку на аутентичность, значением переменной `$secure_link` становится ссылка, выделенная из URI запроса. В противном случае значением переменной `$secure_link` становится пустая строка.

Пример:

```
location /p/ {
    secure_link_secret secret;

    if ($secure_link = "") {
        return 403;
    }

    rewrite ^ /secure/$secure_link;
}

location /secure/ {
    internal;
}
```

По запросу `"/p/5e814704a28d9bc1914ff19fa0c4a00a/link"` будет выполнено внутреннее перенаправление на `"/secure/link"`.

Значение хэша для данного примера на UNIX можно получить так:

```
$ echo -n 'linksecret' | openssl md5 -hex
```

Встроенные переменные

`$secure_link`

Результат проверки ссылки. Конкретное значение зависит от выбранного режима работы.

`$secure_link_expires`

Время окончания срока действия ссылки в секундах с начала эпохи, переданное в запросе. Предназначено исключительно для использования в директиве `secure_link_md5`.

Slice

Фильтр, который разбивает запрос на подзапросы, каждый из которых возвращает определенный диапазон ответа. Фильтр обеспечивает более эффективное кэширование больших ответов.

Пример конфигурации

```
location / {
    slice          1m;
    proxy_cache    cache;
    proxy_cache_key $uri$is_args$args$slice_range;
    proxy_set_header Range $slice_range;
    proxy_cache_valid 200 206 1h;
    proxy_pass      http://localhost:8000;
}
```

В данном примере ответ разбивается на кэшируемые фрагменты размером в 1 мегабайт.

Директивы

slice

Синтаксис	<code>slice</code> размер;
По умолчанию	<code>slice 0</code> ;
Контекст	http, server, location

Задаёт размер фрагмента. Нулевое значение запрещает разбиение ответов на фрагменты.

Предупреждение

Обратите внимание, что слишком низкое значение может привести к излишнему потреблению памяти и открытию большого количества файлов.

Для того, чтобы подзапрос вернул необходимый диапазон, переменная `$slice_range` должна быть передана на проксируемый сервер в качестве поля `Range` заголовка запроса. Если включено кэширование, то необходимо добавить `$slice_range` в ключ кэширования и включить кэширование ответов с кодом 206.

Встроенные переменные

`$slice_range`

текущий диапазон фрагмента в формате HTTP byte range, например `bytes=0-1048575`.

Split Clients

Модуль генерирует переменные для А/В-тестирования, канареечных релизов и других сценариев, которые направляют определенный процент клиентов на один сервер или конфигурацию, а остальных — куда-то еще.

Пример конфигурации

```
http {
    split_clients "${remote_addr}AAA" $variant {
        0.5% .one;
        2.0% .two;
        *    "";
    }

    server {
        location / {
            index index${variant}.html;
        }
    }
}
```

Директивы

split_clients

Синтаксис	<code>split_clients</code> строка \$переменная { ... }
По умолчанию	—
Контекст	http

Создает *\$переменную*, хэшируя *строку*; переменные в *строке* подставляются, результат хэшируется, затем по значению хэша выбирается строковое значение *\$переменной*.

Функция хэширования использует [MurmurHash2](#) (32 бит), и весь диапазон ее значений (с 0 по 4294967295) сопоставляется с корзинами в порядке появления; процентные величины определяют размер корзин. В конце может стоять метасимвол (*); хэши, не попавшие в другие корзины, сопоставляются с приданным ему значением.

Пример:

```
split_clients "${remote_addr}AAA" $variant {
    0.5%          .one;
    2.0%          .two;
    *             "";
}
```

Здесь после подстановки в строке `$remote_addrAAA` значения хэша распределяются следующим образом:

- значения от 0 до 21474835 (0,5%) дают `.one`;
- значения от 21474836 до 107374180 (2%) дают `.two`;
- значения от 107374181 до 4294967295 (все остальные) дают `"` (пустую строку).

SSI

Фильтр, обрабатывающий команды SSI (Server Side Includes) в проходящих через него ответах.

Пример конфигурации

```
location / {
    ssi on;
#    ...
}
```

Директивы

ssi

Синтаксис	<code>ssi on off;</code>
По умолчанию	<code>ssi off;</code>
Контекст	<code>http, server, location, if в location</code>

Разрешает или запрещает обработку команд SSI в ответах.

ssi_last_modified

Синтаксис	<code>ssi_last_modified on off;</code>
По умолчанию	<code>ssi_last_modified off;</code>
Контекст	<code>http, server, location</code>

Позволяет сохранить поле заголовка `Last-Modified` исходного ответа во время обработки SSI для лучшего кэширования ответов.

По умолчанию поле заголовка удаляется, так как содержимое ответа изменяется во время обработки и может содержать динамически созданные элементы или части, которые изменились независимо от исходного ответа.

ssi_min_file_chunk

Синтаксис	<code>ssi_min_file_chunk размер;</code>
По умолчанию	<code>ssi_min_file_chunk 1k;</code>
Контекст	<code>http, server, location</code>

Задаёт минимальный размер частей ответа, хранящихся на диске, начиная с которого имеет смысл посылать их с помощью *sendfile*.

ssi_silent_errors

Синтаксис	<code>ssi_silent_errors on off;</code>
По умолчанию	<code>ssi_silent_errors off;</code>
Контекст	<code>http, server, location</code>

Разрешает не выводить строку *"[an error occurred while processing the directive]"*, если во время обработки SSI произошла ошибка.

ssi_types

Синтаксис	<code>ssi_types mime-mun ...;</code>
По умолчанию	<code>ssi_types text/html;</code>
Контекст	<code>http, server, location</code>

Разрешает обработку команд SSI в ответах с указанными MIME-типами в дополнение к `text/html`. Специальное значение `"*"` соответствует любому MIME-типу.

ssi_value_length

Синтаксис	<code>ssi_value_length длина;</code>
По умолчанию	<code>ssi_value_length 256;</code>
Контекст	<code>http, server, location</code>

Задаёт максимальную длину значений параметров в SSI-командах.

Команды SSI

Общий формат команд SSI такой:

```
<!--# команда параметр1=значение1 параметр2=значение2 ... -->
```

Поддерживаются следующие команды:

block

Описывает блок, который можно использовать как заглушку в команде *include*. Внутри блока могут быть другие команды SSI. Параметр команды:

name

имя блока.

Пример:

```
<!--# block name="one" -->
заглушка
<!--# endblock -->
```

config

Задаёт некоторые параметры, используемые при обработке SSI, а именно:

errmsg

строка, выводимая при ошибке во время обработки SSI. По умолчанию выводится такая строка:

```
`[an error occurred while processing the directive]`
```

timefmt

строка формата, передаваемая функции `strftime()` для вывода даты и времени. По умолчанию используется такой формат:

```
`"%A, %d-%b-%Y %H:%M:%S %Z"`
```

Для вывода времени в секундах подходит формат `"%s"`.

echo

Выводит значение переменной. Параметры команды:

var

имя переменной.

encoding

способ кодирования. Возможны три значения — `none`, `url` и `entity`. По умолчанию используется `entity`.

default

нестандартный параметр, задающий строку, которая выводится, если переменная не определена. По умолчанию выводится строка (`none`).

Команда

```
<!--# echo var="name" default="нет" -->
```

заменяет такую последовательность команд:

```
<!--# if expr="$name" --><!--# echo var="name" --><!--#
    else -->нет<!--# endif -->
```

if

Выполняет условное включение. Поддерживаются следующие команды:

```
<!--# if expr="..." -->
...
<!--# elif expr="..." -->
...
<!--# else -->
...
<!--# endif -->
```

На данный момент поддерживается только один уровень вложенности. Параметр команды:

expr

выражение. В выражении может быть:

- проверка существования переменной:

```
<!--# if expr="$name" -->
```

- сравнение переменной с текстом:

```
<!--# if expr="$name = text" -->
<!--# if expr="$name != text" -->
```

- сравнение переменной с регулярным выражением:

```
<!--# if expr="$name = /text/" -->
<!--# if expr="$name != /text/" -->
```

Если в *text* встречаются переменные, то производится подстановка их значений. В регулярном выражении можно задать позиционные и именованные группы захвата, а затем использовать их через переменные, например:

```
<!--# if expr="$name = /(.)@(?P<domain>.+)/" -->
    <!--# echo var="1" -->
    <!--# echo var="domain" -->
<!--# endif -->
```

include

Включает в ответ результат другого запроса. Параметры команды:

file

задает включаемый файл, например:

```
<!--# include file="footer.html" -->
```

virtual

задает включаемый запрос, например:

```
<!--# include virtual="/remote/body.php?argument=value" -->
```

Несколько запросов, указанных на одной странице и обрабатываемых проксируемыми или FastCGI/uwsgi/SCGI/gRPC-серверами, работают параллельно. Если нужна последовательная обработка, следует воспользоваться параметром *wait*.

stub

нестандартный параметр, задающий имя блока, содержимое которого будет выведено, если тело ответа на включаемый запрос пустое или если при исполнении запроса произошла ошибка, например:

```
<!--# block name="one" -->&nbsp;  <!--# endblock -->
<!--# include virtual="/remote/body.php?argument=value" stub="one" -->
```

Содержимое замещающего блока обрабатывается в контексте включаемого запроса.

wait

нестандартный параметр, указывающий, нужно ли ждать полного исполнения данного запроса, прежде чем продолжать выполнение SSI, например:

```
<!--# include virtual="/remote/body.php?argument=value" wait="yes" -->
```

set

нестандартный параметр, указывающий, что удачный результат выполнения запроса нужно записать в заданную переменную, например:

```
<!--# include virtual="/remote/body.php?argument=value" set="one" -->
```

Максимальный размер ответа задается директивой *subrequest_output_buffer_size*:

```
location /remote/ {
    subrequest_output_buffer_size 64k;
    # ...
}
```

set

Присваивает значение переменной. Параметры команды:

var

имя переменной.

value

значение переменной. Если в присваиваемом значении есть переменные, то производится подстановка их значений.

Встроенные переменные

`$date_gmt`

текущее время в GMT. Формат задается командой *config* с параметром *timefmt*.

`$date_local`

текущее время в локальной временной зоне. Формат задается командой *config* с параметром *timefmt*.

SSL

Обеспечивает работу по протоколу HTTPS.

Примечание

Для этого модуля нужна библиотека OpenSSL.

Пример конфигурации

Для уменьшения загрузки процессора рекомендуется

- установить число рабочих процессов равным числу процессоров,
- разрешить keep-alive соединения,
- включить разделяемый кэш сессий,
- выключить встроенный кэш сессий
- и, возможно, увеличить время жизни сессии (по умолчанию 5 минут):

```
worker_processes auto;

http {

    # ...

    server {
        listen          443 ssl;
        keepalive_timeout 70;

        ssl_protocols    TLSv1.2 TLSv1.3;
        ssl_ciphers       AES128-SHA:AES256-SHA:RC4-SHA:DES-CBC3-SHA:RC4-MD5;
        ssl_certificate   /usr/local/angie/conf/cert.pem;
        ssl_certificate_key /usr/local/angie/conf/cert.key;
        ssl_session_cache shared:SSL:10m;
        ssl_session_timeout 10m;

        # ...

    }
}
```

Директивы

ssl_buffer_size

Синтаксис	ssl_buffer_size размер;
По умолчанию	ssl_buffer_size 16k;
Контекст	http, server

Задаёт размер буфера, используемого при отправке данных.

По умолчанию размер буфера равен 16k, что соответствует минимальным накладным расходам при передаче больших ответов. С целью минимизации времени получения начала ответа (Time To First Byte) может быть полезно использовать меньшие значения, например:

```
ssl_buffer_size 4k;
```

ssl_certificate

Синтаксис	ssl_certificate <i>файл</i> ;
По умолчанию	—
Контекст	http, server

Указывает файл с сертификатом в формате PEM для данного виртуального сервера. Если вместе с основным сертификатом нужно указать промежуточные, то они должны находиться в этом же файле в следующем порядке: сначала основной сертификат, а затем промежуточные. В этом же файле может находиться секретный ключ в формате PEM.

Эта директива может быть указана несколько раз для загрузки сертификатов разных типов, например RSA и ECDSA:

```
server {
    listen          443 ssl;
    server_name     example.com;

    ssl_certificate  example.com.rsa.crt;
    ssl_certificate_key example.com.rsa.key;

    ssl_certificate  example.com.ecdsa.crt;
    ssl_certificate_key example.com.ecdsa.key;

    # ...
}
```

Возможность задавать отдельные цепочки сертификатов для разных сертификатов есть только в OpenSSL 1.0.2 и выше. Для более старых версий следует указывать только одну цепочку сертификатов.

Примечание

В имени файла можно использовать переменные при использовании OpenSSL 1.0.2 и выше:

```
ssl_certificate      $ssl_server_name.crt;
ssl_certificate_key  $ssl_server_name.key;
```

При использовании переменных сертификат загружается при каждой операции SSL-рукопожатия, что может отрицательно влиять на производительность.

Вместо *файла* можно указать значение `data:$переменная`, при котором сертификат загружается из переменной без использования промежуточных файлов.

Ненадлежащее использование подобного синтаксиса может быть небезопасно, например данные секретного ключа могут попасть в лог ошибок.

Примечание

Нужно иметь в виду, что из-за ограничения протокола HTTPS для максимальной совместимости виртуальные серверы должны слушать на разных IP-адресах.

Если задан режим `ssl_ntls`, директива может принимать два аргумента (части ключа для подписи и шифрования) вместо одного:

```
listen ... ssl;

ssl_ntls on;

# двойной сертификат NTLS
ssl_certificate     sign.crt enc.crt;
ssl_certificate_key sign.key enc.key;

# можно использовать наряду с обычным сертификатом RSA
ssl_certificate     rsa.crt;
ssl_certificate_key rsa.key;
```

ssl_certificate_cache

Синтаксис	<code>ssl_certificate_cache off;</code> <code>ssl_certificate_cache max=N [inactive=time] [valid=time];</code>
Значение по умолчанию	<code>ssl_certificate_cache off;</code>
Контекст	http, server

Определяет кэш для хранения SSL-сертификатов и секретных ключей, заданных через переменные.

Директива поддерживает следующие параметры:

- **max** — устанавливает максимальное количество элементов в кэше. При переполнении кэша удаляются наименее недавно использованные (LRU) элементы.
- **inactive** — определяет время, после которого элемент будет удален, если к нему не было обращений в течение этого времени, даже если он еще находится в пределах своего периода **valid**; каждое обращение сбрасывает этот таймер. Значение по умолчанию — 10 секунд.
- **valid** — определяет время, в течение которого элемент кэша считается действительным и может использоваться повторно. Значение по умолчанию — 60 секунд. По истечении этого времени сертификаты перезагружаются или проходят повторную проверку.
- **off** — отключает кэш.

Пример:

```
ssl_certificate     $ssl_server_name.crt;
ssl_certificate_key $ssl_server_name.key;
ssl_certificate_cache max=1000 inactive=20s valid=1m;
```


ssl_certificate_compression

Синтаксис	<code>ssl_certificate_compression on off;</code>
Значение по умолчанию	<code>ssl_certificate_compression off;</code>
Контекст	<code>http, server</code>

Разрешает сжатие TLS 1.3 сертификатов сервера.

Примечание

Директива поддерживается при использовании OpenSSL версии 3.2 и выше; список поддерживаемых алгоритмов сжатия предоставляется библиотекой.

Примечание

Директива поддерживается при использовании BoringSSL; список поддерживаемых алгоритмов сжатия включает `zlib`.

Если включен `ssl_stapling`, сжатие сертификатов отключается.

ssl_certificate_key

Синтаксис	<code>ssl_certificate_key <i>файл</i>;</code>
По умолчанию	<code>—</code>
Контекст	<code>http, server</code>

Указывает файл с секретным ключом в формате PEM для данного виртуального сервера.

Примечание

В имени файла можно использовать переменные при использовании OpenSSL 1.0.2 и выше.

Вместо файла можно указать значение `engine:имя:id`, которое загружает ключ с указанным `id` из движка OpenSSL с заданным именем.

Вместо файла можно указать значение `store:scheme:id`, которое используется для загрузки ключа с указанным `id` и URI-схемой `scheme`, зарегистрированной в OpenSSL provider, например `pkcs11`.

Вместо файла также можно указать значение `data:$переменная`, при котором секретный ключ загружается из переменной без использования промежуточных файлов. При этом следует учитывать, что ненадлежащее использование подобного синтаксиса может быть небезопасно, например данные секретного ключа могут попасть в лог ошибок.

Если задан режим `ssl_nTLS`, директива может принимать два аргумента (части ключа для подписи и шифрования) вместо одного:

```
listen ... ssl;

ssl_nTLS on;

# двойной сертификат NTLS
ssl_certificate    sign.crt enc.crt;
```

```
ssl_certificate_key  sign.key enc.key;

# можно использовать наряду с обычным сертификатом RSA
ssl_certificate  rsa.crt;
ssl_certificate  rsa.key;
```

ssl_ciphers

Синтаксис	<code>ssl_ciphers</code> <i>шифры</i> ;
По умолчанию	<code>ssl_ciphers</code> HIGH:!aNULL:!MD5;
Контекст	http, server

Описывает разрешенные шифры. Шифры задаются в формате, поддерживаемом библиотекой OpenSSL, например:

```
ssl_ciphers ALL:!aNULL:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP;
```

Список шифров зависит от установленной версии OpenSSL. Полный список можно посмотреть с помощью команды `openssl ciphers`.

Предупреждение

Директива `ssl_ciphers` *не* настраивает шифры для TLS 1.3 при использовании OpenSSL. Для настройки шифров TLS 1.3 в OpenSSL используйте директиву `ssl_conf_command`, добавленную для расширенной конфигурации SSL.

- В LibreSSL шифры TLS 1.3 *можно* настраивать с помощью `ssl_ciphers`.
- В BoringSSL шифры TLS 1.3 настроить невозможно.

ssl_client_certificate

Синтаксис	<code>ssl_client_certificate</code> <i>файл</i> ;
По умолчанию	—
Контекст	http, server

Указывает файл с доверенными сертификатами CA в формате PEM, которые используются для проверки клиентских сертификатов и ответов OCSP, если включен `ssl_stapling`.

Список сертификатов будет отправляться клиентам. Если это нежелательно, можно воспользоваться директивой `ssl_trusted_certificate`.

ssl_conf_command

Синтаксис	<code>ssl_conf_command</code> <i>имя значение</i> ;
По умолчанию	—
Контекст	http, server

Задаёт произвольные *конфигурационные команды* OpenSSL.

Примечание

Директива поддерживается при использовании OpenSSL 1.0.2 и выше. Чтобы настроить шифры TLS 1.3 в OpenSSL, используйте команду `ciphersuites`.

На одном уровне может быть указано несколько директив `ssl_conf_command`:

```
ssl_conf_command Options PrioritizeChaCha;
ssl_conf_command Ciphersuites TLS_CHACHA20_POLY1305_SHA256;
```

Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы `ssl_conf_command`.

Предупреждение

Изменение настроек OpenSSL напрямую может привести к неожиданному поведению.

ssl_crl

Синтаксис	<code>ssl_crl файл;</code>
По умолчанию	—
Контекст	http, server

Указывает файл с отозванными сертификатами (CRL) в формате PEM, используемыми для проверки клиентских сертификатов.

ssl_dhparam

Синтаксис	<code>ssl_dhparam файл;</code>
По умолчанию	—
Контекст	http, server

Указывает файл с параметрами для DHE-шифров.

Предупреждение

По умолчанию параметры не заданы, и соответственно DHE-шифры не будут использоваться.

ssl_early_data

Синтаксис	<code>ssl_early_data on off;</code>
По умолчанию	<code>ssl_early_data off;</code>
Контекст	http, server

Разрешает или запрещает TLS 1.3 `early data`.

Запросы, отправленные внутри `early data`, могут быть подвержены атакам повторного воспроизведения (replay). Для защиты от подобных атак на уровне приложения необходимо использовать переменную `$ssl_early_data`.

```
proxy_set_header Early-Data $ssl_early_data;
```

Примечание

Директива поддерживается при использовании OpenSSL 1.1.1 и выше или BoringSSL.

`ssl_ecdh_curve`

Синтаксис	<code>ssl_ecdh_curve <i>кривая</i>;</code>
По умолчанию	<code>ssl_ecdh_curve auto;</code>
Контекст	<code>http, server</code>

Задаёт кривую для ECDHE-шифров.

Примечание

При использовании OpenSSL 1.0.2 и выше можно указывать несколько кривых, например:

```
ssl_ecdh_curve prime256v1:secp384r1;
```

Специальное значение `auto` соответствует встроенному в библиотеку OpenSSL списку кривых для OpenSSL 1.0.2 и выше, или `prime256v1` для более старых версий.

Примечание

При использовании OpenSSL 1.0.2 и выше директива задаёт список кривых, поддерживаемых сервером. Поэтому для работы ECDSA-сертификатов важно, чтобы список включал кривые, используемые в сертификатах.

`ssl_encrypted_hello_key`

Синтаксис	<code>ssl_encrypted_hello_key <i>файл</i>;</code>
По умолчанию	—
Контекст	<code>http, server</code>

Указывает файл с секретным ключом и конфигурацией расширения TLS Encrypted Client Hello (ECH), которое скрывает имя сервера, запрашиваемое клиентом, от наблюдателей в сети. Переменная `$ssl_encrypted_hello` показывает, использовалось ли расширение при рукопожатии.

Директива может быть указана несколько раз, что позволяет настроить ротацию ключей: для расшифровки применяются все ключи, но клиентам для повторных попыток предлагается конфигурация только первого из них.

```
ssl_encrypted_hello_key current.ech;  
ssl_encrypted_hello_key previous.ech;
```

Файл содержит секретный ключ, за которым следует ECHConfigList; оба в формате PEM. В ветке [OpenSSL с поддержкой ECH](#) такой файл создается командой:

```
$ openssl ech -public_name public.example.com -out example.ech
```

В BoringSSL команда `bssl generate-ech` записывает ключ и ECHConfigList в отдельные двоичные файлы, и напрямую использовать их нельзя. Из них нужно собрать один файл PEM: необработанный ключ обернуть в структуру PKCS#8 и закодировать в base64 в виде блока PRIVATE KEY, а следом поместить ECHConfigList, а не отдельный ECHConfig, закодированный в base64, в виде блока ECHCONFIG. Сборки с BoringSSL поддерживают только ключи X25519. Сам формат не зависит от библиотеки TLS, поэтому файл, созданный приведенной выше командой, подходит и для сборки с BoringSSL.

Angie расшифровывает ClientHello до того, как узнает имя сервера, запрошенное через SNI, поэтому ключи необходимо указывать для сервера по умолчанию.

Примечание

Поддержка ECH обеспечивается библиотекой TLS, поэтому Angie необходимо собрать с библиотекой, которая ее реализует: с веткой [OpenSSL с поддержкой ECH](#), которая подключается так:

```
$ ./configure --with-openssl=../openssl-ech
```

либо с BoringSSL, который подключается иначе. В противном случае директива игнорируется с предупреждением в журнале ошибок.

ssl_keylog_file

Синтаксис	<code>ssl_keylog_file <i>файл</i>;</code>
По умолчанию	—
Контекст	http, server

Задаёт файл для журналирования SSL/TLS-ключей соединений в данном контексте, использующий формат `SSLKEYLOGFILE`, который применяется, например, в Wireshark для расшифровки перехваченного TLS-трафика.

Директиву не следует использовать с LibreSSL.

Предупреждение

Файл содержит конфиденциальный ключевой материал TLS, с помощью которого можно расшифровать перехваченный трафик соответствующих соединений. Используйте директиву только для отладки или анализа протокола, ограничьте доступ к файлу и удалите его, когда он больше не нужен.

ssl_ntls

Синтаксис	<code>ssl_ntls on off;</code>
По умолчанию	<code>ssl_ntls off;</code>
Контекст	http, server

Включает серверную поддержку NTLS при использовании TLS библиотеки [Tongsuo](#)

```
listen ... ssl;
ssl_ntls on;
```

ssl_ocsp

Синтаксис	ssl_ocsp on off leaf;
По умолчанию	ssl_ocsp off;
Контекст	http, server

Включает проверку OCSP для цепочки клиентских сертификатов. Параметр `leaf` включает проверку только клиентского сертификата.

Для работы проверки OCSP необходимо дополнительно установить значение директивы `ssl_verify_client` в `on` или `optional`.

Для преобразования имени хоста OCSP-респондера в адрес необходимо дополнительно задать директиву `resolver`.

Пример:

```
ssl_verify_client on;
ssl_ocsp on;
resolver 127.0.0.53;
```

ssl_ocsp_cache

Синтаксис	ssl_ocsp_cache off [shared:имя:размер];
По умолчанию	ssl_ocsp_cache off;
Контекст	http, server

Задаёт имя и размер кэша, который хранит статус клиентских сертификатов для проверки OCSP-ответов. Кэш разделяется между всеми рабочими процессами. Кэш с одинаковым названием может использоваться в нескольких виртуальных серверах.

Параметр `off` запрещает использование кэша.

ssl_ocsp_responder

Синтаксис	ssl_ocsp_responder uri;
По умолчанию	—
Контекст	http, server

Переопределяет URI OCSP-респондера, указанный в расширении сертификата "Authority Information Access" для проверки клиентских сертификатов.

Поддерживаются только OCSP-респондеры на основе `http://`:

```
ssl_ocsp_responder http://ocsp.example.com/;
```

ssl_password_file

Синтаксис	<code>ssl_password_file <i>файл</i>;</code>
По умолчанию	—
Контекст	http, server

Задаёт файл с паролями от секретных ключей, где каждый пароль указан на отдельной строке. Пароли применяются по очереди в момент загрузки ключа.

Пример:

```
http {
    ssl_password_file /etc/keys/global.pass;
    ...

    server {
        server_name www1.example.com;
        ssl_certificate_key /etc/keys/first.key;
    }

    server {
        server_name www2.example.com;

        # вместо файла можно указать именованный канал
        ssl_password_file /etc/keys/fifo;
        ssl_certificate_key /etc/keys/second.key;
    }
}
```

ssl_prefer_server_ciphers

Синтаксис	<code>ssl_prefer_server_ciphers on off;</code>
По умолчанию	<code>ssl_prefer_server_ciphers off;</code>
Контекст	http, server

При использовании протоколов SSLv3 и TLS устанавливает приоритет серверных шифров над клиентскими.

ssl_protocols

Синтаксис	<code>ssl_protocols [SSLv2] [SSLv3] [TLSv1] [TLSv1.1] [TLSv1.2] [TLSv1.3];</code>
По умолчанию	<code>ssl_protocols TLSv1.2 TLSv1.3;</code>
Контекст	http, server

Разрешает указанные протоколы.

Примечание

Параметры TLSv1.1 и TLSv1.2 работают только при использовании OpenSSL 1.0.1 и выше.

Параметр TLSv1.3 работает только при использовании OpenSSL 1.1.1 и выше.

ssl_reject_handshake

Синтаксис	<code>ssl_reject_handshake on off;</code>
По умолчанию	<code>ssl_reject_handshake off;</code>
Контекст	<code>http, server</code>

Если включено, то операции SSL-рукопожатия в блоке `server` будут отклонены.

Например, в этой конфигурации отклоняются все операции SSL-рукопожатия с именем сервера, отличным от `example.com`:

```
server {
    listen          443 ssl default_server;
    ssl_reject_handshake on;
}

server {
    listen          443 ssl;
    server_name     example.com;
    ssl_certificate example.com.crt;
    ssl_certificate_key example.com.key;
}
```

ssl_session_cache

Синтаксис	<code>ssl_session_cache off none [builtin[:размер]] [shared:название:размер];</code>
По умолчанию	<code>ssl_session_cache none;</code>
Контекст	<code>http, server</code>

Задаёт тип и размеры кэшей для хранения параметров сессий. Тип кэша может быть следующим:

off	жёсткое запрещение использования кэша сессий: Angie явно сообщает клиенту, что сессии не могут использоваться повторно.
none	мягкое запрещение использования кэша сессий: Angie сообщает клиенту, что сессии могут использоваться повторно, но на самом деле не хранит параметры сессии в кэше.
builtin	встроенный в OpenSSL кэш, используется в рамках только одного рабочего процесса. Размер кэша задается в сессиях. Если размер не задан, то он равен 20480 сессиям. Использование встроенного кэша может вести к фрагментации памяти.
shared	кэш, разделяемый между всеми рабочими процессами. Размер кэша задается в байтах, в 1 мегабайт может поместиться около 4000 сессий. У каждого разделяемого кэша должно быть произвольное название. Кэш с одинаковым названием может использоваться в нескольких виртуальных серверах. Также он используется для автоматического создания, хранения и периодического обновления ключей сессионных билетов TLS, если они не указаны явно с помощью директивы <code>ssl_session_ticket_key</code> .

Можно использовать одновременно оба типа кэша, например:

```
ssl_session_cache builtin:1000 shared:SSL:10m;
```


однако использование только разделяемого кэша без встроенного должно быть более эффективным.

ssl_session_ticket_key

Синтаксис	<code>ssl_session_ticket_key <i>файл</i>;</code>
По умолчанию	—
Контекст	http, server

Задаёт файл с секретным ключом, применяемым при шифровании и расшифровке сессионных билетов TLS. Директива необходима, если один и тот же ключ нужно использовать на нескольких серверах. По умолчанию используется случайно сгенерированный ключ.

Если указано несколько ключей, то только первый ключ используется для шифрования сессионных билетов TLS. Это позволяет настроить ротацию ключей, например:

```
ssl_session_ticket_key current.key;
ssl_session_ticket_key previous.key;
```

Файл должен содержать 80 или 48 байт случайных данных и может быть создан следующей командой:

```
$ openssl rand 80 > ticket.key
```

В зависимости от размера файла для шифрования будет использоваться либо AES256 (для 80-байтных ключей), либо AES128 (для 48-байтных ключей).

ssl_session_tickets

Синтаксис	<code>ssl_session_tickets on off;</code>
По умолчанию	<code>ssl_session_tickets on;</code>
Контекст	http, server

Разрешает или запрещает возобновление сессий при помощи сессионных билетов TLS.

ssl_session_timeout

Синтаксис	<code>ssl_session_timeout время;</code>
По умолчанию	<code>ssl_session_timeout 5m;</code>
Контекст	http, server

Задаёт время, в течение которого клиент может повторно использовать параметры сессии.

ssl_stapling

Синтаксис	<code>ssl_stapling on off;</code>
По умолчанию	<code>ssl_stapling off;</code>
Контекст	http, server

Разрешает или запрещает [прикрепление OCSP-ответов](#) сервером. Пример:

```
ssl_stapling on;  
resolver 127.0.0.53;
```

Для работы OCSP-прикрепления должен быть известен сертификат издателя сертификата сервера. Если в заданном директивой `ssl_certificate` файле не содержится промежуточных сертификатов, то сертификат издателя сертификата сервера следует поместить в файл, заданный директивой `ssl_trusted_certificate`.

Предупреждение

Для преобразования имени хоста OCSP-респондера в адрес необходимо дополнительно задать директиву `resolver`.

ssl_stapling_file

Синтаксис	<code>ssl_stapling_file файл;</code>
По умолчанию	—
Контекст	http, server

Если значение задано, то вместо опроса OCSP-респондера, указанного в сертификате сервера, ответ берется из указанного файла.

Ответ должен быть в формате DER и может быть сгенерирован командой `openssl ocsp`.

ssl_stapling_responder

Синтаксис	<code>ssl_stapling_responder uri;</code>
По умолчанию	—
Контекст	http, server

Переопределяет URI OCSP-респондера, указанный в расширении сертификата "Authority Information Access".

Поддерживаются только OCSP-респондеры на основе `http://`:

```
ssl_stapling_responder http://ocsp.example.com/;
```

ssl_stapling_verify

Синтаксис	<code>ssl_stapling_verify on off;</code>
По умолчанию	<code>ssl_stapling_verify off;</code>
Контекст	http, server

Разрешает или запрещает проверку ответов OCSP сервером.

Для работоспособности проверки сертификат издателя сертификата сервера, корневой сертификат и все промежуточные сертификаты должны быть указаны как доверенные с помощью директивы `ssl_trusted_certificate`.

ssl_trusted_certificate

Синтаксис	<code>ssl_trusted_certificate <i>файл</i>;</code>
По умолчанию	—
Контекст	http, server

Задаёт файл с доверенными сертификатами СА в формате PEM, которые используются для проверки клиентских сертификатов и ответов OCSP, если включен `ssl_stapling`.

В отличие от `ssl_client_certificate`, список этих сертификатов не будет отправляться клиентам.

ssl_verify_client

Синтаксис	<code>ssl_verify_client on off optional optional_no_ca;</code>
По умолчанию	<code>ssl_verify_client off;</code>
Контекст	http, server

Разрешает проверку клиентских сертификатов. Результат проверки доступен через переменную `$ssl_client_verify`.

optional	запрашивает клиентский сертификат, и если сертификат был предоставлен, проверяет его
optional_no_ca	запрашивает сертификат клиента, но не требует, чтобы он был подписан доверенным сертификатом СА. Это предназначено для случаев, когда фактическая проверка сертификата осуществляется внешним по отношению к Angie сервисом.

ssl_verify_depth

Синтаксис	<code>ssl_verify_depth <i>число</i>;</code>
По умолчанию	<code>ssl_verify_depth 1;</code>
Контекст	http, server

Устанавливает глубину проверки в цепочке клиентских сертификатов.

Обработка ошибок

Модуль `http_ssl` поддерживает несколько нестандартных кодов ошибок, которые можно использовать для перенаправления с помощью директивы `error_page`:

495	при проверке клиентского сертификата произошла ошибка;
496	клиент не предоставил требуемый сертификат;
497	обычный запрос был послан на порт HTTPS.

Перенаправление делается после того, как запрос полностью разобран и доступны такие переменные, как `$request_uri`, `$uri`, `$args` и другие переменные.

Встроенные переменные

Модуль `http_ssl` поддерживает встроенные переменные:

`$ssl_alpn_protocol`

возвращает протокол, выбранный при помощи ALPN во время SSL-рукопожатия, либо пустую строку.

`$ssl_cipher`

возвращает название используемого шифра для установленного SSL-соединения.

`$ssl_ciphers`

возвращает список шифров, поддерживаемых клиентом. Известные шифры указаны по имени, неизвестные указаны в шестнадцатеричном виде, например:

AES128-SHA:AES256-SHA:0x00ff

Примечание

Переменная полностью поддерживается при использовании OpenSSL версии 1.0.2 и выше. При использовании более старых версий переменная доступна только для новых сессий и может содержать только известные шифры.

`$ssl_client_escaped_cert`

возвращает клиентский сертификат в формате PEM (закодирован в формате urlencode) для установленного SSL-соединения.

`$ssl_client_fingerprint`

возвращает SHA1-отпечаток клиентского сертификата для установленного SSL-соединения.

`$ssl_client_i_dn`

возвращает строку "issuer DN" клиентского сертификата для установленного SSL-соединения согласно RFC 2253.

`$ssl_client_i_dn_legacy`

возвращает строку "issuer DN" клиентского сертификата для установленного SSL-соединения.

`$ssl_client_raw_cert`

возвращает клиентский сертификат для установленного SSL-соединения в формате PEM.

`$ssl_client_s_dn`

возвращает строку "subject DN" клиентского сертификата для установленного SSL-соединения согласно RFC 2253.

`$ssl_client_s_dn_legacy`

возвращает строку "subject DN" клиентского сертификата для установленного SSL-соединения.

`$ssl_client_serial`

возвращает серийный номер клиентского сертификата для установленного SSL-соединения.

`$ssl_client_sigalg`

возвращает алгоритм подписи для сертификата клиента в установленном SSL-соединении.

Примечание

Переменная поддерживается только при использовании OpenSSL версии 3.5 и выше. При использовании более старых версий значением переменной будет пустая строка.

Примечание

Переменная доступна только для новых сессий.

`$ssl_client_v_end`

возвращает дату окончания срока действия клиентского сертификата.

`$ssl_client_v_remain`

возвращает число дней, оставшихся до истечения срока действия клиентского сертификата.

`$ssl_client_v_start`

возвращает дату начала срока действия клиентского сертификата.

`$ssl_client_verify`

возвращает результат проверки клиентского сертификата: `SUCCESS`, `FAILED:reason` и, если сертификат не был предоставлен, `NONE`.

`$ssl_curve`

возвращает согласованную кривую, использованную для обмена ключами во время SSL-рукопожатия. Известные кривые указаны по имени, неизвестные указаны в шестнадцатеричном виде, например:

`prime256v1`

Примечание

Переменная поддерживается при использовании OpenSSL версии 3.0 и выше. При использовании более старых версий значением переменной будет пустая строка.

`$ssl_curves`

возвращает список кривых, поддерживаемых клиентом. Известные кривые указаны по имени, неизвестные указаны в шестнадцатеричном виде, например:

`0x001d:prime256v1:secp521r1:secp384r1`

i Примечание

Переменная поддерживается при использовании OpenSSL версии 1.0.2 и выше. При использовании более старых версий значением переменной будет пустая строка.

Переменная доступна только для новых сессий.

`$ssl_early_data`

возвращает 1, если используется TLS 1.3 early data и операция SSL-рукопожатия не завершена, иначе "".

`$ssl_encrypted_hello`

возвращает 1, если ClientHello был расшифрован с помощью расширения TLS Encrypted Client Hello (ECH), иначе "".

`$ssl_protocol`

возвращает протокол установленного SSL-соединения.

`$ssl_server_cert_type`

принимает значения RSA, DSA, ECDSA, ED448, ED25519, SM2, RSA-PSS или unknown в зависимости от типа сертификата и ключа сервера.

`$ssl_server_name`

возвращает имя сервера, запрошенное через SNI.

`$ssl_session_id`

возвращает идентификатор сессии установленного SSL-соединения.

`$ssl_session_reused`

возвращает r, если сессия была использована повторно, иначе ".".

`$ssl_sigalg`

возвращает алгоритм подписи для сертификата сервера в установленном SSL-соединении.

i Примечание

Переменная поддерживается только при использовании OpenSSL версии 3.5 и выше. При использовании более старых версий значением переменной будет пустая строка.

i Примечание

Переменная доступна только для новых сессий.

`$ssl_sigalgs`

возвращает список **алгоритмов подписи**, предложенных клиентом в процессе SSL-рукопожатия, в виде списка, разделённого двоеточиями.

Примечание

При использовании OpenSSL версии 4.0 и выше алгоритмы возвращаются по именам; для алгоритма без имени используется соответствующий шестнадцатеричный код. SSL-библиотеки, предоставляющие прежний API алгоритмов подписи, возвращают все алгоритмы в виде шестнадцатеричных кодов. Если сборка не предоставляет поддерживаемого API алгоритмов подписи, значение переменной пустое.

Stub Status

Предоставляет доступ к базовой информации о состоянии сервера.

Пример конфигурации

```
location = /basic_status {
    stub_status;
}
```

В данной конфигурации создается простая веб-страница с основной информацией о состоянии, которая может выглядеть следующим образом:

```
Active connections: 291
server accepts handled requests
 16630948 16630948 31070465
Reading: 6 Writing: 179 Waiting: 106
```

Директивы

stub_status

Синтаксис	<code>stub_status;</code>
По умолчанию	—
Контекст	server, location

Информация о состоянии будет доступна из данного location.

Данные

Доступна следующая информация:

Active connections

Текущее число активных клиентских соединений, включая Waiting-соединения.

accepts

Суммарное число принятых клиентских соединений.

handled

Суммарное число обработанных соединений. Обычно значение этого параметра совпадает с **accepts**, если не достигнуто какое-нибудь системное ограничение (например, лимит *worker_connections*).

requests

Суммарное число клиентских запросов.

Reading

Текущее число соединений, в которых Angie в настоящий момент читает заголовок запроса.

Writing

Текущее число соединений, в которых Angie в настоящий момент отвечает клиенту.

Waiting

Текущее число бездействующих клиентских соединений в ожидании запроса.

Встроенные переменные

\$connections_active

то же, что и значение *Active connections*;

\$connections_reading

то же, что и значение *Reading*;

\$connections_waiting

то же, что и значение *Waiting*.

\$connections_writing

то же, что и значение *Writing*;

Sub

Фильтр, изменяющий в ответе одну заданную строку на другую.

Пример конфигурации

```
location / {
    sub_filter '<a href="http://127.0.0.1:8080/' '<a href="https://$host/';
    sub_filter '<img src="http://127.0.0.1:8080/' '<img src="https://$host/';
    sub_filter_once on;
}
```

Директивы

sub_filter

Синтаксис	<code>sub_filter строка замена;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт строку, которую нужно заменить, и строку замены. Заменяемая строка проверяется без учёта регистра. В заменяемой строке и в строке замены можно использовать переменные. На одном уровне конфигурации может быть указано несколько директив `sub_filter`. Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы `sub_filter`.

sub_filter_last_modified

Синтаксис	<code>sub_filter_last_modified on off;</code>
По умолчанию	<code>sub_filter_last_modified off;</code>
Контекст	http, server, location

Позволяет сохранить поле заголовка `Last-Modified` исходного ответа во время замены для лучшего кэширования ответов.

По умолчанию поле заголовка удаляется, так как содержимое ответа изменяется во время обработки.

sub_filter_once

Синтаксис	<code>sub_filter_once on off;</code>
По умолчанию	<code>sub_filter_once on;</code>
Контекст	http, server, location

Определяет, сколько раз нужно искать каждую из заменяемых строк: один раз или многократно.

sub_filter_types

Синтаксис	<code>sub_filter_types mime-<i>mun</i> ...;</code>
По умолчанию	<code>sub_filter_types text/html;</code>
Контекст	http, server, location

Разрешает замену строк в ответах с указанными MIME-типами в дополнение к `text/html`. Специальное значение `"*"` соответствует любому MIME-типу.

Tunnel

Модуль `Tunnel` (`ngx_http_tunnel_module`) обрабатывает запросы HTTP/1.1 `CONNECT` и устанавливает сквозное виртуальное соединение между клиентом и проксируемым сервером, позволяя Angie выступать в роли перенаправляющего прокси.

Доступ к туннелю управляется стандартными директивами разграничения доступа: `auth_basic`, `satisfy` и `auth_delay`.

Пример конфигурации

```
http {
    map $host $allow_host {
        hostnames;

        example.org    1;
        *.example.org  1;
    }

    server {
        listen 8000;

        resolver 127.0.0.53;

        if ($allow_host != 1) {
            return 502;
        }

        tunnel_pass;
    }
}
```

Директивы

tunnel_bind

Синтаксис	<code>tunnel_bind address [transparent] off;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт локальный IP-адрес с необязательным портом, который будет использоваться в исходящих соединениях с проксируемым сервером. Значение параметра может содержать переменные. Специальное значение `off` отменяет действие директивы `tunnel_bind`, унаследованной с предыдущего уровня конфигурации, что позволяет системе автоматически назначать локальный IP-адрес и порт.

Параметр `transparent` позволяет задать нелокальный IP-адрес, который будет использоваться в исходящих соединениях с проксируемым сервером, например реальный IP-адрес клиента:

```
tunnel_bind $remote_addr transparent;
```

tunnel_buffer_size

Синтаксис	<code>tunnel_buffer_size размер;</code>
По умолчанию	<code>tunnel_buffer_size 4k 8k;</code>
Контекст	http, server, location

Задаёт *размер* буфера, используемого для чтения данных от проксируемого сервера. Также задаёт *размер* буфера, используемого для чтения данных от клиента. По умолчанию размер буфера равен размеру страницы памяти. В зависимости от платформы это или 4К, или 8К.

tunnel_connect_timeout

Синтаксис	<code>tunnel_connect_timeout</code> время;
По умолчанию	<code>tunnel_connect_timeout 60s</code> ;
Контекст	http, server, location

Задаёт таймаут для установления соединения с проксируемым сервером. Следует учитывать, что этот таймаут обычно не может превышать 75 секунд.

tunnel_next_upstream

Синтаксис	<code>tunnel_next_upstream error timeout off ...</code> ;
По умолчанию	<code>tunnel_next_upstream error timeout</code> ;
Контекст	http, server, location

Определяет, в каких случаях запрос будет передан следующему серверу:

error	произошла ошибка соединения с сервером или чтения данных от него;
timeout	произошёл таймаут во время установления соединения с сервером, передачи ему запроса или чтения данных от него;
off	запрещает передачу запроса следующему серверу.

Передача запроса следующему серверу возможна только при условии, что клиенту ещё ничего не передавалось. Если ошибка или таймаут происходят в середине передачи ответа, исправить это невозможно.

Директива также определяет, что считается неудачной попыткой работы с сервером. Случаи **error** и **timeout** всегда считаются неудачными попытками, даже если они не указаны в директиве.

Передача запроса следующему серверу может быть ограничена количеством попыток и временем.

tunnel_next_upstream_timeout

Синтаксис	<code>tunnel_next_upstream_timeout</code> время;
По умолчанию	<code>tunnel_next_upstream_timeout 0</code> ;
Контекст	http, server, location

Ограничивает время, в течение которого запрос может быть передан следующему серверу. Значение 0 отключает это ограничение.

tunnel_next_upstream_tries

Синтаксис	<code>tunnel_next_upstream_tries number</code> ;
По умолчанию	<code>tunnel_next_upstream_tries 0</code> ;
Контекст	http, server, location

Ограничивает число возможных попыток передачи запроса следующему серверу. Значение 0 отключает это ограничение.

tunnel_pass

Синтаксис	<code>tunnel_pass [address];</code>
По умолчанию	—
Контекст	server, location, if в location

Включает обработку запросов CONNECT и задаёт адрес проксируемого сервера. По умолчанию *адрес* равен `$host:$request_port` и берётся из запроса клиента. В большинстве случаев для `tunnel_pass` не требуется указывать аргументы.

Адрес может быть задан доменным именем или IP-адресом и портом:

```
tunnel_pass localhost:9000;
```

или в виде пути к UNIX-сокету:

```
tunnel_pass unix:/tmp/backend.socket;
```

Если доменному имени соответствует несколько адресов, все они будут использоваться по круговому принципу. Кроме того, адрес может быть задан как группа серверов.

Значение параметра может содержать переменные. В этом случае, если адрес задан доменным именем, имя ищется среди описанных групп серверов, а если не найдено, определяется с помощью *resolver*.

tunnel_read_timeout

Синтаксис	<code>tunnel_read_timeout время;</code>
По умолчанию	<code>tunnel_read_timeout 60s;</code>
Контекст	http, server, location

Задаёт таймаут между двумя последовательными операциями чтения или записи на соединениях с клиентом или проксируемым сервером. Если в течение этого времени данные не передаются, соединение закрывается.

tunnel_send_lowat

Синтаксис	<code>tunnel_send_lowat размер;</code>
По умолчанию	<code>tunnel_send_lowat 0;</code>
Контекст	http, server, location

Если директиве задано ненулевое значение, Angie будет пытаться уменьшить число операций отправки на исходящих соединениях с проксируемым сервером, используя либо флаг `NOTE_LOWAT` метода `kqueue`, либо параметр сокета `SO_SNDLOWAT` с указанным *размером*.

Эта директива игнорируется на Linux, Solaris и Windows.

tunnel_send_timeout

Синтаксис	<code>tunnel_send_timeout время;</code>
По умолчанию	<code>tunnel_send_timeout 60s;</code>
Контекст	<code>http, server, location</code>

Задаёт таймаут при передаче данных проксируемому серверу. Таймаут устанавливается только между двумя последовательными операциями записи, а не на всю передачу. Если проксируемый сервер ничего не получит в течение этого времени, соединение закрывается.

tunnel_socket_keepalive

Синтаксис	<code>tunnel_socket_keepalive on off;</code>
По умолчанию	<code>tunnel_socket_keepalive off;</code>
Контекст	<code>http, server, location</code>

Настраивает поведение "TCP keepalive" для исходящих соединений с проксируемым сервером. По умолчанию для сокета действуют настройки операционной системы. Если директиве задано значение `on`, для сокета включается параметр сокета `SO_KEEPALIVE`.

Upstream

Предоставляет контекст для описания группы серверов, которые могут использоваться в директивах `proxy_pass`, `fastcgi_pass`, `uwsgi_pass`, `scgi_pass`, `memcached_pass` и `grpc_pass`.

Пример конфигурации

```
upstream backend {
    zone backend 1m;
    server backend1.example.com      weight=5;
    server backend2.example.com:8080;
    server backend3.example.com      service=_example._tcp resolve;
    server unix:/tmp/backend3;

    server backup1.example.com:8080   backup;
    server backup2.example.com:8080   backup;
}

resolver 127.0.0.53 status_zone=resolver;

server {
    location / {
        proxy_pass http://backend;
    }
}
```

Директивы

backup_switch

Синтаксис	<code>backup_switch permanent[=<i>время</i>];</code>
По умолчанию	—
Контекст	upstream

Директива включает возможность начать выбор серверов не с основной группы, а с *активной*, то есть той, где в предыдущий раз был успешно найден сервер. Если найти сервер в активной группе для очередного запроса не удастся, и поиск переходит к резервной группе, то уже эта группа становится активной, и последующие запросы сначала направляются на серверы этой группы.

Если параметр **permanent** определен без значения времени, группа остается активной после выбора, и автоматическая перепроверка групп с меньшим уровнем не происходит. Если *время* задано, то активный статус группы истекает через указанный интервал, и балансировщик снова проверяет группы с меньшим уровнем, возвращаясь к ним, если серверы работают нормально.

Пример:

```
upstream my_backend {
    zone my_backend 1m;
    server primary1.example.com;
    server primary2.example.com;

    server backup1.example.com backup;
    server backup2.example.com backup;

    backup_switch permanent=2m;
}
```

Если балансировщик переключается с основных серверов на резервную группу, все последующие запросы обрабатываются этой резервной группой в течение 2 минут. По истечении 2 минут балансировщик повторно проверяет основные серверы и снова делает их активными, если они работают нормально.

bind_conn

Синтаксис	<code>bind_conn <i>значение</i>;</code>
По умолчанию	—
Контекст	upstream

Позволяет привязать серверное соединение к клиентскому в момент, когда *значение*, заданное строкой из переменных, становится отличным от "" и "0".

Если привязанный сервер не может принять следующий запрос по клиентскому соединению — например, после удаления его адреса из группы в результате resolve или пока сервер недоступен, — привязанное соединение закрывается и будет выбран другой сервер согласно настроенному методу балансировки.

Предупреждение

Директива **bind_conn** должна использоваться после всех директив, задающих тот или иной метод балансировки нагрузки, иначе она не будет работать. Если она используется наряду с директивой **sticky**, то **bind_conn** должна стоять после **sticky**.

Предупреждение

При использовании директивы настройки модуля Proxy должны допускать использование постоянных соединений, например:

```
proxy_http_version 1.1;
proxy_set_header Connection "";
```

Типичный пример использования директивы — проксирование соединений с NTLM-аутентификацией, где требуется обеспечить привязку клиента к серверу в начале согласования:

```
map $http_authorization $ntlm {
    ~*~NTLM      1;
    ~*~Negotiate 1;
}

upstream ntlm_backend {
    zone ntlm_backend 1m;
    server 127.0.0.1:8080;
    bind_conn $ntlm;
}

server {
    # ...
    location / {
        proxy_pass http://ntlm_backend;
        proxy_http_version 1.1;
        proxy_set_header Connection "";
        # ...
    }
}
```

feedback

Синтаксис	feedback <i>переменная</i> [inverse] [factor= <i>число</i>] [account= <i>условная_переменная</i>] [last_byte];
По умолчанию	—
Контекст	upstream

Задаёт в **upstream** механизм балансировки нагрузки по обратной связи. Он динамически корректирует решения при балансировке, умножая вес каждого проксируемого сервера на среднее значение обратной связи, которое меняется с течением времени в зависимости от значения *переменной* и подчиняется необязательному условию.

Могут быть заданы следующие параметры:

переменная	Переменная, из которой берется значение обратной связи. Она должна представлять собой метрику производительности или состояния; предполагается, что сервер передает ее в заголовках или иным образом. Значение оценивается при каждом ответе от сервера и учитывается в скользящем среднем согласно настройкам inverse и factor .
inverse	Если параметр задан, значение обратной связи интерпретируется наоборот: более низкие значения указывают на лучшую производительность.
factor	Коэффициент, по которому значение обратной связи учитывается при расчете среднего. Допустимы целые числа от 0 до 99. По умолчанию — 90. Среднее рассчитывается по формуле экспоненциального сглаживания . Чем больше коэффициент, тем меньше новые значения влияют на среднее; если указать 90, то будет взято 90 % от предыдущего значения и лишь 10 % от нового.
account	Указывает условную переменную, которая контролирует, какие ответы учитываются при расчете. Среднее значение обновляется с учетом значения обратной связи из ответа, только если условная переменная этого ответа не равна "" или "0".

Примечание

По умолчанию ответы в ходе активных проверок не включаются в расчет; комбинация переменной `$upstream_probe` с `account` позволяет включить эти ответы или даже исключить все остальное.

last_byte	Позволяет обрабатывать данные от проксируемого сервера после получения полного ответа, а не только заголовка.
-----------	---

Пример:

```
upstream backend {
    zone backend 1m;

    feedback $feedback_value factor=80 account=$condition_value;

    server backend1.example.com;
    server backend2.example.com;
}

map $upstream_http_custom_score $feedback_value {
    "high"          100;
    "medium"        75;
    "low"           50;
    default         10;
}

map $upstream_probe $condition_value {
    "high_priority" "1";
    "low_priority"  "0";
    default        "1";
}
```

Эта конфигурация категоризирует ответы серверов по уровням обратной связи на основе определенных оценок из полей заголовков ответа, а также добавляет условие на `$upstream_probe`, чтобы учитывать только ответы от активной проверки `high_priority` или ответы на обычные клиентские запросы.

hash

Синтаксис	<code>hash <i>ключ</i> [consistent];</code>
По умолчанию	—
Контекст	upstream

Задаёт метод балансировки нагрузки для группы, при котором соответствие клиента серверу определяется при помощи хэшированного значения ключа. В качестве ключа можно использовать переменные. Следует отметить, что любое добавление или удаление серверов в группе может привести к перераспределению большинства ключей на другие серверы. Метод совместим с библиотекой Perl `Cache::Memcached`.

```
hash $remote_addr;
```

При использовании доменных имен, разрешающихся в несколько IP-адресов (например, с параметром `resolve`), сервер не сортирует полученные адреса, поэтому их порядок может различаться на разных серверах, что влияет на распределение клиентов. Чтобы обеспечить одинаковое распределение, используйте параметр `consistent`.

Если задан параметр `consistent`, то вместо вышеописанного метода будет использоваться метод консистентного хэширования `ketama`. Метод гарантирует, что при добавлении сервера в группу или его удалении на другие серверы будет перераспределено минимальное число ключей. Применение метода для кэширующих серверов обеспечивает больший процент попаданий в кэш. Метод совместим с библиотекой Perl `Cache::Memcached::Fast` при значении параметра `ketama_points`, равном 160.

ip_hash

Синтаксис	<code>ip_hash;</code>
По умолчанию	—
Контекст	upstream

Задаёт для группы метод балансировки нагрузки, при котором запросы распределяются по серверам на основе IP-адресов клиентов. В качестве ключа для хэширования используются первые три октета IPv4-адреса клиента или IPv6-адрес клиента целиком. Метод гарантирует, что запросы одного и того же клиента будут всегда передаваться на один и тот же сервер. Если же этот сервер будет считаться недоступным, то запросы этого клиента будут передаваться на другой сервер. С большой долей вероятности это также будет один и тот же сервер.

Если один из серверов нужно убрать на некоторое время, то для сохранения текущего хэширования IP-адресов клиентов этот сервер нужно пометить параметром `down`:

```
upstream backend {
    zone backend 1m;
    ip_hash;

    server backend1.example.com;
    server backend2.example.com;
    server backend3.example.com down;
    server backend4.example.com;
}
```

keepalive

Синтаксис	<code>keepalive <i>соединения</i>;</code>
По умолчанию	—
Контекст	upstream

Задействует кэш соединений для группы серверов.

Параметр *соединения* устанавливает максимальное число неактивных постоянных соединений с серверами группы, которые будут сохраняться в кэше каждого рабочего процесса. При превышении этого числа наиболее давно не используемые соединения закрываются.

Примечание

Следует особо отметить, что директива *keepalive* не ограничивает общее число соединений с серверами группы, которые рабочие процессы Angie могут открыть. Параметр соединения следует устанавливать достаточно консервативно, чтобы серверы группы по-прежнему могли обрабатывать новые входящие соединения.

Примечание

Закэшированные соединения сопоставляются с серверами группы по адресу. При удалении адреса из группы в результате resolve или команды API DELETE закэшированные для этого адреса соединения не закрываются: они занимают место в кэше до закрытия сервером, истечения *keepalive timeout* или вытеснения и используются повторно, только если этот адрес возвращается в группу. Директивы *proxy_connection_drop*, *fastcgi_connection_drop*, *uwsgi_connection_drop*, *scgi_connection_drop* и *grpc_connection_drop* их не закрывают: они действуют только на активные соединения.

Предупреждение

Директива *keepalive* должна использоваться после всех директив, задающих тот или иной метод балансировки нагрузки, иначе она не будет работать.

Пример конфигурации группы серверов memcached с постоянными соединениями:

```
upstream memcached_backend {
    zone memcached_backend 1m;
    server 127.0.0.1:11211;
    server 10.0.0.2:11211;

    keepalive 32;
}

server {
    #...

    location /memcached/ {
        set $memcached_key $uri;
        memcached_pass memcached_backend;
    }
}
```

```
}
```

Для HTTP директиву *proxy_http_version* следует установить в "1.1", а поле заголовка *Connection* — очистить:

```
upstream http_backend {
    zone http_backend 1m;
    server 127.0.0.1:8080;

    keepalive 16;
}

server {
    #...

    location /http/ {
        proxy_pass http://http_backend;
        proxy_http_version 1.1;
        proxy_set_header Connection "";
    }
}
```

Примечание

Хоть это и не рекомендуется, но также возможно использование постоянных соединений с HTTP/1.0, путем передачи поля заголовка "Connection: Keep-Alive" серверу группы.

Для работы постоянных соединений с FastCGI-серверами потребуется включить директиву *fastcgi_keep_conn*:

```
upstream fastcgi_backend {
    zone fastcgi_backend 1m;
    server 127.0.0.1:9000;

    keepalive 8;
}

server {
    #...

    location /fastcgi/ {
        fastcgi_pass fastcgi_backend;
        fastcgi_keep_conn on;
    }
}
```

Примечание

Протоколы SCGI и uwsgi не определяют семантику постоянных соединений.

keepalive_requests

Синтаксис	<code>keepalive_requests <i>число</i>;</code>
По умолчанию	<code>keepalive_requests 1000;</code>
Контекст	upstream

Задаёт максимальное число запросов, которые можно сделать по одному постоянному соединению. После того как сделано максимальное число запросов, соединение закрывается.

Периодическое закрытие соединений необходимо для освобождения памяти, выделенной под конкретные соединения. Поэтому использование слишком большого максимального числа запросов может приводить к чрезмерному потреблению памяти и не рекомендуется.

keepalive_time

Синтаксис	<code>keepalive_time <i>время</i>;</code>
По умолчанию	<code>keepalive_time 1h;</code>
Контекст	upstream

Ограничивает максимальное время, в течение которого могут обрабатываться запросы в рамках постоянного соединения. По достижении заданного времени соединение закрывается после обработки очередного запроса.

keepalive_timeout

Синтаксис	<code>keepalive_timeout <i>время</i>;</code>
По умолчанию	<code>keepalive_timeout 60s;</code>
Контекст	upstream

Задаёт таймаут, в течение которого неактивное постоянное соединение с сервером группы не будет закрыто.

Примечание

Этот таймаут управляет неактивными соединениями с серверами группы. Его не следует путать с клиентской директивой `keepalive_timeout` (в контексте `http`, `server` или `location`); он также не влияет на время ожидания установления нового соединения с сервером группы, которое задается таймаутом соединения соответствующего протокола, например `proxy_connect_timeout`.

least_bandwidth

Синтаксис	<code>least_bandwidth both upstream downstream [<i>factor=number</i>];</code>
По умолчанию	—
Контекст	upstream

Модуль балансировки нагрузки `least_bandwidth` выбирает проксируемый сервер, который использует наименьший объем пропускной способности, стремясь равномерно распределить сетевой трафик.

фик, направляя новые сессии на сервер с наименьшим потреблением пропускной способности. Соответственно, чем меньше трафика, тем сервер считается менее загруженным и тем больше запросов будет на него отправляться.

upstream	Директива учитывает только пропускную способность, используемую от клиента к серверу .
downstream	Директива учитывает только пропускную способность, используемую от сервера к клиенту .
both	Директива учитывает сумму пропускной способности, используемой вверх (от клиента к серверу) и вниз по потоку (от сервера к клиенту).
factor	Необязательный коэффициент сглаживания для расчета скользящего среднего, в диапазоне от 0 до 99 (по умолчанию 90). Значение задается в процентах: при 90 берется 90 % от предыдущего среднего и 10 % от нового измерения. Более высокий коэффициент означает более медленную адаптацию к изменениям.

Модуль периодически вычисляет среднее использование пропускной способности для каждого проксируемого сервера, используя формулу скользящего среднего для сглаживания колебаний со временем. Когда начинается новая сессия, он выбирает проксируемый сервер с наименьшим средним использованием пропускной способности, скорректированным с учетом веса сервера, если он указан. Если несколько серверов имеют одинаковое минимальное среднее использование пропускной способности, применяется взвешенный метод round-robin.

least_conn

Синтаксис	least_conn;
По умолчанию	—
Контекст	upstream

Задаёт для группы метод балансировки нагрузки, при котором запрос передается серверу с наименьшим числом активных соединений, с учетом весов серверов. Если подходит сразу несколько серверов, они выбираются циклически (в режиме round-robin) с учетом их весов.

least_time

Синтаксис	least_time header last_byte [factor=число] [account=условная_переменная];
По умолчанию	—
Контекст	upstream

Задаёт для группы метод балансировки нагрузки, при котором вероятность передачи запроса активному серверу обратно пропорциональна среднему времени его ответа; чем оно меньше, тем больше запросов будет получать сервер.

header	Директива учитывает среднее время получения заголовков ответа.
last_byte	Директива использует среднее время получения полного ответа.

factor	Выполняет ту же функцию, что и <i>response_time_factor</i> , и переопределяет его, если параметр задан.
account	Указывает условную переменную, которая контролирует, какие ответы учитываются при расчете. Среднее значение обновляется, только если условная переменная ответа не равна "" или "0".

Примечание

По умолчанию активные проверки не включаются в расчет; комбинация переменной *\$upstream_probe* с **account** позволяет включить эти ответы или даже исключить все остальное.

Текущие значения представлены как **header_time** (только заголовки) и **response_time** (ответы целиком) в объекте **health** сервера среди метрик апстрима в API.

queue

<i>Синтаксис</i>	queue <i>число</i> [timeout = <i>время</i>] [priority = <i>\$переменная</i>];
По умолчанию	—
<i>Контекст</i>	upstream

Если для запроса не удастся назначить проксируемый сервер с первой попытки (например, при краткосрочном перебое в работе или всплеске нагрузки с достижением предела *max_conns*), запрос не отклоняется; вместо этого Angie пытается поставить его в очередь на обработку.

Численный параметр директивы задает максимальное количество запросов в очереди *рабочего процесса*. Если очередь целиком заполнена, клиенту отдается ошибка 502 (Bad Gateway). При этом, если задана приоритизация запросов (**priority=\$var**), то новые приоритетные запросы будут вытеснять из очереди старые низкоприоритетные. Таким образом, новые приоритетные запросы будут обрабатываться в первую очередь и не будут теряться в условиях критической перегрузки.

Параметр **priority=\$переменная** позволяет задать приоритизацию запросов в очереди и обрабатывать только запросы с наивысшим приоритетом. Переменная, из которой берется значение приоритета запроса, должна быть целым положительным числом; большее значение означает более высокий приоритет, а отсутствующее или некорректное значение трактуется как 0. Первыми обрабатываются запросы с наибольшим текущим приоритетом; запросы с меньшим приоритетом ждут, пока в очереди есть более приоритетные, а при переполнении очереди отбрасываются первыми. Это позволяет в условиях критической перегрузки сохранить обслуживание наиболее важных клиентов, отсекая второстепенные запросы. В качестве критерия для определения приоритета запроса могут использоваться GeoIP-данные (страна, определенная подсеть) или авторизация пользователя на сервисе и пр.

Пример наивысшего приоритета запросам из определенной страны:

```
map $geoip_country_code $priority_val {
    "RU"    100;
    default 1;
}

upstream backend {
    zone backend 1m;
    server backend.example.com:8080;
    queue 100 priority=$priority_val;
}
```

Пример понижения приоритета запросам от ботов:

```
map $http_user_agent $is_bot {
    default                0;

    # Регистронезависимый (~*) маппинг для распространенных вредоносных ботов и
    ↪ скраперов
    ~*malicious_bot        1;
    ~*scraping_bot          1;
    ~*bingbot               1;
    ~*curl                  1;
    ~*wget                  1;
    ~*libwww-perl           1;

    # ИИ-краулеры (если необходимо)
    ~*GPTBot                1;
    ~*ClaudeBot             1;

    # Пустые User-Agent-ы
    ""                      1;
}

map $is_bot $priority_val {
    default                100;
    0                      100;
    1                      0;
}

upstream backend {
    zone backend 1m;
    server backend.example.com:8080;
    queue 100 priority=$priority_val;
}
```

Примечание

К запросам в очереди также применяется логика директивы `proxy_next_upstream`. В частности, если для запроса был выбран сервер, но передать его туда не удалось, то он может вернуться в очередь.

Если сервер для передачи запроса в очереди не был выбран за *время* `timeout` (по умолчанию — 60 секунд), клиенту отдается ошибка 502 (Bad Gateway). Еще из очереди удаляются запросы от клиентов, преждевременно закрывших соединение; в *API* есть счетчики состояний запросов, проходящих через очередь.

Предупреждение

Директива `queue` должна использоваться после всех директив, задающих тот или иной метод балансировки нагрузки, иначе она не будет работать.

random

Синтаксис	<code>random [two];</code>
По умолчанию	—
Контекст	upstream

Задаёт для группы метод балансировки нагрузки, при котором запрос передаётся случайно выбранному серверу, с учётом весов серверов.

Если указан необязательный параметр `two`, Angie случайным образом выбирает два сервера, из которых выбирает сервер, используя метод `least_conn`, при котором запрос передаётся на сервер с наименьшим количеством активных соединений.

response_time_factor

Синтаксис	<code>response_time_factor число;</code>
По умолчанию	<code>response_time_factor 90;</code>
Контекст	upstream

Задаёт для метода балансировки нагрузки `least_time` коэффициент сглаживания **предыдущего** значения при вычислении среднего времени ответа по формуле **экспоненциально взвешенного скользящего среднего**.

Чем больше указанное *число*, тем меньше новые значения влияют на среднее; если указать `90`, то будет взято 90 % от предыдущего значения и лишь 10 % от нового. Допустимые значения — от 0 до 99 включительно.

Текущие результаты вычислений представлены как `header_time` (только заголовки) и `response_time` (ответы целиком) в объекте `health` сервера среди метрик апстрима в API.

Примечание

При подсчёте учитываются только успешные ответы; что считать неуспешным ответом, определяют директивы `proxy_next_upstream`, `fastcgi_next_upstream`, `uwsgi_next_upstream`, `scgi_next_upstream`, `memcached_next_upstream` и `grpc_next_upstream`. Кроме того, значение `header_time` пересчитывается, только если получены и обработаны все заголовки, а `response_time` — только если получен весь ответ.

server

Синтаксис	<code>server адрес [параметры];</code>
По умолчанию	—
Контекст	upstream

Задаёт адрес и другие параметры сервера. Адрес может быть указан в виде доменного имени или IP-адреса, и необязательного порта, или в виде пути UNIX-сокета, который указывается после префикса `unix:`. Если порт не указан, используется порт 80. Доменное имя, которому соответствует несколько IP-адресов, задаёт сразу несколько серверов.

Могут быть заданы следующие параметры:

<code>weight=число</code>	Задаёт вес сервера. По умолчанию — 1.
<code>max_conns=число</code>	Ограничивает максимальное число одновременных активных соединений к проксируемому серверу. Значение по умолчанию равно 0 и означает, что ограничения нет. Если группа не находится в зоне разделяемой памяти, то ограничение работает отдельно для каждого рабочего процесса.

i Примечание

При включённых неактивных постоянных соединениях, нескольких рабочих процессах и зоне разделяемой памяти суммарное число активных и неактивных соединений с проксируемым сервером может превышать значение `max_conns`.

`max_fails=число` — задаёт число неудачных попыток связи с сервером, которые должны произойти в течение заданного `fail_timeout` времени для того, чтобы сервер считался недоступным; после этого он будет повторно проверен через то же самое время.

Что считается неудачной попыткой, определяется директивами `proxy_next_upstream`, `fastcgi_next_upstream`, `uwsgi_next_upstream`, `scgi_next_upstream`, `memcached_next_upstream` и `grpc_next_upstream`.

При превышении `max_fails` сервер также признаётся неработающим с точки зрения `upstream_probe`; клиентские запросы не будут направляться к нему, пока проверки не признают его работающим.

i Примечание

Если директива `server` в группе разрешается в несколько серверов, её настройка `max_fails` применяется к каждому серверу отдельно.

Если после разрешения всех директив `server` в апстриме остаётся только один сервер, настройка `max_fails` не действует и будет проигнорирована.

<code>max_fails=1</code>	Число попыток по умолчанию;
<code>max_fails=0</code>	Отключает учёт попыток.

`fail_timeout=время` — задаёт период времени, в течение которого должно произойти определённое число неудачных попыток связи с сервером (`max_fails`), чтобы сервер считался недоступным. Затем сервер остаётся недоступным в течение того же самого времени, прежде чем будет проверен повторно.

Значение по умолчанию — 10 секунд.

i Примечание

Если директива `server` в группе разрешается в несколько серверов, её настройка `fail_timeout` применяется к каждому серверу отдельно.

Если после разрешения всех директив `server` в апстриме остаётся только один сервер, настройка `fail_timeout` не действует и будет проигнорирована.

backup	Помечает сервер как запасной (создается отдельная группа резервирования серверов). На этот сервер будут передаваться запросы в случае, если не работает вся основная группа серверов. Можно создать несколько групп резервирования серверов и задать уровень для каждой группы с помощью <code>backup=<уровень_группы></code>. Например, <code>backup=1</code> (или просто <code>backup</code>) — группа резервных серверов первого уровня, (в API отображается как <code>true</code>), <code>backup=2</code> — группа резервных серверов второго уровня, (в API отображается как соответствующее число). Если директива <code>server</code> использует SRV-записи DNS для получения адресов проксируемых серверов, то уровень группы резервирования можно задать с помощью параметра <code>backup=prio</code>. В этом случае абсолютное значение приоритета <code>priority=n</code> из SRV-записи будет использовано в качестве уровня резервной группы: <code>backup=n</code>. Если параметр <code>backup=prio</code> не задан, уровень резервирования будет определяться через относительное значение приоритета <code>priority=n</code> из SRV-записи. В этом случае директива <code>backup=n</code> задает базовый уровень отсчета (например <code>backup=3</code>), а относительные уровни резервирования отсчитываются от этого уровня. Сервер с наименьшим значением <code>priority</code> получит уровень <code>backup=3</code>, следующий за ним — <code>backup=4</code> и так далее. Если уровень отсчета <code>backup=n</code> не задан, отсчет будет идти от нуля: сервер с наименьшим значением <code>priority</code> получит <code>backup=0</code> и станет основным, за ним — <code>backup=1</code>, резервный, и так далее. Если задана директива <code>backup_switch</code>, также применяется ее логика активного резервирования.
down	Помечает сервер как постоянно недоступный.
drain	Помечает сервер как разгружаемый (<code>draining</code>); это значит, что он получает только запросы сессий, привязанных ранее через <code>sticky</code> . В остальном поведение такое же, как в режиме <code>down</code> .

Предупреждение

Параметр `backup` нельзя использовать совместно с методами балансировки нагрузки `hash`, `ip_hash` и `random`.

Параметры `down` и `drain` взаимно исключают.

resolve	Позволяет отслеживать изменения списка IP-адресов, соответствующего доменному имени, и обновлять его без перезагрузки конфигурации. При этом группа должна находиться в зоне разделяемой памяти; также должен быть определен преобразователь имен в адреса. Соединения, закешированные для адреса, не закрываются при удалении этого адреса из группы.
service=имя	Включает преобразование SRV-записей DNS и задает имя сервиса. Для работы параметра необходимо задать параметр resolve у сервера, не указывая порт сервера при имени хоста. Если в имени службы нет точек, формируется имя по стандарту RFC: к имени службы добавляется префикс _, затем через точку добавляется _tcp. Так, имя службы http даст в результате _http._tcp. Angie разрешает SRV-записи, объединяя нормализованное имя службы и имя хоста и получая список серверов для полученной комбинации через DNS, вместе с их приоритетами и весами. • SRV-записи с наивысшим приоритетом (те, которые имеют минимальное значение приоритета) разрешаются как основные серверы, а прочие записи становятся запасными серверами. Если backup установлено с server, SRV-записи с наивысшим приоритетом разрешаются как запасные серверы, а прочие записи игнорируются. • Вес аналогичен параметру weight директивы server. Если вес задан как в самой директиве, так и в SRV-записи, используется вес, установленный в директиве.

В этом примере выполняется поиск записи **_http._tcp.backend.example.com**:

```
server backend.example.com service=http resolve;
```

sid=id	Задаёт ID сервера в группе. Если параметр не задан, то ID задается как шестнадцатеричный MD5-хэш IP-адреса и порта или пути UNIX-сокета.
slow_start=время	Задаёт время восстановления веса сервера, возвращающегося к работе при балансировке нагрузки методом round-robin или <i>least_conn</i>. Если параметр задан и сервер после сбоя снова считается работающим с точки зрения max_fails и <i>upstream_probe</i>, то такой сервер равномерно набирает указанный для него вес в течение заданного времени. Если параметр не задан, то в аналогичной ситуации сервер сразу начинает работу с указанным для него весом.

Примечание

Если в апстриме задан только один **server**, **slow_start** не работает и будет игнорироваться.

spare	Помечает сервер как запасной (spare) в рамках одной группы серверов. Запасные серверы простаивают, пока основные серверы группы справляются с нагрузкой; когда основные серверы становятся недоступны, запасные вводятся в работу (в порядке убывания веса), пока мощность группы не восстановится, а после восстановления основных серверов снова переводятся в режим простоя. Можно добавить несколько запасных серверов. Подробнее см. <i>запасные серверы</i>.
--------------	--

state

Синтаксис	<code>state <i>файл</i>;</code>
По умолчанию	—
Контекст	upstream

Указывает *файл*, где постоянно хранится список серверов апстрима. Для хранения таких файлов специально создается каталог `/var/lib/angie/state/` с соответствующими правами доступа, и в конфигурации остается добавить лишь имя файла:

```
upstream backend {
    zone backend 1m;
    state /var/lib/angie/state/<ИМЯ ФАЙЛА>;
}
```

Список серверов здесь имеет формат, аналогичный **server**. Содержимое файла изменяется при любом изменении серверов в разделе `/config/http/upstreams/` через API конфигурации.

Предупреждение

Чтобы использовать директиву **state** в блоке **upstream**, в нем не должно быть директив **server**, но нужна зона разделяемой памяти (*zone*).

sticky

Синтаксис	<code>sticky cookie <i>name</i> [<i>attr=значение</i>]...;</code> <code>sticky route <i>значение</i>...;</code> <code>sticky learn zone=<i>зона</i> create=\$<i>create_var1</i>... lookup=\$<i>lookup_var1</i>... [<i>header</i>]</code> <code>[<i>norefresh</i>] [<i>timeout=время</i>];</code> <code>sticky learn [<i>zone=зона</i>] lookup=\$<i>lookup_var1</i>... remote_action=<i>uri</i></code> <code>remote_result=\$<i>remote_var</i> [<i>norefresh</i>] [<i>timeout=время</i>];</code>
По умолчанию	—
Контекст	upstream

Настраивает привязку клиентских сессий к проксируемым серверам в режиме, заданном первым параметром; для разгрузки серверов, у которых задана директива **sticky**, можно использовать опцию **drain** в блоке **server**.

Предупреждение

Директива **sticky** должна использоваться после всех директив, задающих тот или иной метод балансировки нагрузки, иначе она не будет работать. Если она используется наряду с директивой *bind_conn*, то *bind_conn* должна стоять после **sticky**.

Режим cookie

Этот режим использует cookie для хранения сессий. Он подходит для ситуаций, когда cookie уже используются для управления сессиями.

Здесь запрос от клиента, пока не привязанного к какому-то серверу, отправляется на сервер, выбираемый согласно настроенному методу балансировки. При этом Angie устанавливает cookie с

уникальным значением, идентифицирующим сервер.

Имя cookie (**name**) задается директивой **sticky**, а значение (**value**) соответствует параметру **sid** директивы **server**. Учтите, что параметр дополнительно хэшируется, если задана директива **sticky_secret**.

Последующие запросы от клиента, содержащие такой cookie, передаются на сервер, заданный значением cookie, то есть сервер с указанным **sid**. Если выбрать сервер не удастся или выбранный сервер не может обработать запрос, то будет выбран другой сервер согласно настроенному методу балансировки.

Директива позволяет назначать атрибуты такого cookie; единственный атрибут, устанавливаемый по умолчанию, — **path=/**. Значения атрибутов задаются строками с переменными. Чтобы удалить атрибут, задайте для него пустое значение: **attr=**. Так, **sticky cookie path=** задает cookie без атрибута **path**.

Здесь Angie создает cookie **srv_id** со сроком действия в 1 час и доменом, заданным переменной:

```
upstream backend {
    zone backend 1m;
    server backend1.example.com:8080;
    server backend2.example.com:8080;

    sticky cookie srv_id domain=$my_domain max-age=3600;
}
```

Режим **route**

Этот режим использует предопределенные идентификаторы маршрутов, которые могут быть встроены в URL, cookie или другие свойства запроса. Он менее гибок, так как зависит от предопределенных значений, но лучше подходит, если такие идентификаторы уже используются.

Здесь проксируемый сервер при получении запроса может назначить клиенту маршрут и вернуть его идентификатор способом, известным им обоим. В качестве идентификатора маршрута должно использоваться значение параметра **sid** директивы **server**. Учтите, что параметр дополнительно хэшируется, если задана директива **sticky_secret**.

Последующие запросы от клиентов, желающих использовать этот маршрут, должны содержать выданный сервером идентификатор, причем так, чтобы он попал в переменные Angie, например в cookie или аргументы запроса.

В параметрах директивы указываются строки, которые могут содержать переменные, чтобы извлечь идентификатор маршрута. Чтобы выбрать сервер, куда передается поступивший запрос, используется первое непустое значение; она затем сравнивается с параметром **sid** директивы **server**. Если выбрать сервер не удастся или выбранный сервер не может обработать запрос, то будет выбран другой сервер согласно настроенному методу балансировки.

Здесь Angie ищет идентификатор маршрута в cookie **route**, затем в аргументе запроса **route**:

```
upstream backend {
    zone backend 1m;
    server backend1.example.com:8080 "sid=server 1";
    server backend2.example.com:8080 "sid=server 2";

    sticky route $cookie_route $arg_route;
}
```

Режим **learn**

В этом режиме для привязки клиента к конкретному проксируемому серверу используется динамически генерируемый ключ; этот режим более гибок, так как назначает серверы на ходу, хранит сесии в зоне разделяемой памяти и поддерживает различные способы передачи идентификаторов сессий.

Здесь сессия создается на основе ответа проксируемого сервера. С параметрами `create` и `lookup` перечисляются переменные, указывающие, как создаются новые и ищутся существующие сессии. Оба параметра можно использовать по несколько раз.

Идентификатором сессии служит значение первой непустой переменной, указанной с `create`; например, это может быть cookie с проксируемого сервера.

Сессии хранятся в зоне разделяемой памяти; ее имя и размер задаются параметром `zone`. Если к сессии не было обращений в течение времени `timeout`, она удаляется. Значение по умолчанию — 1 час.

По умолчанию Angie продлевает срок действия сессии, обновляя метку времени последнего обращения при каждом использовании. Параметр `norefresh` отключает это поведение: сессия истечет строго по таймауту, даже если продолжает использоваться. Такой режим удобен, когда требуется принудительно завершать сессию по истечении времени, например, при интеграции с внешними менеджерами сессий.

Последующие запросы от клиентов, желающих использовать сессию, должны содержать ее идентификатор. Параметр `lookup` ищет идентификатор сессии в пользовательском запросе по заданному для него списку переменных, останавливаясь на первой непустой. Если ничего не найдено — запрос считается новым. Значение найденного идентификатора сопоставляется с сессиями в разделяемой памяти. Если выбрать сервер не удастся или выбранный сервер не может обработать запрос, то будет выбран другой сервер согласно настроенному методу балансировки.

Параметр `header` позволяет создать сессию сразу после получения заголовков ответа от проксируемого сервера. Без него сессия создается только после завершения обработки запроса.

В примере Angie создает сессию, устанавливая в ответе cookie с именем `examplecookie`:

```
upstream backend {
    zone backend 1m;
    server backend1.example.com:8080;
    server backend2.example.com:8080;

    sticky learn
        create=$upstream_cookie_examplecookie
        lookup=$cookie_examplecookie
        zone=client_sessions:1m;
}
```

Режим `learn` с `remote_action`

Параметры `remote_action` и `remote_result` позволяют динамически назначать идентификаторы сессий и управлять ими с использованием удаленного хранилища сессий. При этом зона разделяемой памяти выступает в роли локального кэша, а удаленное хранилище является авторитетным источником. Поэтому параметр `create` несовместим с `remote_action`, так как идентификаторы сессий должны создаваться удаленно.

Если к сессии не было обращений в течение времени `timeout`, она удаляется. Значение по умолчанию — 1 час. Настройка `remote_action` не влияет на таймаут.

По умолчанию Angie продлевает срок действия сессии, обновляя метку времени последнего обращения при каждом ее использовании. Параметр `norefresh` меняет это поведение: сессия истекает строго по таймауту, даже если используется.

Параметр `zone` в конфигурации `sticky` с `remote_action` необязателен. Если он не задан, Angie полностью полагается на удаленное хранилище: не кэширует сессии локально (хотя позволяет кэшировать ответы хранилища через `proxy_cache`) и обращается к удаленному хранилищу всякий раз, когда требуется получить или создать сессию.

Общий принцип работы режима таков: если идентификатор сессии не найден локально или истек таймаут, Angie отправляет синхронный подзапрос в некое удаленное хранилище, заданное параметром `remote_action`.

При поступлении HTTP-запроса Angie выполняет следующие действия:

- Сначала извлекается идентификатор сессии из первой непустой переменной в списке `lookup`. Если все переменные пустые, используется обычный алгоритм балансировки нагрузки без привязки.
- Если задан параметр `zone`, Angie ищет существующую сессию в локальной разделяемой памяти. При обнаружении используется связанный с ней сервер и обработка завершается.
- Если сессия не найдена локально или параметр `zone` не задан, сервер выбирается как обычно, согласно настроенному методу балансировки. Затем Angie отправляет в удаленное хранилище, заданное параметром `remote_action`, синхронный HTTP-подзапрос, который должен содержать в понятном хранилищу виде:
 - идентификатор *сессии* из параметра `lookup` (в конфигурации это переменная `$sticky_sessid`);
 - идентификатор предварительно выбранного *сервера*: значение параметра `sid=` из директивы `server`, если оно задано, либо MD5-хэш имени сервера (в конфигурации это переменная `$sticky_sid`).

Проще всего можно передать эти идентификаторы через HTTP-заголовки с помощью `proxy_set_header`.

- Удаленное хранилище обрабатывает запрос и возвращает HTTP-ответ:

Ответ с кодом 200, 201 или 204 подтверждает выбранный сервер. Если задан параметр `zone`, сессия сохраняется в зоне разделяемой памяти.

Ответ с кодом 409 указывает на конфликт (лишь при наличии `zone`): данный идентификатор сессии уже существует, но связан с другим сервером. Удаленное хранилище должно одновременно с этим вернуть правильный идентификатор сервера в HTTP-заголовке; извлечь его можно через `remote_result`.

При получении от хранилища любого другого HTTP-кода (включая ошибки сети и таймауты) либо несуществующего идентификатора сервера Angie использует изначально выбранный сервер.

Идентификатор сервера извлекается из ответа удаленного хранилища через параметр `remote_result`: в нем можно указывать переменные с префиксом `upstream_http_`, которые создаются Angie автоматически для доступа к заголовкам HTTP-ответов от удаленного хранилища. Например, заголовок `X-Sid: server1` в таком ответе становится доступным в переменной `$upstream_http_x_sid` со значением `server1`.

В следующем примере Angie создает сессию, использует переменную `$cookie_bar` для начального идентификатора сессии, а альтернативные идентификаторы сессий, возвращенные удаленным хранилищем, сохраняет в `$upstream_http_x_sticky_sid`:

```
http {

    upstream u1 {

        server srv1;
        server srv2;

        sticky learn zone=sz:1m
            lookup=$cookie_bar
            remote_action=/remote_session
            remote_result=$upstream_http_x_sticky_sid;

        zone z 1m;
    }

    server {
```

```
listen localhost;

location / {

    proxy_pass http://u1/;
}

location /remote_session {

    internal;
    proxy_set_header X-Sticky-Sessid $sticky_sessid;
    proxy_set_header X-Sticky-Sid $sticky_sid;
    proxy_set_header X-Sticky-Last $msec;
    proxy_pass http://remote;
}
}
```

Ниже показан упрощенный пример конфигурации. Удаленное хранилище возвращает идентификатор сессии в заголовке X-Sid и таким образом подтверждает или переопределяет выбор Angie:

```
http {

    proxy_cache_path c1 keys_zone=s1:1m;

    upstream tc_0 {
        server 10.0.0.1 sid=web-server-01;
        server 10.0.0.2 sid=web-server-02;

        sticky learn
            lookup=$arg_id
            remote_action=@create_session
            remote_result=$upstream_http_x_sid;
    }

    server {
        listen 127.0.0.1:8080;

        location / {
            proxy_pass http://tc_0/;
        }

        # Запрос к удаленному хранилищу сессий
        location @create_session {
            internal;

            proxy_set_header X-Sticky-Sessid $sticky_sessid;
            proxy_set_header X-Sticky-Sid $sticky_sid;
            proxy_set_header X-Sticky-Last $msec;

            proxy_pass http://session_backend;

            proxy_connect_timeout 1s;
            proxy_read_timeout 1s;

            proxy_cache s1;
            proxy_cache_valid 200 1d;
        }
    }
}
```



```
        proxy_cache_key    "$scheme$proxy_host$request_uri$sticky_sessid";
    }
}
}
```

Здесь при следующем ответе от удаленного хранилища:

```
HTTP/1.1 200 OK
...
X-Sid: web-server-01
X-Session-Backend: backend-pool-1
```

Становятся доступными две переменные:

- `$upstream_http_x_sid`, со значением `web-server-01`;
- `$upstream_http_x_session_backend`, со значением `backend-pool-1`.

Так как переменная `$upstream_http_x_sid` указана в параметре `remote_result`, то ее значение будет использовано для выбора сервера с `sid=web-server-01`.

Директива `sticky` учитывает состояние серверов в *upstream*:

- Серверы, помеченные как `down` или временно недоступные из-за сбоев, исключаются из выбора.
- Серверы, которые достигли максимального количества соединений (при использовании `max_conns`), временно пропускаются.
- Серверы с опцией `drain` могут быть выбраны для создания новых сессий в режиме `sticky` при совпадении идентификаторов.
- Если ранее недоступный сервер восстанавливается, `sticky` автоматически возобновляет его использование.

Поведение `sticky` можно дополнительно настроить директивами `sticky_secret` и `sticky_strict`. Если в ходе работы `sticky` выбрать сервер не удастся или он недоступен, запрос будет обработан согласно выбранному методу балансировки нагрузки, если только не включена директива `sticky_strict`. В режиме `sticky_strict on`; запрос отклоняется с ошибкой.

Зоны разделяемой памяти, указываемые в параметре `zone` директивы `sticky`, не могут использоваться совместно различными *upstream*; каждая группа должна использовать свою собственную зону.

sticky_secret

Синтаксис	<code>sticky_secret строка;</code>
По умолчанию	—
Контекст	<code>upstream</code>

Добавляет *строку* как соль в функцию MD5-хэширования для директивы `sticky` в режимах `cookie` и `route`. *Строка* может содержать переменные, например `$remote_addr`:

```
upstream backend {
    zone backend 1m;
    server backend1.example.com:8080;
    server backend2.example.com:8080;

    sticky cookie cookie_name;
```

```
sticky_secret my_secret.$remote_addr;
}
```

Соль добавляется после хэшируемого значения; чтобы независимо проверить механизм хэширования:

```
$ echo -n "<VALUE><SALT>" | md5sum
```

sticky_strict

Синтаксис	<code>sticky_strict on off;</code>
По умолчанию	<code>sticky_strict off;</code>
Контекст	<code>upstream</code>

При включении Angie будет возвращать клиенту ошибку HTTP 502, если желаемый сервер недоступен, вместо использования любого другого доступного сервера, как это происходит, когда в группе нет доступных серверов.

upstream

Синтаксис	<code>upstream <i>имя</i> { ... }</code>
По умолчанию	<code>—</code>
Контекст	<code>http</code>

Описывает группу серверов. Серверы могут слушать на разных портах. Кроме того, можно одновременно использовать серверы, слушающие на TCP- и UNIX-сокетах.

Пример:

```
upstream backend {
    zone backend 1m;
    server backend1.example.com weight=5;
    server 127.0.0.1:8080 max_fails=3 fail_timeout=30s;
    server unix:/tmp/backend3;

    server backup1.example.com backup;
}
```

По умолчанию запросы распределяются по серверам циклически (в режиме round-robin) с учетом весов серверов. В вышеприведенном примере каждые 7 запросов будут распределены так: 5 запросов на backend1.example.com и по одному запросу на второй и третий серверы. Распределение является *равномерным*: запросы к серверу с большим весом распределяются по всему циклу, а не отправляются подряд одной группой.

Если при попытке работы с сервером происходит ошибка, то запрос передается следующему серверу, и так далее до тех пор, пока не будут опробованы все работающие серверы. Если не удастся получить успешный ответ ни от одного из серверов, то клиенту будет возвращен результат работы с последним сервером.

Примечание

По умолчанию сервер, у которого изредка происходят неудачные попытки, не достигающие порога `max_fails`, временно получает меньшую долю запросов и восстанавливает свою полную до-

лю в течение последующих запросов. Это отличается от `slow_start`, который плавно возвращает сервер к работе только после того, как сервер был помечен недоступным и затем восстановился.

zone

Синтаксис	<code>zone имя [размер];</code>
По умолчанию	—
Контекст	upstream

Задаёт имя и размер зоны разделяемой памяти, в которой хранятся конфигурация группы и её рабочее состояние, разделяемые между рабочими процессами. В одной и той же зоне могут быть сразу несколько групп. В этом случае достаточно указать размер только один раз.

Примечание

Содержимое зоны сохраняется при перезагрузке только в том случае, если настроенный **размер** не изменился. Любое изменение — увеличение или уменьшение — приводит к пересозданию зоны с потерей всех данных.

Примечание

Метрики группы собираются, только если настроена эта зона. Без нее группа не отображается в `/status/http/upstreams/<upstream>`, в виджете «HTTP Upstreams» и в выводе Prometheus, и предупреждение при этом не выводится. См. пример конфигурации.

Встроенные переменные

Модуль `http_upstream` поддерживает следующие встроенные переменные:

`$sticky_sessid`

Используется с `remote_action` в *sticky*; хранит начальный идентификатор сессии, взятый из `lookup`.

`$sticky_sid`

Используется с `remote_action` в *sticky*; хранит идентификатор сервера, предварительно связанный с сессией.

`$upstream_addr`

хранит IP-адрес и порт или путь к UNIX-сокету сервера группы. Если при обработке запроса были сделаны обращения к нескольким серверам, то их адреса разделяются запятой, например:

`192.168.1.1:80, 192.168.1.2:80, unix:/tmp/sock`

Если произошло внутреннее перенаправление от одной группы серверов на другую с помощью `X-Accel-Redirect` или *error_page*, то адреса, соответствующие разным группам серверов, разделяются двоеточием, например:

`192.168.1.1:80, 192.168.1.2:80, unix:/tmp/sock : 192.168.10.1:80, 192.168.10.2:80`

Если сервер не может быть выбран, то переменная хранит *имя* группы серверов.

`$upstream_bytes_received`

число байт, полученных от сервера группы. Значения нескольких соединений разделяются запятыми и двоеточиями подобно адресам в переменной `$upstream_addr`.

`$upstream_bytes_sent`

число байт, переданных на сервер группы. Значения нескольких соединений разделяются запятыми и двоеточиями подобно адресам в переменной `$upstream_addr`.

`$upstream_cache_key`

содержит используемый ключ кэширования.

`$upstream_cache_status`

хранит статус доступа к кэшу ответов. Статус может быть одним из MISS, BYPASS, EXPIRED, STALE, UPDATING, REVALIDATED или HIT:

- MISS: Ответ не найден в кэше, и запрос передан на сервер.
- BYPASS: Кэш обойден, и запрос напрямую передан на сервер.
- EXPIRED: Ответ в кэше устарел, и на сервер передан новый запрос для обновления контента.
- STALE: Ответ в кэше устарел, но по-прежнему передается клиентам, пока через какое-то время не произойдет обновление контента с сервера.
- UPDATING: Ответ в кэше устарел, но по-прежнему передается клиентам, пока уже идущее обновление контента с сервера не завершится.
- REVALIDATED: Ответ в кэше устарел, но был успешно перепроверен и не нуждается в обновлении с сервера.
- HIT: Ответ был взят из кэша.

Если запрос пошел в обход кэша без обращения к нему, переменная не устанавливается.

`$upstream_connect_time`

хранит время, затраченное на установление соединения с сервером группы; время хранится в секундах с точностью до миллисекунд. В случае SSL включает в себя время, потраченное на рукопожатие. Времена нескольких соединений разделяются запятыми и двоеточиями подобно адресам в переменной `$upstream_addr`.

`$upstream_cookie_<имя>`

cookie с указанным именем, переданный сервером группы в поле **Set-Cookie** заголовка ответа. Необходимо иметь в виду, что cookie запоминаются только из ответа последнего сервера.

`$upstream_current_addr`

адрес проксируемого сервера, выбранного для текущей попытки соединения; в отличие от `$upstream_addr`, где накапливается список адресов всех серверов, к которым были обращения при повторных попытках, эта переменная всегда содержит только один адрес и становится доступна сразу после выбора сервера, в том числе во время вычисления переменных в директиве `proxy_bind`.

`$upstream_header_time`

хранит время, затраченное на получение заголовка ответа от сервера группы; время хранится в секундах с точностью до миллисекунд. Времена нескольких ответов разделяются запятыми и двоеточиями подобно адресам в переменной `$upstream_addr`.

`$upstream_http_<имя>`

хранят поля заголовка ответа сервера. Например, поле заголовка ответа **Server** доступно в переменной `$upstream_http_server`. Правила преобразования имен полей заголовка ответа в имена переменных такие же, как для переменных с префиксом `$http_`. Необходимо иметь в виду, что поля заголовка запоминаются только из ответа последнего сервера.

`$upstream_queue_time`

хранит время, проведенное запросом в очереди до очередного выбора сервера и выраженное в секундах с точностью до миллисекунд. Времена нескольких попыток разделяются запятыми и двоеточиями подобно адресам в переменной `$upstream_addr`.

`$upstream_request_method`

метод запроса, используемый при обращении к upstream. Он может отличаться от метода запроса клиента, когда кэширование преобразует HEAD в GET или задана директива `proxy_method`.

`$upstream_response_length`

хранит длину ответа, полученного от сервера группы; длина хранится в байтах. Длины нескольких ответов разделяются запятыми и двоеточиями подобно адресам в переменной `$upstream_addr`.

`$upstream_response_time`

хранит время, затраченное на получение ответа от сервера группы; время хранится в секундах с точностью до миллисекунд. Времена нескольких ответов разделяются запятыми и двоеточиями подобно адресам в переменной `$upstream_addr`.

`$upstream_status`

хранит статус, который Angie записывает для каждой попытки обращения к серверу группы. Для DoH-запросов усеченный ответ по UDP записывается со статусом 426, а принятый DNS-ответ — со статусом 200. Статусы нескольких попыток разделяются запятыми и двоеточиями подобно адресам в переменной `$upstream_addr`.

`$upstream_sticky_status`

Статус привязанных запросов.

""	Запрос отправлен в группу серверов, где привязка не включена.
NEW	Запрос не содержит информации о привязке к серверу.
HIT	Запрос с привязкой отправлен на желаемый сервер.
MISS	Запрос с привязкой отправлен на сервер, выбранный по алгоритму балансировки.

Статусы из нескольких соединений разделяются запятыми и двоеточиями подобно адресам в переменной `$upstream_addr`.

`$upstream_trailer_<имя>`

хранит поля из конца ответа, полученного от сервера группы.

`$upstream_transport`

хранит транспортный протокол (TCP или UDP), использованный для каждой попытки соединения с сервером группы. Значения нескольких попыток разделяются запятыми и двоеточиями подобно адресам в переменной `$upstream_addr`.

Upstream Probe

Реализует активные проверки работоспособности (health probes) для Upstream.

Пример конфигурации

```
server {
    listen ...;

    location /backend {
        ...
        proxy_pass http://backend;

        upstream_probe backend_probe
            uri=/probe
            port=10004
            interval=5s
            test=$good
            essential
            fails=3
            passes=3
            max_body=10m
            mode=idle;
    }
}
```

Директивы

`upstream_probe`

<i>Синтаксис</i>	<code>upstream_probe</code> <i>имя</i> [<code>uri=адрес</code>] [<code>port=число</code>] [<code>interval=время</code>] [<code>method=метод</code>] [<code>test=условие</code>] [<code>ping</code>] [<code>ping_timeout=время</code>] [<code>essential</code>] [<code>persistent</code>] [<code>fails=число</code>] [<code>passes=число</code>] [<code>max_body=размер</code>] [<code>mode=always</code> <code>idle</code> <code>onfail</code>];
По умолчанию	—
<i>Контекст</i>	<code>location</code>

Задаёт активную проверку работоспособности серверов тех *upstream*, которые указаны в директивах `proxy_pass`, `uwsgi_pass` и т. д. в том же контексте `location`, где находится директива `upstream_probe`. При этом Angie ADC регулярно выполняет запросы согласно указанным параметрам к каждому серверу в составе апстрима.

Сервер проходит проверку, если запрос к нему успешно выполняется с учетом всех параметров самой директивы `upstream_probe` и всех параметров, влияющих на использование апстримов тем контекстом `location`, где она задана. Это касается в том числе директив `proxy_next_upstream`, `uwsgi_next_upstream` и пр., а также `proxy_set_header` и т. д.

Чтобы использовать проверки, в апстриме необходима зона разделяемой памяти (zone). Для одного апстрима можно определить несколько проверок.

Могут быть заданы следующие параметры:

имя	Обязательное имя проверки.
uri	URI запроса, который добавляется к аргументу <i>proxy_pass</i> , <i>uwsgi_pass</i> и т. д. По умолчанию — <i>/</i> .
port	Альтернативный порт для запроса.
interval	Интервал между проверками. По умолчанию — 5s.
method	HTTP-метод запроса проверки. По умолчанию — GET.
test	Проверяемое при запросе условие; задается строкой с переменными. Если результат подстановки переменных — "" или "0", проверка не пройдена.
ping	Использовать проверку методом ICMP Echo вместо HTTP-запроса. При выборе этого режима Angie ADC отправляет эхо-запрос (ICMP Echo Request) на IP-адрес сервера и ожидает ответ (Echo Reply). Проверка осуществляется по IP-адресу, а номер порта из <i>upstream</i> игнорируется.
 Примечание Параметры <i>uri</i>, <i>port</i>, <i>method</i> и <i>test</i> несовместимы с <i>ping</i> и игнорируются. На Linux рабочие процессы должны быть запущены от имени группы, разрешенной в <i>/proc/sys/net/ipv4/ping_group_range</i>. На FreeBSD необходимы права суперпользователя или наличие соответствующих <i>capability</i> для использования raw-сокетов. При использовании ICMP-проверок переменные, связанные с HTTP-запросом, всегда пусты, так как запрос фактически не выполняется; результат определяется только фактом получения ICMP Echo Reply.	
ping_timeout	Время ожидания ICMP-ответа; по умолчанию — 1s. Используется только вместе с <i>ping</i> .
essential	Если параметр задан, то изначально состояние сервера подлежит уточнению и клиентские запросы не передаются ему, пока проверка не будет пройдена.
persistent	Установка этого параметра требует сначала включить <i>essential</i> ; серверы с <i>persistent</i> , работавшие до <i>перезагрузки конфигурации</i> , начинают получать запросы без необходимости сначала пройти эту проверку.
fails	Число последовательных неуспешных запросов, при котором проверка считает сервер неработающим. По умолчанию — 1.
passes	Число последовательных успешных запросов, при котором проверка считает сервер работающим. По умолчанию — 1.
max_body	Максимальный объем памяти для тела ответа. По умолчанию — 256k.
mode	Режим проверки в зависимости от работоспособности серверов: • <i>always</i> — серверы проверяются независимо от состояния; • <i>idle</i> — проверяются неработающие серверы, а также серверы, где с последнего клиентского запроса прошло время <i>interval</i>. • <i>onfail</i> — проверяются серверы только в неработающем состоянии. По умолчанию — <i>always</i> .

Пример:

```
upstream backend {
    zone backend 1m;

    server backend1.example.com;
    server backend2.example.com;
}

map $upstream_status $good {
    200      "1";
}
```

```
server {
    listen ...;

    location /backend {
        ...
        proxy_pass http://backend;

        upstream_probe backend_probe
            uri=/probe
            port=10004
            interval=5s
            test=$good
            essential
            persistent
            fails=3
            passes=3
            max_body=10m
            mode=idle;
    }
}
```

Детали работы:

- Изначально сервер не получает клиентские запросы, пока не пройдет *все* заданные для него проверки с параметром **essential** (пропуская помеченные как **persistent**, если конфигурация перезагружена и до этого сервер считался работающим). Если таких проверок нет, сервер считается работающим.
- Сервер считается неработающим и не получает клиентские запросы, если *какая-либо* заданная для него проверка достигает своего порога **fails** или сам сервер достигает порога **max_fails**.
- Чтобы неработающий сервер снова мог считаться работающим, *все* заданные для него проверки должны достичь своего порога **passes**; после этого учитывается порог **max_fails**.

Встроенные переменные

Модуль `http_upstream_probe` поддерживает следующие встроенные переменные:

`$upstream_probe`

Имя активной сейчас проверки `upstream_probe`.

`$upstream_probe_body`

Тело ответа от сервера, полученного при проверке `upstream_probe`. Его размер ограничен параметром `max_body`.

UserID

Выдает cookie для идентификации клиентов. Для записи в лог полученных и выданных cookie можно использовать встроенные переменные `$uid_got` и `$uid_set`. Модуль совместим с модулем `mod_uid` для Apache.

Пример конфигурации

```
userid      on;
userid_name uid;
userid_domain example.com;
userid_path /;
```



```
userid_expires 365d;
userid_p3p      'policyref="/w3c/p3p.xml", CP="CUR ADM OUR NOR STA NID" ';
```

Директивы

userid

Синтаксис	<code>userid on v1 log off;</code>
По умолчанию	<code>userid off;</code>
Контекст	http, server, location

Разрешает или запрещает выдачу cookie и запись приходящих cookie в лог:

on	разрешает выдачу cookie версии 2 и запись приходящих cookie в лог;
v1	разрешает выдачу cookie версии 1 и запись приходящих cookie в лог;
log	запрещает выдачу cookie, но разрешает запись приходящих cookie в лог;
off	запрещает выдачу cookie и запись приходящих cookie в лог.

userid_domain

Синтаксис	<code>userid_domain <i>имя</i> none;</code>
По умолчанию	<code>userid_domain none;</code>
Контекст	http, server, location

Задаёт домен, для которого устанавливается cookie. Параметр **none** запрещает выдавать домен для cookie.

userid_expires

Синтаксис	<code>userid_expires время max off;</code>
По умолчанию	<code>userid_expires off;</code>
Контекст	http, server, location

Задаёт время, в течение которого браузер должен хранить cookie. Параметр **max** устанавливает срок хранения cookie до 31 декабря 2037 года 23:55:55 GMT. Указание параметра **off** позволяет ограничить время действия cookie сессией браузера.

userid_flags

Синтаксис	<code>userid_flags off <i>флаг</i> ...;</code>
По умолчанию	<code>userid_flags off;</code>
Контекст	http, server, location

Если параметр не **off**, задаёт один или несколько дополнительных флагов для cookie: *secure*, *httponly*, *samesite=strict*, *samesite=lax*, *samesite=none*.

userid_mark

Синтаксис	<code>userid_mark буква цифра = off;</code>
По умолчанию	<code>userid_mark off;</code>
Контекст	http, server, location

Если параметр не `off`, включает механизм маркировки cookie и задает символ, используемый в качестве метки. Этот механизм позволяет добавить или изменить `userid_p3p` и/или время хранения cookie, но при этом оставить неизменным идентификатор клиента. Меткой может быть любая буква английского алфавита (с учетом регистра), цифра или знак "=".

Если метка задана, то она сравнивается с первым дополняющим символом в base64 представлении идентификатора клиента, передаваемом в cookie. Если они не совпадают, то cookie перепосылается с заданной меткой, временем хранения и заголовком РЗР.

userid_name

Синтаксис	<code>userid_name имя;</code>
По умолчанию	<code>userid_name uid;</code>
Контекст	http, server, location

Задаёт имя cookie.

userid_p3p

Синтаксис	<code>userid_p3p строка none;</code>
По умолчанию	<code>userid_p3p none;</code>
Контекст	http, server, location

Задаёт значение для поля заголовка РЗР, которое будет выдаваться вместе с cookie. Если задано специальное значение `none`, то в ответе не будет заголовка РЗР.

userid_path

Синтаксис	<code>userid_path путь;</code>
По умолчанию	<code>userid_path /;</code>
Контекст	http, server, location

Задаёт путь, для которого устанавливается cookie.

userid_service

Синтаксис	<code>userid_service номер;</code>
По умолчанию	<code>userid_service <IP-адрес сервера>;</code>
Контекст	http, server, location

Если идентификаторы выдаются несколькими серверами (сервисами), то каждому сервису следует назначить свой собственный номер, для обеспечения уникальности выдаваемых идентификаторов клиентов. По умолчанию для cookie первой версии используется ноль. Для cookie второй версии по умолчанию используется число, составленное из последних четырех октетов IP-адреса сервера.

Встроенные переменные

`$uid_got`

Имя cookie и полученный идентификатор клиента.

`$uid_reset`

Если значением является непустая строка не равная 0, то клиентские идентификаторы перевыдаются. Специальное значение `log` дополнительно приводит к выдаче сообщений о перевыданных идентификаторах в *error_log*.

`$uid_set`

Имя cookie и выданный идентификатор клиента.

uWSGI

Позволяет передавать запросы uWSGI-серверу.

Пример конфигурации

```
location / {
    include uwsgi_params;
    uwsgi_pass localhost:9000;
}
```

Директивы

uwsgi_bind

Синтаксис	<code>uwsgi_bind адрес [transparent] off;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт локальный IP-адрес с необязательным портом, который будет использоваться в исходящих соединениях с uwsgi-сервером. В значении параметра можно использовать переменные. Специальное значение `off` отменяет действие унаследованной с предыдущего уровня конфигурации директивы *uwsgi_bind*, позволяя системе самостоятельно выбирать локальный IP-адрес и порт.

По умолчанию это значение вычисляется один раз при инициализации запроса, до выбора uwsgi-сервера. В Angie PRO вычисление происходит после каждого выбора uwsgi-сервера (в том числе при повторных попытках), поэтому адрес привязки может зависеть от выбранного сервера; например, с помощью переменной `$upstream_current_addr`. Если вычисленное значение пусто, привязка не выполняется.

Параметр `transparent` позволяет задать нелокальный IP-адрес, который будет использоваться в исходящих соединениях с uwsgi-сервером, например, реальный IP-адрес клиента:

```
uwsgi_bind $remote_addr transparent;
```

Для работы параметра обычно требуется запустить рабочие процессы Angie с привилегиями суперпользователя. В Linux это не требуется, так как если указан параметр *transparent*, то рабочие процессы наследуют *capability CAP_NET_RAW* из главного процесса.

i Примечание

Необходимо настроить таблицу маршрутизации ядра для перехвата сетевого трафика с uwsgi-сервера.

uwsgi_buffer_size

Синтаксис	<code>uwsgi_buffer_size размер;</code>
По умолчанию	<code>uwsgi_buffer_size 4k 8k;</code>
Контекст	<code>http, server, location</code>

Задаёт размер буфера, в который будет читаться первая часть ответа, получаемого от uwsgi-сервера. В этой части ответа обычно находится небольшой заголовок ответа. По умолчанию размер одного буфера равен размеру страницы памяти. В зависимости от платформы это или 4К, или 8К, однако его можно сделать меньше.

uwsgi_buffering

Синтаксис	<code>uwsgi_buffering on off;</code>
По умолчанию	<code>uwsgi_buffering on;</code>
Контекст	<code>http, server, location</code>

Разрешает или запрещает использовать буферизацию ответов uwsgi-сервера.

on	Angie отправляет ответ клиенту параллельно с чтением ответа от uwsgi-сервера; клиент не ждет ответа целиком. Ответ читается в буферы, заданные директивами <code>uwsgi_buffer_size</code> и <code>uwsgi_buffers</code> . Буфер передается на отправку, как только заполнится целиком, причем суммарный размер буферов, занятых отправкой, ограничен значением <code>uwsgi_busy_buffers_size</code> . Заполненный частично буфер ждет, пока не заполнится или пока не закончится ответ, поэтому данные обычно попадают к клиенту целыми буферами. Часть ответа записывается на диск во временный файл, когда все буферы заполнены, а клиент читает медленнее, чем отдает uwsgi-сервер. Ответы, которые кэшируются или сохраняются на диске, всегда проходят через временный файл. Запись во временные файлы контролируется директивами <code>uwsgi_max_temp_file_size</code> и <code>uwsgi_temp_file_write_size</code> . При буферизации uwsgi-сервер освобождается сразу, как только отдаст ответ целиком: соединение с ним не удерживается все то время, пока ответ читает медленный клиент.
off	Каждая порция данных, прочитанная от uwsgi-сервера, сразу же передается клиенту, без какого-либо минимального размера: прочитанные 10 байт из буфера 4К отправляются немедленно. Angie использует один буфер, заданный директивой <code>uwsgi_buffer_size</code> , поэтому за один раз он читает не больше этого размера. Временный файл не используется, поэтому ответы не кэшируются и не сохраняются на диске, а <code>uwsgi_limit_rate</code> и <code>limit_rate</code> не действуют. Режим <code>off</code> подходит для ответов, которые должны доходить до клиента по мере их формирования: Server-Sent Events, длинные опросы (long polling), потоковая передача.

Буферизация может быть также включена или выключена путем передачи значения "yes" или "no" в поле `X-Accel-Buffering` заголовка ответа. Эту возможность можно запретить с помощью директивы `uwsgi_ignore_headers`.

uwsgi_buffers

Синтаксис	<code>uwsgi_buffers</code> <i>число</i> размер;
По умолчанию	<code>uwsgi_buffers 8 4k 8k;</code>
Контекст	http, server, location

Задаёт число и размер буферов для одного соединения, в которые будет читаться ответ, получаемый от uwsgi-сервера.

По умолчанию размер одного буфера равен размеру страницы. В зависимости от платформы это или 4К, или 8К.

uwsgi_busy_buffers_size

Синтаксис	<code>uwsgi_busy_buffers_size</code> размер;
По умолчанию	<code>uwsgi_busy_buffers_size 8k 16k;</code>
Контекст	http, server, location

При включенной буферизации ответов uwsgi-сервера ограничивает суммарный размер буферов, которые могут быть заняты для отправки ответа клиенту, пока ответ еще не прочитан целиком. Оставшиеся буферы тем временем могут использоваться для чтения ответа и, при необходимости, буферизации части ответа во временный файл.

По умолчанию размер ограничен величиной двух буферов, заданных директивами `uwsgi_buffer_size` и `uwsgi_buffers`.

`uwsgi_cache`

Синтаксис	<code>uwsgi_cache зона off;</code>
По умолчанию	<code>uwsgi_cache off;</code>
Контекст	<code>http, server, location</code>

Задаёт зону разделяемой памяти, используемой для кэширования. Одна и та же зона может использоваться в нескольких местах. В значении параметра можно использовать переменные.

<code>off</code>	запрещает кэширование, унаследованное с предыдущего уровня конфигурации.
------------------	--

Примечание

При использовании `uwsgi_cache` обычно необходимо также задать директиву `uwsgi_cache_valid`, чтобы явно указать время действия кэшированных ответов. Если она не задана, Angie не использует значения по умолчанию, а определяет время хранения ответа в кэше на основе HTTP-заголовков ответа от сервера в следующем порядке приоритета:

1. Заголовок `X-Accel-Expires` (наивысший приоритет).
2. Заголовок `Cache-Control` с параметрами `max-age` или `s-maxage`.
3. Заголовок `Expires`.

Если ни один заголовок не содержит допустимого значения или их нет вовсе, ответ не будет кэшироваться, так как определить срок его действия нельзя.

`uwsgi_cache_background_update`

Синтаксис	<code>uwsgi_cache_background_update on off;</code>
По умолчанию	<code>uwsgi_cache_background_update off;</code>
Контекст	<code>http, server, location</code>

Позволяет запустить фоновый подзапрос для обновления просроченного элемента кэша, в то время как клиенту возвращается устаревший кэшированный ответ.

Предупреждение

Использование устаревшего кэшированного ответа в момент его обновления должно быть разрешено.

uwsgi_cache_bypass

Синтаксис	<code>uwsgi_cache_bypass ...;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт условия, при которых ответ не будет браться из кэша. Если значение хотя бы одного из строковых параметров непустое и не равно "0", то ответ не берётся из кэша:

```
uwsgi_cache_bypass $cookie_nocache $arg_nocache$arg_comment;
uwsgi_cache_bypass $http_pragma $http_authorization;
```

В значении параметров можно использовать переменные.

Можно использовать совместно с директивой `uwsgi_no_cache`.

uwsgi_cache_key

Синтаксис	<code>uwsgi_cache_key строка;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт ключ для кэширования. В значении ключа можно использовать переменные. Например,

```
uwsgi_cache_key localhost:9000$request_uri;
```

uwsgi_cache_lock

Синтаксис	<code>uwsgi_cache_lock on off;</code>
По умолчанию	<code>uwsgi_cache_lock off;</code>
Контекст	http, server, location

Если включено, одновременно только одному запросу будет позволено заполнить новый элемент кэша, идентифицируемый согласно директиве `uwsgi_cache_key`, передав запрос на uwsgi-сервер. Остальные запросы этого же элемента будут либо ожидать появления ответа в кэше, либо освобождения блокировки этого элемента, в течение времени, заданного директивой `uwsgi_cache_lock_timeout`.

uwsgi_cache_lock_age

Синтаксис	<code>uwsgi_cache_lock_age время;</code>
По умолчанию	<code>uwsgi_cache_lock_age 5s;</code>
Контекст	http, server, location

Если последний запрос, переданный на uwsgi-сервер для заполнения нового элемента кэша, не завершился за указанное время, на uwsgi-сервер может быть передан ещё один запрос.

uwsgi_cache_lock_timeout

Синтаксис	uwsgi_cache_lock_timeout время;
По умолчанию	uwsgi_cache_lock_timeout 5s;
Контекст	http, server, location

Задаёт таймаут для `uwsgi_cache_lock`. По истечении указанного времени запрос будет передан на uwsgi-сервер, однако ответ не будет кэширован.

uwsgi_cache_max_range_offset

Синтаксис	uwsgi_cache_max_range_offset <i>число</i> ;
По умолчанию	—
Контекст	http, server, location

Задаёт смещение в байтах для запросов с указанием диапазона запрашиваемых байт (byte-range requests). Если диапазон находится за указанным смещением, range-запрос будет передан на uwsgi-сервер и ответ не будет кэширован.

uwsgi_cache_methods

Синтаксис	uwsgi_cache_methods GET HEAD POST ...;
По умолчанию	uwsgi_cache_methods GET HEAD;
Контекст	http, server, location

Если метод запроса клиента указан в этой директиве, то ответ будет кэширован. Методы "GET" и "HEAD" всегда добавляются в список, но тем не менее рекомендуется перечислять их явно. См. также директиву `uwsgi_no_cache`.

uwsgi_cache_min_uses

Синтаксис	uwsgi_cache_min_uses <i>число</i> ;
По умолчанию	uwsgi_cache_min_uses 1;
Контекст	http, server, location

Задаёт число запросов, после которого ответ будет кэширован.

Предупреждение

Метаданные кэша хранятся в разделяемой памяти. Ручное удаление файлов кэша не сбрасывает счетчики и может привести к непредсказуемому поведению. Для полного сброса остановите сервер, удалите директорию кэша и запустите снова.

Примечание

Сторонние модули очистки кэша (например, Cache Purge) удаляют только файлы, но не сбрасывают счетчик `uwsgi_cache_min_uses`. Директива предназначена для защиты кэша от загрязнения редкими запросами, и сброс счетчика при очистке может негативно повлиять на производительность.

uwsgi_cache_path

Синтаксис	<code>uwsgi_cache_path <i>путь</i> [levels=<i>уровни</i>] [use_temp_path=on off] keys_zone=<i>имя:размер</i>[:file=<i>файл</i>] [inactive=<i>время</i>] [max_size=<i>размер</i>] [min_free=<i>размер</i>] [manager_files=<i>число</i>] [manager_sleep=<i>время</i>] [manager_threshold=<i>время</i>] [loader_files=<i>число</i>] [loader_sleep=<i>время</i>] [loader_threshold=<i>время</i>];</code>
По умолчанию	—
Контекст	http

Задаёт путь и другие параметры кэша. Данные кэша хранятся в файлах. Именем файла в кэше является результат функции MD5 от ключа кэширования.

levels	задаёт уровни иерархии кэша: можно задать от 1 до 3 уровней, на каждом уровне допускаются значения 1 или 2.
---------------	---

Например, при использовании

```
uwsgi_cache_path /data/angie/cache levels=1:2 keys_zone=one:10m;
```

имена файлов в кэше будут такого вида:

```
/data/angie/cache/c/29/b7f54b2df7773722d382f4809d65029c
```

Кэшируемый ответ сначала записывается во временный файл, а потом этот файл переименовывается. Временные файлы и кэш могут располагаться на разных файловых системах. Однако нужно учитывать, что в этом случае вместо дешевой операции переименовывания в пределах одной файловой системы файл копируется с одной файловой системы на другую. Поэтому лучше, если кэш будет находиться на той же файловой системе, что и каталог с временными файлами.

<code>use_temp_path=on</code>	определяет каталог, который будет использоваться для временных файлов
<code> off</code>	
<code>on</code>	Если параметр не задан или установлен в значение <code>on</code> , будет использоваться каталог, задаваемый директивой <code>uwsgi_temp_path</code> для данного <code>location</code>
<code>off</code>	временные файлы будут располагаться непосредственно в каталоге кэша
<code>keys_zone</code>	Задаёт имя и размер зоны разделяемой памяти для хранения всех активных ключей и информации о данных. Метаданные кэша хранятся в разделяемой памяти. Размер зоны задается в байтах и должен составлять не менее двух системных страниц памяти. Также допустимы суффиксы <code>k</code> и <code>m</code>; суффикс <code>g</code>, в отличие от <code>max_size</code>, здесь не поддерживается. Зоны размером в 1 мегабайт достаточно для хранения около 8000 ключей. Когда с <code>keys_zone</code> задан необязательный <i>файл</i>, Angie сбрасывает содержимое этой зоны на диск при завершении работы основного процесса и пытается восстановить ее по тому же адресу памяти при следующем запуске или после обновления бинарных файлов, чтобы обеспечить более надежную сохранность данных и сократить время загрузки кэша. Если восстановление области невозможно из-за изменения ее размера, несовместимости версий бинарных файлов или по другим причинам, Angie запишет в журнал предупреждение (<code>failed to restore zone at address</code>) и не будет использовать механизм восстановления зоны. Вместо этого несовместимый файл будет переименован в <code>.old</code>; вы можете либо удалить его, либо восстановить его имя и вернуть Angie к конфигурации и версии, в которых этот файл был изначально создан.
 Предупреждение Убедитесь, что путь к <i>файлу</i> указан правильно и имеет необходимые права доступа для использования Angie, а также защищен от несанкционированного доступа; относительные пути основаны на префиксе.	
<code>inactive</code>	Если к данным кэша не обращаются в течение времени, заданного этим параметром, они удаляются независимо от их свежести; каждое обращение сбрасывает этот таймер. Это отличается от вытеснения по размеру, выполняемого менеджером кэша (см. ниже): удаление по неактивности происходит, даже когда кэш не достиг <code>max_size</code>, и не зависит от сроков кэширования, заданных директивой <code>uwsgi_cache_valid</code>. По умолчанию <code>inactive</code> равен 10 минутам.

Специальный процесс **менеджера кэша** следит за максимальным размером кэша, а также за минимальным объемом свободного места на файловой системе с кэшем, и удаляет наименее востребованные данные при превышении максимального размера кэша или недостаточном объеме свободного места. Удаление данных происходит итерациями.

max_size	максимальное пороговое значение размера кэша Значение задается в байтах; допустимы суффиксы k , m и g . По умолчанию размер кэша не ограничен.
min_free	минимальное пороговое значение объема свободного места на файловой системе с кэшем Задается в тех же единицах, что и max_size . По умолчанию 0 — отслеживание свободного места отключено. На платформах, где статистика файловой системы недоступна, параметр игнорируется, а Angie записывает в журнал предупреждение min_free is not supported on this platform, ignored .
manager_files	максимальное количество удаляемых элементов кэша за одну итерацию По умолчанию: 100
manager_threshold	ограничивает время работы одной итерации По умолчанию: 200 миллисекунд
manager_sleep	время, в течение которого выдерживается пауза между итерациями По умолчанию: 50 миллисекунд

Через минуту после старта Angie активируется специальный процесс **загрузчика кэша**, который загружает в зону кэша информацию о ранее кэшированных данных, хранящихся на файловой системе. Загрузка также происходит итерациями.

loader_files	максимальное количество элементов кэша к загрузке в одну итерацию По умолчанию: 100
loader_threshold	ограничивает время работы одной итерации По умолчанию: 200 миллисекунд
loader_sleep	время, в течение которого выдерживается пауза между итерациями По умолчанию: 50 миллисекунд

uwsgi_cache_revalidate

Синтаксис	<code>uwsgi_cache_revalidate on off;</code>
По умолчанию	<code>uwsgi_cache_revalidate off;</code>
Контекст	<code>http, server, location</code>

Разрешает ревалидацию просроченных элементов кэша при помощи условных запросов с полями заголовка `If-Modified-Since` и `If-None-Match`.

uwsgi_cache_use_stale

Синтаксис	<code>uwsgi_cache_use_stale error timeout invalid_header updating http_500 http_503 http_403 http_404 http_429 non_idempotent off ...;</code>
По умолчанию	<code>uwsgi_cache_use_stale off;</code>
Контекст	<code>http, server, location</code>

Определяет, в каких случаях можно использовать устаревший кэшированный ответ. Параметры директивы совпадают с параметрами директивы `uwsgi_next_upstream`.

error	Позволяет использовать устаревший кэшированный ответ при невозможности выбора uwsgi-сервера для обработки запроса.
updating	Дополнительный параметр, разрешает использовать устаревший кэшированный ответ, если на данный момент он уже обновляется. Это позволяет минимизировать число обращений к uwsgi-серверам при обновлении кэшированных данных.
non_idempotent	Позволяет использовать устаревший кэшированный ответ, если запрос с неидемпотентным методом уже был отправлен uwsgi-серверу, а при дальнейшей работе с этим сервером произошла любая ошибка; из неидемпотентных методов кэшировать можно только POST, и он должен быть разрешен директивой <code>uwsgi_cache_methods</code> .

Использование устаревшего кэшированного ответа может также быть разрешено непосредственно в заголовке ответа на определенное количество секунд после того, как ответ устарел:

- Расширение `stale-while-revalidate` поля заголовка `Cache-Control` разрешает использовать устаревший кэшированный ответ, если на данный момент он уже обновляется.
- Расширение `stale-if-error` поля заголовка `Cache-Control` разрешает использовать устаревший кэшированный ответ в случае ошибки.

Примечание

Такой способ менее приоритетен, чем задание параметров директивы.

Чтобы минимизировать число обращений к uwsgi-серверам при заполнении нового элемента кэша, можно воспользоваться директивой `uwsgi_cache_lock`.

uwsgi_cache_valid

Синтаксис	<code>uwsgi_cache_valid [код ...] время;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт время кэширования для разных кодов ответа. Например, директивы

```
uwsgi_cache_valid 200 302 10m;
uwsgi_cache_valid 404 1m;
```

задают время кэширования 10 минут для ответов с кодами 200 и 302 и 1 минуту для ответов с кодом 404.

Если указано только время кэширования,

```
uwsgi_cache_valid 5m;
```

то кэшируются только ответы 200, 301 и 302.

Кроме того, можно кэшировать любые ответы с помощью параметра `any`:

```
uwsgi_cache_valid 200 302 10m;
uwsgi_cache_valid 301 1h;
uwsgi_cache_valid any 1m;
```

Примечание

Параметры кэширования могут также быть заданы непосредственно в заголовке ответа. Такой способ приоритетнее, чем задание времени кэширования с помощью директивы.

- Поле заголовка **X-Accel-Expires** задает время кэширования ответа в секундах. Значение *0* запрещает кэшировать ответ. Если значение начинается с префикса *@*, оно задает абсолютное время в секундах с начала эпохи, до которого ответ может быть кэширован.
- Если в заголовке нет поля **X-Accel-Expires**, параметры кэширования определяются по полям заголовка **Expires** или **Cache-Control**.
- Ответ, в заголовке которого есть поле **Set-Cookie**, не будет кэшироваться.
- Ответ, в заголовке которого есть поле **Vary** со специальным значением *"*"*, не будет кэшироваться. Ответ, в заголовке которого есть поле **Vary** с другим значением, будет кэширован с учетом соответствующих полей заголовка запроса.

Обработка одного или более из этих полей заголовка может быть отключена при помощи директивы *uwsgi_ignore_headers*.

uwsgi_connect_timeout

Синтаксис	<code>uwsgi_connect_timeout время;</code>
По умолчанию	<code>uwsgi_connect_timeout 60s;</code>
Контекст	<code>http, server, location</code>

Задаёт таймаут для установления соединения с uwsgi-сервером. Необходимо иметь в виду, что этот таймаут обычно не может превышать 75 секунд.

uwsgi_connection_drop

Синтаксис	<code>uwsgi_connection_drop время on off;</code>
По умолчанию	<code>uwsgi_connection_drop off;</code>
Контекст	<code>http, server, location</code>

Настраивает завершение всех активных соединений с проксируемым сервером, если он был удален из группы или помечен как постоянно недоступный в результате процесса *resolver* или команды **API DELETE**. Неактивные соединения в кэше не затрагиваются.

Соединение завершается, когда обрабатывается следующее событие чтения или записи для клиента или проксируемого сервера.

Установка *времени* включает таймаут до завершения соединения; при выборе значения *on* соединения завершаются немедленно.

uwsgi_force_ranges

Синтаксис	<code>uwsgi_force_ranges on off;</code>
По умолчанию	<code>uwsgi_force_ranges off;</code>
Контекст	<code>http, server, location</code>

Включает поддержку диапазонов запрашиваемых байт (byte-range) для кэшированных и некешированных ответов uwsgi-сервера вне зависимости от наличия поля **Accept-Ranges** в заголовках этих ответов.

uwsgi_hide_header

Синтаксис	<code>uwsgi_hide_header поле;</code>
По умолчанию	—
Контекст	http, server, location

По умолчанию Angie не передает клиенту поля заголовка **Date**, **Server**, **X-Pad** и **X-Accel-...** из ответа uwsgi-сервера. Директива `uwsgi_hide_header` задает дополнительные поля, которые не будут передаваться. Если же передачу полей нужно разрешить, можно воспользоваться директивой `uwsgi_pass_header`.

uwsgi_ignore_client_abort

Синтаксис	<code>uwsgi_ignore_client_abort on off;</code>
По умолчанию	<code>uwsgi_ignore_client_abort off;</code>
Контекст	http, server, location

Определяет, закрывать ли соединение с uwsgi-сервером в случае, если клиент закрыл соединение, не дождавшись ответа.

uwsgi_ignore_headers

Синтаксис	<code>uwsgi_ignore_headers поле ...;</code>
По умолчанию	—
Контекст	http, server, location

Запрещает обработку некоторых полей заголовка из ответа uwsgi-сервера. В директиве можно указать поля **X-Accel-Redirect**, **X-Accel-Expires**, **X-Accel-Limit-Rate**, **X-Accel-Buffering**, **X-Accel-Charset**, **Expires**, **Cache-Control**, **Set-Cookie** и **Vary**.

Если не запрещено, обработка этих полей заголовка заключается в следующем:

- **X-Accel-Expires**, **Expires**, **Cache-Control**, **Set-Cookie** и **Vary** задают параметры кэширования ответа;
- **X-Accel-Redirect** производит внутреннее перенаправление на указанный URI;
- **X-Accel-Limit-Rate** задает ограничение скорости передачи ответа клиенту;
- **X-Accel-Buffering** включает или выключает буферизацию ответа;
- **X-Accel-Charset** задает желаемую кодировку ответа.

uwsgi_intercept_errors

Синтаксис	<code>uwsgi_intercept_errors on off;</code>
По умолчанию	<code>uwsgi_intercept_errors off;</code>
Контекст	http, server, location

Определяет, передавать ли клиенту ответы uwsgi-сервера с кодом больше либо равным 300, или же перехватывать их и перенаправлять на обработку Angie с помощью директивы `error_page`.

uwsgi_interface

Синтаксис	<code>uwsgi_interface имя;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт сетевой интерфейс, который будет использоваться в исходящих соединениях с uwsgi-сервером; сокет привязывается к нему параметром сокета `SO_BINDTODEVICE`.

uwsgi_limit_rate

Синтаксис	<code>uwsgi_limit_rate скорость;</code>
По умолчанию	<code>uwsgi_limit_rate 0;</code>
Контекст	http, server, location

Ограничивает скорость чтения ответа от uwsgi-сервера. *Скорость* задается в байтах в секунду; можно использовать переменные.

0	отключает ограничение скорости
---	--------------------------------

Примечание

Ограничение устанавливается на запрос, поэтому, если Angie одновременно откроет два соединения к uwsgi-серверу, суммарная скорость будет вдвое выше заданного ограничения. Ограничение работает только в случае, если включена буферизация ответов uwsgi-сервера.

uwsgi_max_temp_file_size

Синтаксис	<code>uwsgi_max_temp_file_size размер;</code>
По умолчанию	<code>uwsgi_max_temp_file_size 1024m;</code>
Контекст	http, server, location

Если включена буферизация ответов uwsgi-сервера, и ответ не влезает целиком в буферы, заданные директивами `uwsgi_buffer_size` и `uwsgi_buffers`, часть ответа может быть записана во временный файл. Эта директива задает максимальный размер временного файла. Размер данных, сбрасываемых во временный файл за один раз, задается директивой `uwsgi_temp_file_write_size`.

0	отключает возможность буферизации ответов во временные файлы
---	--

Примечание

Данное ограничение не распространяется на ответы, которые будут кэшированы или сохранены на диске.

uwsgi_modifier1

Синтаксис	<code>uwsgi_modifier1 число;</code>
По умолчанию	<code>uwsgi_modifier1 0;</code>
Контекст	<code>http, server, location</code>

Задаёт значение поля `modifier1` в заголовке пакета `uwsgi`.

uwsgi_modifier2

Синтаксис	<code>uwsgi_modifier2 число;</code>
По умолчанию	<code>uwsgi_modifier2 0;</code>
Контекст	<code>http, server, location</code>

Задаёт значение поля `modifier2` в заголовке пакета `uwsgi`.

uwsgi_next_upstream

Синтаксис	<code>uwsgi_next_upstream error timeout invalid_header http_500 http_503 http_403 http_404 http_429 non_idempotent off ...;</code>
По умолчанию	<code>uwsgi_next_upstream error timeout;</code>
Контекст	<code>http, server, location</code>

Определяет, в каких случаях запрос будет передан следующему в группе `upstream` серверу:

error	произошла ошибка соединения с сервером, передачи ему запроса или чтения заголовка ответа сервера;
timeout	произошел таймаут во время соединения с сервером, передачи ему запроса или чтения заголовка ответа сервера;
invalid_header	сервер вернул пустой или неверный ответ;
http_500	сервер вернул ответ с кодом 500;
http_503	сервер вернул ответ с кодом 503;
http_403	сервер вернул ответ с кодом 403;
http_404	сервер вернул ответ с кодом 404;
http_429	сервер вернул ответ с кодом 429;
non_idempotent	обычно запросы с неидемпотентным методом (<i>POST</i> , <i>LOCK</i> , <i>PATCH</i>) не передаются на другой сервер, если запрос серверу группы уже был отправлен; включение параметра явно разрешает повторять подобные запросы;
off	запрещает передачу запроса следующему серверу.

Примечание

Необходимо понимать, что передача запроса следующему серверу возможна только при условии, что клиенту еще ничего не передавалось. То есть, если ошибка или таймаут возникли в середине передачи ответа клиенту, то действие директивы на такой запрос не распространяется.

Директива также определяет, что считается неудачной попыткой работы с сервером.

error timeout invalid_header	всегда считаются неудачными попытками, даже если они не указаны в директиве
http_500 http_503 http_429	считаются неудачными попытками, только если они указаны в директиве
http_403 http_404	никогда не считаются неудачными попытками

Передача запроса следующему серверу может быть ограничена по количеству попыток и по времени.

uwsgi_next_upstream_timeout

Синтаксис	<code>uwsgi_next_upstream_timeout время;</code>
По умолчанию	<code>uwsgi_next_upstream_timeout 0;</code>
Контекст	http, server, location

Ограничивает время, в течение которого возможна передача запроса следующему серверу.

0	отключает это ограничение
---	---------------------------

uwsgi_next_upstream_tries

Синтаксис	<code>uwsgi_next_upstream_tries число;</code>
По умолчанию	<code>uwsgi_next_upstream_tries 0;</code>
Контекст	http, server, location

Ограничивает число допустимых попыток для передачи запроса следующему серверу.

0	отключает это ограничение
---	---------------------------

uwsgi_no_cache

Синтаксис	<code>uwsgi_no_cache строка ...;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт условия, при которых ответ не будет сохраняться в кэш. Если значение хотя бы одного из строковых параметров непустое и не равно "0", то ответ не будет сохранен:

```
uwsgi_no_cache $cookie_nocache $arg_nocache$arg_comment;  
uwsgi_no_cache $http_pragma $http_authorization;
```

В значении параметров можно использовать переменные.

Можно использовать совместно с директивой `uwsgi_cache_bypass`.

uwsgi_param

Синтаксис	<code>uwsgi_param параметр значение [if_not_empty];</code>
Значение по умолчанию	—
Контекст	http, server, location

Устанавливает параметр, который будет передан серверу uwsgi. Значение может содержать переменные. Эти директивы наследуются с предыдущего уровня конфигурации только в том случае, если на текущем уровне директивы `uwsgi_param` отсутствуют.

Стандартные переменные окружения CGI следует передавать в виде заголовков uwsgi, как показано в файле `uwsgi_params`, поставляемом вместе с дистрибутивом:

```
location / {
    include uwsgi_params;
    # ...
}
```

В стандартном файле `uwsgi_params` параметр `REQUEST_METHOD` задается как `$upstream_request_method`.

Если директива указана с параметром `if_not_empty`, такой параметр будет передан серверу только в случае, если его значение не является пустым:

```
uwsgi_param HTTPS $https if_not_empty;
```

uwsgi_pass

Синтаксис	<code>uwsgi_pass [протокол://] адрес;</code>
По умолчанию	—
Контекст	location, if в location

Задаёт протокол и адрес uwsgi-сервера. В качестве протокола можно указать `uwsgi` или `suwsgi` (secured uwsgi, uwsgi через SSL). Адрес может быть указан в виде доменного имени или IP-адреса, и порта:

```
uwsgi_pass localhost:9000;
uwsgi_pass uwsgi://localhost:9000;
uwsgi_pass suwsgi://[2001:db8::1]:9090;
```

или в виде пути UNIX-сокета, который указывается после слова `unix` и заключается в двоеточия:

```
uwsgi_pass unix:/tmp/uwsgi.socket;
```

Если доменному имени соответствует несколько адресов, то все они будут использоваться по очереди (round-robin). Кроме того, в качестве адреса можно указать группу серверов.

В значении параметра можно использовать переменные. В этом случае, если адрес указан в виде доменного имени, имя ищется среди описанных групп серверов и если не найдено, то определяется с помощью `resolver`'а.

Примечание

Если `uwsgi_pass` стоит в `location` с косой чертой в конце префикса (например, `location /name/`), и при этом в директиве `auto_redirect` указано `default`, запросы без косой черты в конце будут перенаправляться (`/name -> /name/`).

`uwsgi_pass_header`

Синтаксис	<code>uwsgi_pass_header поле ...;</code>
По умолчанию	—
Контекст	<code>http, server, location</code>

Разрешает передавать от uwsgi-сервера клиенту запрещенные для передачи поля заголовка.

`uwsgi_pass_request_body`

Синтаксис	<code>uwsgi_pass_request_body on off;</code>
По умолчанию	<code>uwsgi_pass_request_body on;</code>
Контекст	<code>http, server, location</code>

Позволяет запретить передачу исходного тела запроса на uwsgi-сервер. См. также директиву `uwsgi_pass_request_headers`.

`uwsgi_pass_request_headers`

Синтаксис	<code>uwsgi_pass_request_headers on off;</code>
По умолчанию	<code>uwsgi_pass_request_headers on;</code>
Контекст	<code>http, server, location</code>

Позволяет запретить передачу полей заголовка исходного запроса на uwsgi-сервер. См. также директиву `uwsgi_pass_request_body`.

`uwsgi_read_timeout`

Синтаксис	<code>uwsgi_read_timeout время;</code>
По умолчанию	<code>uwsgi_read_timeout 60s;</code>
Контекст	<code>http, server, location</code>

Задаёт таймаут при чтении ответа uwsgi-сервера. Таймаут устанавливается не на всю передачу ответа, а только между двумя операциями чтения. Если по истечении этого времени uwsgi-сервер ничего не передаст, соединение закрывается.

`uwsgi_request_buffering`

Синтаксис	<code>uwsgi_request_buffering on off;</code>
По умолчанию	<code>uwsgi_request_buffering on;</code>
Контекст	<code>http, server, location</code>

Разрешает или запрещает использовать буферизацию тела запроса клиента.

on	тело запроса полностью читается от клиента перед отправкой запроса на uwsgi-сервер.
off	тело запроса отправляется на uwsgi-сервер сразу же по мере его поступления. В этом случае запрос не может быть передан следующему серверу, если Angie уже начал отправку тела запроса.

Если для отправки тела исходного запроса используется HTTP/1.1 и передача данных частями (chunked transfer encoding), то тело запроса буферизуется независимо от значения директивы.

uwsgi_send_timeout

Синтаксис	<code>uwsgi_send_timeout время;</code>
По умолчанию	<code>uwsgi_send_timeout 60s;</code>
Контекст	<code>http, server, location</code>

Задаёт таймаут при передаче запроса uwsgi-серверу. Таймаут устанавливается не на всю передачу запроса, а только между двумя операциями записи. Если по истечении этого времени uwsgi-сервер не примет новых данных, соединение закрывается.

uwsgi_socket_keepalive

Синтаксис	<code>uwsgi_socket_keepalive on off;</code>
По умолчанию	<code>uwsgi_socket_keepalive off;</code>
Контекст	<code>http, server, location</code>

Конфигурирует поведение "TCP keepalive" для исходящих соединений к uwsgi-серверу.

" "	По умолчанию для сокета действуют настройки операционной системы.
on	для сокета включается параметр <code>SO_KEEPALIVE</code>

uwsgi_ssl_certificate

Синтаксис	<code>uwsgi_ssl_certificate <i>файл</i>;</code>
По умолчанию	—
Контекст	<code>http, server, location</code>

Задаёт файл с сертификатом в формате PEM для аутентификации на uwsgi-сервере. В имени файла можно использовать переменные.

uwsgi_ssl_certificate_cache

Синтаксис	<code>uwsgi_ssl_certificate_cache off;</code> <code>uwsgi_ssl_certificate_cache max=<i>N</i> [inactive=<i>time</i>] [valid=<i>time</i>];</code>
Значение по умолчанию	<code>uwsgi_ssl_certificate_cache off;</code>
Контекст	<code>http, server, location</code>

Определяет кэш для хранения SSL-сертификатов и секретных ключей, заданных через переменные.

Директива поддерживает следующие параметры:

- **max** — устанавливает максимальное количество элементов в кэше. При переполнении кэша удаляются наименее недавно использованные (LRU) элементы.
- **inactive** — определяет время, после которого элемент будет удален, если к нему не было обращений в течение этого времени, даже если он еще находится в пределах своего периода **valid**; каждое обращение сбрасывает этот таймер. Значение по умолчанию — 10 секунд.
- **valid** — определяет время, в течение которого элемент кэша считается действительным и может использоваться повторно. Значение по умолчанию — 60 секунд. По истечении этого времени сертификаты перезагружаются или проходят повторную проверку.
- **off** — отключает кэш.

Пример:

```
uwsgi_ssl_certificate      $uwsgi_ssl_server_name.crt;
uwsgi_ssl_certificate_key  $uwsgi_ssl_server_name.key;
uwsgi_ssl_certificate_cache max=1000 inactive=20s valid=1m;
```

uwsgi_ssl_certificate_key

Синтаксис	<code>uwsgi_ssl_certificate_key <i>файл</i>;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт файл с секретным ключом в формате PEM для аутентификации на uwsgi-сервере.

Вместо файла можно указать значение "`engine:имя:id`", которое загружает ключ с указанным *id* из OpenSSL engine с заданным именем.

Вместо файла можно указать значение "`store:scheme:id`", которое используется для загрузки ключа с указанным *id* и URI-схемой *scheme*, зарегистрированной в OpenSSL provider, например `pkcs11`.

В имени файла можно использовать переменные.

uwsgi_ssl_ciphers

Синтаксис	<code>uwsgi_ssl_ciphers <i>шифры</i>;</code>
По умолчанию	<code>uwsgi_ssl_ciphers DEFAULT;</code>
Контекст	http, server, location

Описывает разрешенные шифры для запросов к uwsgi-серверу. Шифры задаются в формате, поддерживаемом библиотекой OpenSSL.

Список шифров зависит от установленной версии OpenSSL. Полный список можно посмотреть с помощью команды `openssl ciphers`.

Предупреждение

Директива `uwsgi_ssl_ciphers` не настраивает шифры для TLS 1.3 при использовании OpenSSL. Для настройки шифров TLS 1.3 в OpenSSL используйте директиву `uwsgi_ssl_conf_command`, добавленную для расширенной конфигурации SSL.

- В LibreSSL шифры TLS 1.3 *можно* настраивать с помощью `uwsgi_ssl_ciphers`.
- В BoringSSL шифры TLS 1.3 настроить невозможно.

uwsgi_ssl_conf_command

Синтаксис	<code>uwsgi_ssl_conf_command</code> <i>имя значение</i> ;
По умолчанию	—
Контекст	http, server, location

Задаёт произвольные конфигурационные *команды* OpenSSL при установлении соединения с uwsgi HTTPS-сервером.

Примечание

Директива поддерживается при использовании OpenSSL 1.0.2 и выше. Чтобы настроить шифры TLS 1.3 в OpenSSL, используйте команду `ciphersuites`.

На одном уровне может быть указано несколько директив `uwsgi_ssl_conf_command`. Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы `uwsgi_ssl_conf_command`.

Предупреждение

Следует учитывать, что изменение настроек OpenSSL напрямую может привести к неожиданному поведению.

uwsgi_ssl_crl

Синтаксис	<code>uwsgi_ssl_crl</code> <i>файл</i> ;
По умолчанию	—
Контекст	http, server, location

Указывает файл с отозванными сертификатами (CRL) в формате PEM, используемыми при проверке сертификата suwsgi-сервера.

uwsgi_ssl_name

Синтаксис	<code>uwsgi_ssl_name</code> <i>имя</i> ;
По умолчанию	<code>uwsgi_ssl_name</code> <code>`имя хоста из uwsgi_pass`</code>
Контекст	http, server, location

Позволяет переопределить имя сервера, используемое при проверке сертификата uwsgi HTTPS-сервера, а также для передачи его через SNI при установлении соединения с suwsgi-сервером.

Имя сервера также можно задать с помощью переменных.

По умолчанию используется имя хоста из URL'a, заданного директивой `uwsgi_pass`.

uwsgi_ssl_password_file

Синтаксис	<code>uwsgi_ssl_password_file <i>файл</i>;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт файл с паролями от секретных ключей, где каждый пароль указан на отдельной строке. Пароли применяются по очереди в момент загрузки ключа.

uwsgi_ssl_protocols

Синтаксис	<code>uwsgi_ssl_protocols [SSLv2] [SSLv3] [TLSv1] [TLSv1.1] [TLSv1.2] [TLSv1.3];</code>
По умолчанию	<code>uwsgi_ssl_protocols TLSv1.2 TLSv1.3;</code>
Контекст	http, server, location

Разрешает указанные протоколы для запросов к uwsgi-серверу.

uwsgi_ssl_server_name

Синтаксис	<code>uwsgi_ssl_server_name on off;</code>
По умолчанию	<code>uwsgi_ssl_server_name off;</code>
Контекст	http, server, location

Разрешает или запрещает передачу имени сервера, заданного директивой `uwsgi_ssl_name`, через расширение [Server Name Indication](#) протокола TLS (SNI, RFC 6066) при установлении соединения с защищённым uwsgi-сервером.

uwsgi_ssl_session_reuse

Синтаксис	<code>uwsgi_ssl_session_reuse on off;</code>
По умолчанию	<code>uwsgi_ssl_session_reuse on;</code>
Контекст	http, server, location

Определяет, использовать ли повторно SSL-сессии при работе с uwsgi-сервером. Если в логах появляются ошибки `"SSL3_GET_FINISHED:digest check failed"`, то можно попробовать выключить повторное использование сессий.

uwsgi_ssl_trusted_certificate

Синтаксис	<code>uwsgi_ssl_trusted_certificate <i>файл</i>;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт файл с доверенными сертификатами СА в формате PEM, используемыми при проверке сертификата uwsgi HTTPS-сервера.

uwsgi_ssl_verify

Синтаксис	<code>uwsgi_ssl_verify on off;</code>
По умолчанию	<code>uwsgi_ssl_verify off;</code>
Контекст	<code>http, server, location</code>

Разрешает или запрещает проверку сертификата suwsgi-сервера.

uwsgi_ssl_verify_depth

Синтаксис	<code>uwsgi_ssl_verify_depth <i>число</i>;</code>
По умолчанию	<code>uwsgi_ssl_verify_depth 1;</code>
Контекст	<code>http, server, location</code>

Устанавливает глубину проверки в цепочке сертификатов suwsgi-сервера.

uwsgi_store

Синтаксис	<code>uwsgi_store on off <i>строка</i>;</code>
По умолчанию	<code>uwsgi_store off;</code>
Контекст	<code>http, server, location</code>

Разрешает сохранение на диск файлов.

<code>on</code>	сохраняет файлы в соответствии с путями, указанными в директивах <i>alias</i> или <i>root</i>
<code>off</code>	запрещает сохранение файлов

Имя файла можно задать явно с помощью строки с переменными:

```
uwsgi_store /data/www$original_uri;
```

Время изменения файлов выставляется согласно полученному полю **Last-Modified** в заголовке ответа. Ответ сначала записывается во временный файл, а потом этот файл переименовывается. Временный файл и постоянное место хранения ответа могут располагаться на разных файловых системах. Однако нужно учитывать, что в этом случае вместо дешевой операции переименовывания в пределах одной файловой системы файл копируется с одной файловой системы на другую. Поэтому лучше, если сохраняемые файлы будут находиться на той же файловой системе, что и каталог с временными файлами, задаваемый директивой `uwsgi_temp_path` для данного *location*.

Директиву можно использовать для создания локальных копий статических неизменяемых файлов:

```
location /images/ {
    root          /data/www;
    error_page    404 = /fetch$uri;
}

location /fetch/ {
    internal;

    uwsgi_pass    backend:9000;
```



```
...

uwsgi_store      on;
uwsgi_store_access user:rw group:rw all:r;
uwsgi_temp_path  /data/temp;

alias            /data/www/;
}
```

uwsgi_store_access

Синтаксис	<code>uwsgi_store_access пользователи:права ...;</code>
По умолчанию	<code>uwsgi_store_access user:rw;</code>
Контекст	http, server, location

Задаёт права доступа для создаваемых файлов и каталогов, например,

```
uwsgi_store_access user:rw group:rw all:r;
```

Если заданы какие-либо права для *group* или *all*, то права для *user* указывать необязательно:

```
uwsgi_store_access group:rw all:r;
```

uwsgi_tcp_congestion

Синтаксис	<code>uwsgi_tcp_congestion алгоритм;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт алгоритм управления перегрузкой TCP (параметр сокета TCP_CONGESTION) для исходящих соединений с uwsgi-сервером; к соединениям через UNIX-сокеты параметр не применяется.

Имя алгоритма не проверяется на этапе конфигурации: если алгоритм неизвестен или недоступен, при установке исходящего соединения в лог записывается сообщение уровня **alert**, а для соединения действует системный алгоритм по умолчанию.

uwsgi_temp_file_write_size

Синтаксис	<code>uwsgi_temp_file_write_size размер;</code>
По умолчанию	<code>uwsgi_temp_file_write_size 8k 16k;</code>
Контекст	http, server, location

Ограничивает размер данных, сбрасываемых во временный файл за один раз, при включенной буферизации ответов uwsgi-сервера во временные файлы. По умолчанию размер ограничен двумя буферами, заданными директивами `uwsgi_buffer_size` и `uwsgi_buffers`. Максимальный размер временного файла задается директивой `uwsgi_max_temp_file_size`.

uwsgi_temp_path

Синтаксис	<code>uwsgi_temp_path <i>путь</i> [<i>уровень1</i> [<i>уровень2</i> [<i>уровень3</i>]]]`;</code>
По умолчанию	<code>uwsgi_temp_path uwsgi_temp;</code> (путь зависит от параметра сборки <code>--http-uwsgi-temp-path</code>)
Контекст	http, server, location

Задаёт имя каталога для хранения временных файлов с данными, полученными от uwsgi-серверов. В каталоге может использоваться иерархия подкаталогов до трёх уровней. Например, при такой конфигурации

```
uwsgi_temp_path /spool/angie/uwsgi_temp 1 2;
```

временный файл будет следующего вида:

```
/spool/angie/uwsgi_temp/7/45/00000123457
```

См. также параметр `use_temp_path` директивы `uwsgi_cache_path`.

HTTP/2

Обеспечивает поддержку HTTP/2.

Пример конфигурации

```
server {
    listen 443 ssl;

    http2 on;

    ssl_certificate server.crt;
    ssl_certificate_key server.key;
}
```

Примечание

Чтобы принимать HTTP/2-соединения по TLS, необходимо наличие поддержки расширения "Application-Layer Protocol Negotiation" (ALPN) протокола TLS, появившейся в [OpenSSL](#) версии 1.0.2.

Если директива `ssl_prefer_server_ciphers` установлена в значение "on", шифры должны быть настроены таким образом, чтобы соответствовать списку запрещённых наборов шифров [RFC 9113](#), [Appendix A](#) а также поддерживаться клиентами.

Директивы

http2

Синтаксис	<code>http2 on off;</code>
По умолчанию	<code>http2 off;</code>
Контекст	http, server

Разрешает протокол HTTP/2.

http2_body_preload_size

Синтаксис	http2_body_preload_size размер;
По умолчанию	—
Контекст	http, server

Задаёт размер буфера для каждого запроса, в который может сохраняться тело запроса до того, как оно начнет обрабатываться.

http2_chunk_size

Синтаксис	http2_chunk_size размер;
По умолчанию	http2_chunk_size 8k;
Контекст	http, server, location

Задаёт максимальный размер частей, на которое будет разделяться тело ответа. Слишком маленькое значение может привести к росту накладных расходов. Слишком большое значение может негативно сказаться на приоритизации из-за блокировки очереди.

http2_max_concurrent_pushes

Устарело

Синтаксис	http2_max_concurrent_pushes <i>число</i> ;
По умолчанию	http2_max_concurrent_pushes 10;
Контекст	http, server

Ограничивает максимальное число параллельных push-запросов в соединении.

http2_max_concurrent_streams

Синтаксис	http2_max_concurrent_streams <i>число</i> ;
По умолчанию	http2_max_concurrent_streams 128;
Контекст	http, server

Задаёт максимальное число параллельных HTTP/2-потоков в соединении.

http2_push

Устарело

Синтаксис	<code>http2_push uri off;</code>
По умолчанию	<code>http2_push off;</code>
Контекст	<code>http, server, location</code>

Заблаговременно отправляет (**push**) запрос к заданному `uri` вместе с ответом на оригинальный запрос. Будут обработаны только относительные URI с абсолютными путями, например:

```
http2_push /static/css/main.css;
```

В значении `uri` можно использовать переменные.

На одном уровне конфигурации можно указать несколько `http2_push` директив. Параметр `off` отменяет действие унаследованных с предыдущего уровня конфигурации директив `http2_push`.

http2_push_preload

Устарело

Синтаксис	<code>http2_push_preload on off;</code>
По умолчанию	<code>http2_push_preload off;</code>
Контекст	<code>http, server, location</code>

Разрешает автоматическое преобразование **preload links**, указанных в полях "Link" заголовка ответа, в **push**-запросы.

http2_recv_buffer_size

Синтаксис	<code>http2_recv_buffer_size размер;</code>
По умолчанию	<code>http2_recv_buffer_size 256k;</code>
Контекст	<code>http</code>

Задаёт размер входного буфера для рабочего процесса.

Встроенные переменные

Модуль `http_v2` поддерживает следующие встроенные переменные:

`$http2`

согласованный идентификатор протокола:

<code>h2</code>	для HTTP/2 через TLS
<code>h2c</code>	для HTTP/2 через незашифрованный TCP
<code>" "</code>	пустая строка для остальных случаев

HTTP/3

Обеспечивает поддержку протокола HTTP/3 для соединений с клиентами, а также для соединений с проксируемыми серверами, настраиваемых при помощи следующих директив из модуля Proxy:

- `proxy_http3_hq`
- `proxy_http3_max_concurrent_streams`
- `proxy_http3_max_table_capacity`
- `proxy_http3_stream_buffer_size`
- `proxy_http_version`
- `proxy_pass`
- `proxy_quic_active_connection_id_limit`
- `proxy_quic_gso`
- `proxy_quic_host_key`

Пример конфигурации

```
http {
    log_format quic '$remote_addr - $remote_user [$time_local] '
                   '"$request" $status $body_bytes_sent '
                   '"$http_referer" "$http_user_agent" "$http3"';

    access_log logs/access.log quic;

    server {
        # для лучшей совместимости рекомендуется
        # использовать одинаковый порт для http/3 и https
        listen 8443 quic reuseport;
        listen 8443 ssl;

        ssl_certificate     certs/example.com.crt;
        ssl_certificate_key certs/example.com.key;

        location / {
            # используется для объявления о поддержке http/3
            add_header Alt-Svc 'h3=":8443"; ma=86400';
        }
    }
}
```

Примечание

Чтобы принимать HTTP/3-соединения по TLS, необходимо наличие поддержки протокола TLSv1.3, появившейся в OpenSSL версии 1.1.1.

Для поддержки 0-RTT нужна библиотека OpenSSL версии 3.5.1 или выше. Также возможно использование библиотек BoringSSL, LibreSSL или QuicTLS.

До версии 1.29.1 поддержка 0-RTT не могла быть разрешена при использовании OpenSSL независимо от значения директивы `ssl_early_data`.

Для запросов HTTP/3, если заголовок Host не передан, переменная `$http_host` инициализируется значением псевдозаголовка `:authority`.

Кроме того, опцию `reuseport` можно указать только в одной из директив `listen ... quic` на сервере. Остальные директивы `listen ... quic` должны быть указаны без нее.

Директивы

http3

Синтаксис	<code>http3 on off;</code>
По умолчанию	<code>http3 on;</code>
Контекст	<code>http, server</code>

Разрешает согласование протокола HTTP/3.

http3_hq

Синтаксис	<code>http3_hq on off;</code>
По умолчанию	<code>http3_hq off;</code>
Контекст	<code>http, server</code>

Разрешает согласование протокола HTTP/0.9, используемого в функциональных тестах QUIC.

Предупреждение

Включайте этот режим только для запуска специализированных тестов, которым в явной форме необходим данный режим.

http3_max_concurrent_streams

Синтаксис	<code>http3_max_concurrent_streams <i>число</i>;</code>
По умолчанию	<code>http3_max_concurrent_streams 128;</code>
Контекст	<code>http, server</code>

Инициализирует настройки HTTP/3 и QUIC, а также задает максимальное число параллельных HTTP/3-потоков в соединении.

http3_max_table_capacity

Синтаксис	<code>http3_max_table_capacity <i>число</i>;</code>
Значение по умолчанию	<code>http3_max_table_capacity 4096;</code>
Контекст	<code>http, server</code>

Определяет емкость динамической таблицы для серверных соединений.

i Примечание

Похожая директива `proxy_http3_max_table_capacity` задает это значение для прокси-соединений. Чтобы избежать ошибок, использование динамической таблицы отключается при включенном кэшировании в режиме проксирования.

`http3_stream_buffer_size`

Синтаксис	<code>http3_stream_buffer_size</code> размер;
По умолчанию	<code>http3_stream_buffer_size 64k</code> ;
Контекст	http, server

Задаёт размер буфера, используемого для чтения и записи QUIC-потокa.

`quic_active_connection_id_limit`

Синтаксис	<code>quic_active_connection_id_limit</code> <i>число</i> ;
По умолчанию	<code>quic_active_connection_id_limit 2</code> ;
Контекст	http, server

Устанавливает значение транспортного параметра QUIC `active_connection_id_limit`. Это максимальное значение ID соединений, возможное для хранения на сервере.

`quic_bpf`

Синтаксис	<code>quic_bpf on off</code> ;
По умолчанию	<code>quic_bpf off</code> ;
Контекст	main

Разрешает маршрутизацию пакетов QUIC при помощи eBPF. Если маршрутизация включена, то обеспечивается поддержка миграции QUIC-соединений.

i Примечание

Директива поддерживается только на Linux 5.7+.

`quic_gso`

Синтаксис	<code>quic_gso on off</code> ;
По умолчанию	<code>quic_gso off</code> ;
Контекст	http, server

Разрешает отправку оптимизированного пакетного режима при помощи segmentation offloading.

Примечание

Оптимизированная отправка поддерживается только на Linux с поддержкой UDP_SEGMENT.

quic_host_key

Синтаксис	quic_host_key <i>файл</i> ;
По умолчанию	—
Контекст	http, server

Задаёт *файл* с секретным ключом, применяемым при шифровании stateless reset и address validation токенов. По умолчанию создается случайный ключ при каждой перезагрузке. Токены, созданные при помощи старых ключей, не принимаются.

quic_retry

Синтаксис	quic_retry on off;
По умолчанию	quic_retry off;
Контекст	http, server

Разрешает функциональность [QUIC Address Validation](#), в том числе отправку нового токена в *Retry*-пакете или *NEW_TOKEN* frame и валидацию токена, полученного в *Initial*-пакете.

Встроенные переменные

Модуль *http_v3* поддерживает следующие встроенные переменные:

\$http3

согласованный идентификатор протокола:

h3	для HTTP/3-соединений
hq	для hq-соединений
""	пустая строка для остальных случаев

\$quic_connection

порядковый номер QUIC-соединения

4.10.8.5 Потокковые модули

<i>Stream</i>	Основная функциональность потокового сервера для балансировки протоколов TCP и UDP на уровне L4.
<i>Access</i>	Контроль доступа на основе IP-адресов и диапазонов CIDR.
<i>ACME</i>	Автоматическое получение и обновление SSL-сертификатов по протоколу ACME для потоковых серверов.
<i>Geo</i>	Преобразование IP-адресов в заданные значения переменных.
<i>Limit Conn</i>	Ограничение числа одновременных соединений для защиты от перегрузки.
<i>Log</i>	Настройка журнала сессий для отслеживания обращений к ресурсам с целью мониторинга и анализа.
<i>Map</i>	Преобразование переменных на основе predetermined пар "ключ-значение".
<i>Metric</i>	Пользовательские числовые метрики в API статистики реального времени.
<i>MQTT Preread</i>	Чтение идентификатора клиента и имени пользователя из соединения по протоколу MQTT до момента принятия решения о балансировке.
<i>Pass</i>	Передача принятых соединений напрямую в настроенный слушающий сокет.
<i>Proxy</i>	Настройка проксирования к другим серверам.
<i>RDP Preread</i>	Чтение cookie из соединения по протоколу RDP до момента принятия решения о балансировке.
<i>RealIP</i>	Определение адреса и порта клиента при работе за другим прокси-сервером.
<i>Return</i>	Отправка в ответ клиенту при его подключении заданного значения без дальнейшего проксирования.
<i>Set</i>	Установка заданных значений переменных.
<i>Split Clients</i>	Создание переменных для A/B-тестирования, канареечных релизов, шардинга и других сценариев, требующих разделения по пропорциональным группам.
<i>SSL</i>	Терминация протоколов SSL/TLS и DTLS.
<i>SSL Preread</i>	Извлечение информации из сообщения ClientHello без терминации SSL/TLS и до момента принятия решения о балансировке.
<i>Upstream</i>	Настройка групп проксируемых серверов для балансировки нагрузки.
<i>Upstream Probe</i>	Настройка активных проверок работоспособности для групп проксируемых серверов.

Потоковый модуль

Access

Модуль позволяет ограничить доступ для определенных адресов клиентов.

Пример конфигурации

```
server {
    ...
    deny 192.168.1.1;
    allow 192.168.1.0/24;
    allow 10.1.1.0/16;
    allow 2001:0db8::/32;
    deny all;
}
```

Правила проверяются в порядке их записи до первого соответствия. В данном примере доступ разрешен только для IPv4-сетей 10.1.1.0/16 и 192.168.1.0/24, кроме адреса 192.168.1.1, и для IPv6-сети 2001:0db8::/32.

Директивы

allow

Синтаксис	<code>allow адрес CIDR unix: all;</code>
По умолчанию	—
Контекст	stream, server

Разрешает доступ для указанной сети или адреса. Если указано специальное значение `unix:`, разрешает доступ для всех UNIX-сокетов.

deny

Синтаксис	<code>deny адрес CIDR unix: all;</code>
По умолчанию	—
Контекст	stream, server

Запрещает доступ для указанной сети или адреса. Если указано специальное значение `unix:`, запрещает доступ для всех UNIX-сокетов.

ACME

Позволяет автоматически получать сертификаты с использованием протокола [ACME](#) для серверов, определенных в контексте `stream`.

Примечание

Для корректной работы блок `stream` должен располагаться после блока `http`. Это связано с тем, что потоковый модуль использует определения клиентов, созданные при разборе HTTP-конфигурации.

Пример конфигурации

В этом примере сертификаты, полученные ACME-клиентом `example` (объявленным в блоке `http`), защищают TCP-сервер; по умолчанию используется HTTP-проверка:

```
# HTTP-часть
http {

    resolver 127.0.0.53;

    # ACME-клиент для потоковой части
    acme_client example https://acme-v02.api.letsencrypt.org/directory;

    # Сервер для HTTP-проверки
    server {

        listen 80;
        return 444;
    }
}
```

```
# Потокковая часть
stream {

    server {

        listen 12345 ssl;
        proxy_pass backend_upstream;

        ssl_certificate $acme_cert_example;
        ssl_certificate_key $acme_cert_key_example;

        server_name example.com www.example.com;
        acme example; # ссылка на АСМЕ-клиент, определенный в HTTP-части
    }

    upstream backend_upstream {

        server 127.0.0.1:54321;
    }
}
```

Примеры конфигурации и инструкции по настройке, в том числе для DNS-проверки, см. в разделе *АСМЕ в потоковом модуле*.

Директивы

acme

Синтаксис	<code>acme имя;</code>
По умолчанию	—
Контекст	server

Указывает клиент АСМЕ из HTTP-модуля, который получает сертификат для допустимых идентификаторов этого блока *server*. Единый сертификат охватывает все допустимые доменные имена и IP-адреса, указанные в директивах *server_name* всех блоков *server*, которые ссылаются на клиент с именем *имя*; если изменится конфигурация *server_name*, сертификат будет обновлен для учета изменений.

При каждом запуске Angie или перезагрузке его конфигурации для всех идентификаторов, у которых отсутствует действующий сертификат, запрашиваются новые сертификаты. Возможные причины включают истечение срока действия сертификатов, отсутствие файлов или невозможность прочитать их, а также изменения в настройках сертификатов.

Примечание

Имена, заданные через регулярные выражения, не поддерживаются и будут пропускаться.

Домены со звездочкой поддерживаются только в режиме `challenge=dns` в `acme_client`.

Адреса IPv4 и IPv6 поддерживаются, если клиент не использует `challenge=dns`. При включении DNS-проверки IP-адреса пропускаются.

Эта директива может быть указана несколько раз для загрузки сертификатов разных типов, например RSA и ECDSA:

```
server {

    listen 12345 ssl;
    server_name example.com www.example.com;

    ssl_certificate $acme_cert_rsa;
    ssl_certificate_key $acme_cert_key_rsa;

    ssl_certificate $acme_cert_ecdsa;
    ssl_certificate_key $acme_cert_key_ecdsa;

    acme rsa;
    acme ecdsa;
}
```

Встроенные переменные

`$acme_cert_<имя>`

Содержимое последнего файла сертификата (если он есть), полученного клиентом с этим *именем*.

`$acme_cert_key_<имя>`

Содержимое файла ключа сертификата, используемого клиентом с этим *именем*.

Примечание

Файл сертификата доступен, только если клиент АСМЕ получил хотя бы один сертификат, а вот файл ключа доступен сразу после запуска.

Geo

Создает переменные, значения которых зависят от IP-адреса клиента.

Пример конфигурации

```
geo $geo {
    default          0;

    127.0.0.1        2;
    192.168.1.0/24    1;
    10.1.0.0/16       1;

    ::1              2;
    2001:0db8::/32    1;
}
```

Директивы

geo

Синтаксис	<code>geo [<i>\$адрес</i>] <i>\$переменная</i> { ... }</code>
По умолчанию	—
Контекст	stream

Описывает для указанной переменной зависимость значения от IP-адреса клиента. По умолчанию адрес берется из переменной `$remote_addr`, но его также можно получить из другой переменной, например:

```
geo $arg_remote_addr $geo {
    ...;
}
```

Примечание

Поскольку переменные вычисляются только в момент использования, само по себе наличие даже большого числа объявлений переменных `geo` не влечет за собой никаких дополнительных расходов на обработку соединений.

Если значение переменной не представляет из себя правильный IP-адрес, то используется адрес "255.255.255.255".

Адреса задаются либо префиксами в формате CIDR (включая одиночные адреса), либо в виде диапазонов.

Также поддерживаются следующие специальные параметры:

delete	Удаляет описанную сеть.
default	Значение переменной, если адрес клиента не соответствует ни одному из заданных адресов. При задании адресов в формате CIDR вместо default можно использовать 0.0.0.0/0 и ::/0. Если параметр default не указан, значением по умолчанию будет пустая строка.
include	Включает файл с адресами и значениями. Имя файла может содержать glob-шаблоны, в этом случае подключаются все совпадающие файлы. Включений может быть несколько.
ranges	Указывает, что адреса задаются в виде диапазонов. Этот параметр должен быть первым. Для ускорения загрузки гео-базы нужно располагать адреса в порядке возрастания.
volatile	Указывает, что переменная не кэшируется.

Пример:

```
geo $country {
    default      ZZ;
    include      conf/geo.conf;
    delete      127.0.0.0/16;

    127.0.0.0/24 US;
    127.0.0.1/32  RU;
    10.1.0.0/16   RU;
    192.168.1.0/24 UK;
}
```

В файле `conf/geo.conf` могут быть такие строки:

```
10.2.0.0/16    RU;
192.168.2.0/24 RU;
```

В качестве значения выбирается максимальное совпадение, например, для адреса 127.0.0.1 будет выбрано значение RU, а не US.

Пример описания диапазонов:

```
geo $country {
    ranges;
    default                ZZ;
    127.0.0.0-127.0.0.0    US;
    127.0.0.1-127.0.0.1    RU;
    127.0.0.2-127.0.0.255  US;
    10.1.0.0-10.1.255.255  RU;
    192.168.1.0-192.168.1.255 UK;
}
```

Limit Conn

Позволяет ограничить число соединений по заданному ключу, в частности, число соединений с одного IP-адреса.

Пример конфигурации

```
stream {
    limit_conn_zone $binary_remote_addr zone=addr:10m;

    ...

    server {

        ...

        limit_conn        addr 1;
        limit_conn_log_level error;
    }
}
```

Директивы

limit_conn

Синтаксис	<code>limit_conn зона число;</code>
По умолчанию	—
Контекст	stream, server

Задаёт зону разделяемой памяти и максимально допустимое число соединений для одного значения ключа. При превышении этого числа сервер закроет соединение. Например, директивы

```
limit_conn_zone $binary_remote_addr zone=addr:10m;

server {
    ...
    limit_conn addr 1;
}
```

разрешают одновременно обрабатывать не более одного соединения с одного IP-адреса.

Допустимо одновременное указание нескольких директив `limit_conn`, при этом будет срабатывать любое из ограничений.

Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы `limit_conn`.

limit_conn_dry_run

Синтаксис	limit_conn_dry_run on off;
По умолчанию	limit_conn_dry_run off;
Контекст	stream, server

Включает режим пробного запуска. В данном режиме число соединений не ограничивается, однако в зоне разделяемой памяти текущее число избыточных соединений учитывается как обычно.

limit_conn_log_level

Синтаксис	limit_conn_log_level info notice warn error;
По умолчанию	limit_conn_log_level error;
Контекст	stream, server

Задаёт желаемый уровень записи в лог случаев ограничения числа соединений.

limit_conn_zone

Синтаксис	limit_conn_zone <i>ключ</i> zone = <i>название:размер</i> ;
По умолчанию	—
Контекст	stream

Задаёт параметры зоны разделяемой памяти, которая хранит состояние для разных значений ключа. Состояние в частности содержит текущее число соединений. В качестве ключа можно использовать переменные. Запросы с пустым значением ключа не учитываются.

Пример использования:

```
limit_conn_zone $binary_remote_addr zone=addr:10m;
```

Здесь в качестве ключа используется IP-адрес клиента, задаваемый переменной \$binary_remote_addr.

Длина значения \$binary_remote_addr равна 4 байтам для IPv4-адресов или 16 байтам для IPv6-адресов. При этом размер состояния всегда равен 32 или 64 байтам на 32-битных платформах и 64 байтам на 64-битных. В зоне размером 1 мегабайт может разместиться около 32 тысяч состояний размером 32 байта или 16 тысяч состояний размером 64 байта. При переполнении зоны сервер закрывает соединение.

Встроенные переменные

\$limit_conn_status

хранит результат ограничения числа соединений: PASSED, REJECTED или REJECTED_DRY_RUN

Log

Модуль записывает логи запросов в указанном формате.

Пример конфигурации

```
log_format basic '$remote_addr [$time_local] '
                '$protocol $status $bytes_sent $bytes_received '
                '$session_time';

access_log /spool/logs/angie-access.log basic buffer=32k;
```

Директивы

access_log

Синтаксис	<code>access_log <i>путь</i> [<i>формат</i> [<i>buffer=размер</i>] [<i>gzip[=степень]</i>] [<i>flush=время</i>] [<i>if=условие</i>]];</code> <code>access_log off;</code>
По умолчанию	<code>access_log off;</code>
Контекст	stream, server

Задаёт *путь*, *формат* и настройки буферизованной записи в лог. На одном уровне конфигурации может использоваться несколько логов. Запись в syslog настраивается указанием префикса "*syslog:*" в первом параметре. Специальное значение *off* отменяет все директивы *access_log* для текущего уровня.

Если задан размер буфера с помощью параметра *buffer* или указан параметр *gzip*, то запись будет буферизованной.

Предупреждение

Размер буфера должен быть не больше размера атомарной записи в дисковый файл. Для FreeBSD этот размер неограничен.

При включенной буферизации данные записываются в файл:

- если очередная строка лога не помещается в буфер;
- если данные в буфере находятся дольше интервала времени, заданного параметром *flush*;
- при переоткрытии лог-файла или завершении рабочего процесса.

Если задан параметр *gzip*, то буфер будет сжиматься перед записью в файл. Степень сжатия может быть задана в диапазоне от 1 (быстрее, но хуже сжатие) до 9 (медленнее, но лучше сжатие). По умолчанию используются буфер размером 64K байт и степень сжатия 1. Данные сжимаются атомарными блоками, и в любой момент времени лог-файл может быть распакован или прочитан с помощью утилиты "*zcat*".

Пример:

```
access_log /path/to/log.gz basic gzip flush=5m;
```

Примечание

Для поддержки gzip-сжатия логов Angie должен быть собран с библиотекой *zlib*.

В пути файла можно использовать переменные, но такие логи имеют некоторые ограничения:

- пользователь, с правами которого работают рабочие процессы, должен иметь права на создание файлов в каталоге с такими логами;

- не работает буферизация;
- файл открывается для каждой записи в лог и сразу же после записи закрывается. Следует однако иметь в виду, что поскольку дескрипторы часто используемых файлов могут храниться в кэше, то при ротации логов в течение времени, заданного параметром *valid* директивы *open_log_file_cache*, запись может продолжаться в старый файл.

Параметр *if* включает условную запись в лог. Сессия не будет записываться в лог, если результатом вычисления условия является *"0"* или пустая строка.

log_format

Синтаксис	<code>log_format имя [escape=default json none] строка ...;</code>
По умолчанию	—
Контекст	stream

Задаёт формат лога, например:

```
log_format proxy '$remote_addr [$time_local] '
                '$protocol $status $bytes_sent $bytes_received '
                '$session_time "$upstream_addr" '
                '"$upstream_bytes_sent" "$upstream_bytes_received" "$upstream_
↪connect_time";
```

Параметр **escape** позволяет задать экранирование символов **json** или **default** в переменных, по умолчанию используется **default**. Значение **none** отключает экранирование символов.

При использовании **default** символы *"", "\",* а также символы со значениями меньше 32 или больше 126 экранируются как *"\xXX"*. Если значение переменной не найдено, то в качестве значения в лог будет записываться дефис *"-"*.

При использовании **json** экранируются все символы, недопустимые в JSON строках: символы *"",* и *"\"* экранируются как *"\"* и *"\""*, символы со значениями меньше 32 экранируются как *"\n",* *"\r",* *"\t",* *"\b",* *"\f"* или *"u00XX"*.

open_log_file_cache

Синтаксис	<code>open_log_file_cache max=N [inactive=время] [min_uses=N] [valid=время];</code>
По умолчанию	<code>open_log_file_cache off;</code>
Контекст	http, server, location

Задаёт кэш, в котором хранятся дескрипторы файлов часто используемых логов, имена которых заданы с использованием переменных. Параметры:

max	Задаёт максимальное число дескрипторов в кэше; при переполнении кэша наименее востребованные (LRU) дескрипторы закрываются.
inactive	Задаёт время, после которого кэшированный дескриптор закрывается, если к нему не было обращений в течение этого времени; по умолчанию — 10 секунд.
min_uses	Задаёт минимальное число использований файла в течение времени, заданного параметром inactive , после которого дескриптор файла будет оставаться открытым в кэше; по умолчанию — 1.
valid	Указывает, через какое время нужно проверять, что файл ещё существует под тем же именем; по умолчанию — 60 секунд.
off	Запрещает кэширование.

Пример использования:

```
open_log_file_cache max=1000 inactive=20s valid=1m min_uses=2;
```

Map

Создает переменные, значения которых зависят от значений других переменных.

Пример конфигурации

```
map $remote_addr $limit {
    127.0.0.1      "";
    default        $binary_remote_addr;
}

limit_conn_zone $limit zone=addr:10m;
limit_conn addr 1;
```

Директивы

map

Синтаксис	<code>map строка \$переменная { ... };</code>
По умолчанию	—
Контекст	stream

Создает новую переменную. Ее значение зависит от первого параметра, заданного строкой с переменными, например:

```
set $var1 "foo";
set $var2 "bar";

map $var1$var2 $new_variable {
    default "foobar_value";
}
```

Здесь переменная `$new_variable` будет иметь значение, составленное из двух переменных `$var1` и `$var2`, или значение по умолчанию, если эти переменные не определены.

Примечание

Поскольку переменные вычисляются только в момент использования, само по себе наличие даже большого числа объявлений переменных `map` не влечет за собой никаких дополнительных расходов на обработку запросов.

Параметры внутри блока `map` задают соответствие между исходными и результирующими значениями.

Исходные значения задаются строками или регулярными выражениями.

Строки проверяются без учета регистра.

Перед регулярным выражением ставится символ `~`, если при сравнении следует учитывать регистр символов, либо символы `~*`, если регистр символов учитывать не нужно. Регулярное выражение может содержать именованные и позиционные группы захвата, которые могут затем использоваться в других директивах совместно с результирующей переменной.

Если исходное значение совпадает с именем одного из специальных параметров, описанных ниже, перед ним следует поставить символ \.

В качестве результирующего значения можно указать текст, переменную и их комбинации.

Также поддерживаются следующие специальные параметры:

default <i>значение</i>	задает результирующее значение, если исходное значение не совпадает ни с одним из перечисленных. Если параметр default не указан, результирующим значением по умолчанию будет пустая строка.
hostnames	указывает, что в качестве исходных значений можно использовать маску для первой или последней части имени хоста. Этот параметр следует указывать перед списком значений.

Например,

```
*.example.com 1;
example.*      1;
```

Вместо двух записей

```
example.com    1;
*.example.com  1;
```

можно использовать одну:

```
.example.com 1;
```

include <i>файл</i>	включает файл со значениями. Включений может быть несколько.
volatile	указывает, что переменная не кэшируется.

Если исходному значению соответствует несколько из указанных вариантов, например, одновременно подходят и маска, и регулярное выражение, будет выбран первый подходящий вариант в следующем порядке приоритета:

1. Строковое значение без маски.
2. Самое длинное строковое значение с маской в начале, например `*.example.com`.
3. Самое длинное строковое значение с маской в конце, например `mail.*`.
4. Первое подходящее регулярное выражение (в порядке следования в конфигурационном файле).
5. Значение по умолчанию (**default**).

map_hash_bucket_size

Синтаксис	<code>map_hash_bucket_size</code> размер;
По умолчанию	<code>map_hash_bucket_size 32 64 128</code> ;
Контекст	<code>stream</code>

Задаёт размер корзины в хэш-таблицах для переменных `map`. Значение по умолчанию зависит от размера строки кэша процессора. Подробнее настройка хэш-таблиц обсуждается отдельно.

map_hash_max_size

Синтаксис	map_hash_max_size размер;
По умолчанию	map_hash_max_size 2048;
Контекст	stream

Задаёт максимальный размер хэш-таблиц для переменных *map*. Подробнее настройка хэш-таблиц обсуждается отдельно.

Metric

Модуль `ngx_stream_metric_module` позволяет создавать вычисляемые в реальном времени произвольные метрики для потокового (TCP и UDP) трафика. Значения таких метрик сохраняются в разделяемой памяти и отображаются в реальном времени в ветке API `/status/stream/metric_zones/`. Поддерживаются различные типы агрегации данных (счетчики, гистограммы, скользящие средние и др.) с группировкой по произвольным ключам.

Схема ответа приведена в справочнике API зон потоковых метрик.

Примечание

При перезагрузке конфигурации зона, созданная директивой `metric_zone` или `metric_complex_zone`, сохраняет накопленные значения только при неизменной структуре хранимых данных. Зона начинает работу пустой, если от предыдущей конфигурации отличаются размер зоны, количество метрик или режим, если изменился параметр `count` режима `average mean`, если параметр `window` режима `average mean` переключен в `off` или обратно или если изменилось количество корзин `histogram`; причина при этом записывается в *error_log* на уровне `info`. Остальные параметры, например `factor` режима `average exp`, интервал `window` режима `average mean` или границы корзин `histogram`, можно менять без потери накопленных значений.

Пример конфигурации

Подсчет соединений по адресу клиента:

```
stream {
    metric_zone connections:1m count;

    server {
        listen 12345;

        metric connections $remote_addr on=connect;
    }
}

http {
    server {
        listen 80;

        location /status/ {
            allow 127.0.0.1;
            deny all;
            api /status/;
        }
    }
}
```

Если выполнить соединение с портом 12345:

```
$ nc 127.0.0.1 12345 </dev/null
```

В реальном времени происходит обновление метрики `connections`:

```
{
  "stream": {
    "metric_zones": {
      "connections": {
        "discarded": 0,
        "metrics": {
          "127.0.0.1": 1
        }
      }
    }
  }
}
```

Директивы

metric

Синтаксис	<code>metric название ключ=значение [on=connect prered end];</code>
По умолчанию	—
Контекст	stream, server

Подсчитывает значение метрики с указанным *названием* зоны разделяемой памяти.

Параметры:

- *ключ* — произвольная строка (часто переменная), по которой группируются значения. Максимальная длина — 255 байт; более длинные ключи обрезаются. В выводе API Angie добавляет ... к любому ключу длиной 255 байт, даже если до сохранения его длина составляла ровно 255 байт. Ключ может сам содержать символы = — значением считается только текст после последнего =, поэтому все, что стоит перед ним (включая вложенные =), становится ключом;
- *значение* — число (может быть переменной), обрабатываемое выбранным режимом. Если пропущено, считается 0. Если параметр невозможно преобразовать в число, считается 1;
- *on* — необязательный параметр, указывающий, в какой момент обработки сессии происходит подсчет метрики:
 - Если *on=connect*, подсчет происходит на фазе **Pre-access**, до чтения каких-либо данных от клиента;
 - Если *on=prered*, подсчет происходит один раз за сессию — в момент, когда клиенту отправляются первые данные, полученные от проксируемого с помощью `proxy_pass` сервера или сформированные директивой `return`; к этому моменту такие модули, как `proxy_protocol`, `ssl_prered` и `mqtt_prered`, уже обработали начальные данные клиента;
 - Если *on=end* (по умолчанию), подсчет происходит на фазе **Log**, когда сессия завершается.

Директив `metric` может быть несколько. Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы `metric`. Поэтому одна директива `metric` в блоке `server` отменяет все правила, унаследованные от `stream`, на всех трех стадиях *on*, а не только на своей.

Пример использования:

```
metric sessions $remote_addr=$session_time on=end;
```

i Примечание

Метрики с пустым ключом или неправильной парой `ключ=значение` не учитываются. Пропущенное значение учитывается как 0:

```
metric foo $bar; # Эквивалентно $bar=0
```

Что полезно, например, для режима `count`, который игнорирует числовое значение и реагирует лишь на факт обновления метрики.

i Примечание

Важно помнить, что переменные вычисляются в разных фазах. Например, невозможно использовать `$upstream_bytes_sent` (количество байт, отправленных апстриму) при `on=connect` (до того, как соединение куда-либо проксировано).

metric_complex_zone

Синтаксис	<code>metric_complex_zone</code>	<i>название:размер</i>	<code>[expire=on off]</code>
	<code>[discard_key=название] { ... }</code>		
По умолчанию	—		
Контекст	stream		

Определяет *составную метрику* — набор метрик с независимыми режимами. Каждая строка в теле блока задает *название подметрики*, *режим* и необязательные *параметры* режима. Размер зоны должен составлять не менее восьми системных страниц памяти.

Пример использования:

```
metric_complex_zone sessions:1m expire=on discard_key="old" {
    # название подметрики    режим работы    параметры
    min_time                  min;
    avg_time                  average exp    factor=60;
    max_time                  max;
    total                     count;
}
```

В API дереве шаблон такой составной метрики выглядит следующим образом:

```
{
  "discarded": 3,
  "metrics": {
    "ключ1": {
      "min_time": 20,
      "avg_time": 50,
      "max_time": 80,
      "total": 2
    },
    "old": {
      "min_time": 3,
```

```

    "avg_time": 40,
    "max_time": 152,
    "total": 80
  }
}

```

metric_zone

Синтаксис	<code>metric_zone <i>название:размер</i> [expire=on off] [discard_key=<i>название</i>] <i>режим</i> [<i>параметры</i>];</code>
По умолчанию	—
Контекст	stream

Создает зону разделяемой памяти указанного *размера* с именем *название*, в которой хранятся метрики. Имя зоны является узлом в ветке `/status/stream/metric_zones/`. Размер зоны должен составлять не менее восьми системных страниц памяти.

Параметры:

- `expire=<on|off>` — поведение при переполнении зоны:
 - Если `on`, самые старые метрики по времени обновления отбрасываются, освобождая память под поступающие;
 - Если `off` (по умолчанию) — отбрасываются поступающие метрики, сохраняя информацию об установленных.
- `discard_key=<название>` — задает метрику с ключом *название*, в которой сохраняются значения отброшенных метрик. По умолчанию такая метрика не создается. Зарезервированный ключ нельзя обновлять вручную.
- *режим* — алгоритм обработки значений (см. раздел Режимы работы);
- *параметры* — дополнительная настройка выбранного режима (например, `factor` для `average exp`).

Пример использования:

```
metric_zone session_time:1m max;
```

В API дереве шаблон зоны разделяемой памяти выглядит следующим образом:

```

{
  "discarded": 0,
  "metrics": {
    "ключ1": 123,
    "ключ2": 10.5
  }
}

```

Примечание

В зоне размером 1 мегабайт при размере ключа 39 байт и с одним режимом метрики может разместиться около 8 тысяч записей с уникальным ключом.

Режимы работы

Список доступных режимов работы метрик:

- `count` — счетчик обращений;
- `gauge` — стрелка (инкремент/декремент);
- `last` — последнее поступившее значение;
- `min` — минимальное значение;
- `max` — максимальное значение;
- `average exp` — скользящее среднее (параметр `factor`);
- `average mean` — среднее за окно (параметры `window` и `count`);
- `histogram` — распределение по "бакетам" (перечень пороговых значений).

В примерах ниже используется конечная точка `/status/` из примера конфигурации. Результат каждого примера доступен по адресу `/status/stream/metric_zones/<зона>/metrics/`.

count

Каждое обновление метрики увеличивает счетчик на 1 независимо от переданного значения.

Значение по умолчанию — 0.

Примеры:

```
metric_zone count:1m count;

# В качестве составной метрики:
#
# metric_complex_zone count:1m {
#     some_metric_name count;
# }

server {
    listen 12345;

    metric count KEY;
}
```

Обновление метрики:

```
$ nc 127.0.0.1 12345 </dev/null
$ nc 127.0.0.1 12345 </dev/null
$ nc 127.0.0.1 12345 </dev/null
$ nc 127.0.0.1 12345 </dev/null
```

Ожидаемое значение метрики в API:

```
{
  "KEY": 4
}
```

gauge

Счетчик увеличивает или уменьшает свое значение в зависимости от знака переданного числа. Положительное значение увеличивает счетчик, отрицательное — уменьшает. Значение 0 не изменяет счетчик.

Значение по умолчанию — 0.

Примеры:

```
metric_zone gauge:1m gauge;

# В качестве составной метрики:
#
# metric_complex_zone gauge:1m {
#     some_metric_name gauge;
# }

server {
    listen 12351;
    metric gauge KEY;
}

server {
    listen 12352;
    metric gauge KEY=5;
}

server {
    listen 12353;
    metric gauge KEY=-5;
}

server {
    listen 12354;
    metric gauge KEY=8;
}
```

Обновление метрики:

```
$ nc 127.0.0.1 12351 </dev/null
```

Ожидаемое значение метрики в API:

```
{
  "KEY": 0
}
```

Дальнейшие обновления:

```
$ nc 127.0.0.1 12352 </dev/null
$ nc 127.0.0.1 12353 </dev/null
$ nc 127.0.0.1 12354 </dev/null
```

Ожидаемое значение метрики в API:

```
{
  "KEY": 8
}
```

last

Хранит последнее полученное значение без какой-либо агрегации. Если значение опущено, используется 0.

Примеры:

```
metric_zone last:1m last;

# В качестве составной метрики:
#
# metric_complex_zone last:1m {
#     some_metric_name last;
# }

server {
    listen 12361;
    metric last KEY;
}

server {
    listen 12362;
    metric last KEY=8000;
}

server {
    listen 12363;
    metric last KEY=37;
}

server {
    listen 12364;
    metric last KEY=-3.5;
}
```

Обновление метрики:

```
$ nc 127.0.0.1 12361 </dev/null
```

Ожидаемое значение метрики в API:

```
{
  "KEY": 0
}
```

Дальнейшие обновления:

```
$ nc 127.0.0.1 12362 </dev/null
$ nc 127.0.0.1 12363 </dev/null
$ nc 127.0.0.1 12364 </dev/null
```

Ожидаемое значение метрики в API:

```
{
  "KEY": -3.5
}
```

min

Сохраняет минимальное из двух значений — уже сохраненного и нового.

Примеры:

```
metric_zone min:1m min;
```

```
# В качестве составной метрики:
#
# metric_complex_zone min:1m {
#     some_metric_name min;
# }

server {
    listen 12371;
    metric min KEY=42.999;
}

server {
    listen 12372;
    metric min KEY=-512;
}

server {
    listen 12373;
    metric min KEY=1;
}
```

Обновление метрики:

```
$ nc 127.0.0.1 12371 </dev/null
$ nc 127.0.0.1 12372 </dev/null
$ nc 127.0.0.1 12373 </dev/null
```

Ожидаемое значение метрики в API:

```
{
  "KEY": -512
}
```

max

Сохраняет наибольшее из двух значений — уже сохраненного и нового.

Примеры:

```
metric_zone max:1m max;

# В качестве составной метрики:
#
# metric_complex_zone max:1m {
#     some_metric_name max;
# }

server {
    listen 12381;
    metric max KEY=42.999;
}

server {
    listen 12382;
    metric max KEY=-512;
}

server {
```

```
listen 12383;
metric max KEY=1;
}
```

Обновление метрики:

```
$ nc 127.0.0.1 12381 </dev/null
$ nc 127.0.0.1 12382 </dev/null
$ nc 127.0.0.1 12383 </dev/null
```

Ожидаемое значение метрики в API:

```
{
  "KEY": 42.999
}
```

average exp

Вычисляет среднее значение по алгоритму экспоненциального сглаживания.

Принимает необязательный параметр **factor=<число>** — коэффициент, по которому установленное значение учитывается при расчете среднего. Допустимы целые числа от 0 до 99. По умолчанию — 90.

Чем больше коэффициент, тем сильнее новые значения влияют на среднее. Если указать 90, то будет взято 90% от нового значения и лишь 10% от предыдущего.

Примеры:

```
metric_zone avg_exp:1m average exp factor=60;

# В качестве составной метрики:
#
# metric_complex_zone avg_exp:1m {
#   some_metric_name average exp factor=60;
# }

server {
  listen 12391;
  metric avg_exp KEY=100;
}

server {
  listen 12392;
  metric avg_exp KEY=200;
}

server {
  listen 12393;
  metric avg_exp KEY=0;
}

server {
  listen 12394;
  metric avg_exp KEY=8;
}

server {
  listen 12395;
```

```
metric avg_exp KEY=30;
}
```

Обновление метрики:

```
$ nc 127.0.0.1 12391 </dev/null
$ nc 127.0.0.1 12392 </dev/null
$ nc 127.0.0.1 12393 </dev/null
$ nc 127.0.0.1 12394 </dev/null
$ nc 127.0.0.1 12395 </dev/null
```

Ожидаемое значение метрики в API:

```
{
  "KEY": 30.16
}
```

average mean

Вычисляет среднее арифметическое. Принимает необязательные параметры `window=<off|время>` и `count=<число>`, задающие соответственно интервал времени и объем выборки для усреднения. По умолчанию: `window=off` (учитывается вся выборка) и `count=10`.

Примечание

Например, `window=5s` будет учитывать только события последних 5 секунд. Параметр `window` не может принимать значение 0. Параметр `count=число` управляет размером выборки (закэшированных значений) для более плавного подсчета среднего.

Примеры:

```
metric_zone avg_mean:1m average mean window=5s count=8;

# В качестве составной метрики:
#
# metric_complex_zone avg_mean:1m {
#   some_metric_name average mean window=5s count=8;
# }

server {
  listen 12401;
  metric avg_mean KEY=0.1;
}

server {
  listen 12402;
  metric avg_mean KEY=0.4;
}

server {
  listen 12403;
  metric avg_mean KEY=10;
}

server {
  listen 12404;
```

```
metric avg_mean KEY=1;
}
```

Обновление метрики:

```
$ nc 127.0.0.1 12401 </dev/null
$ nc 127.0.0.1 12401 </dev/null
$ nc 127.0.0.1 12402 </dev/null
$ nc 127.0.0.1 12403 </dev/null
$ nc 127.0.0.1 12404 </dev/null
$ nc 127.0.0.1 12404 </dev/null
```

Ожидаемое значение метрики в API:

```
{
  "KEY": 2.1
}
```

Если подождать 5 секунд с момента последнего обновления метрики, то ожидаемое значение метрики в API:

```
{
  "KEY": 0
}
```

histogram

Создает набор "бакетов", увеличивая соответствующий счетчик, если новое значение не превосходит порога бакета. Формат параметров — список числовых порогов. Полезен для анализа распределения, например длительности сессий.

В качестве обязательных параметров передаются *числа* — предельные значения бакетов, обычно перечисляемые в порядке возрастания.

Примечание

Значение бакета `inf` или `+Inf` можно использовать для захвата всех значений, превышающих максимальный бакет.

Примеры:

```
metric_zone hist:1m histogram 0.1 0.2 0.5 1 2 inf;

# В качестве составной метрики:
#
# metric_complex_zone hist:1m {
#   some_metric_name histogram 0.1 0.2 0.5 1 2 inf;
# }

server {
  listen 12411;
  metric hist KEY=0.25;
}

server {
  listen 12412;
  metric hist KEY=2;
```

```
}

server {
    listen 12413;
    metric hist KEY=1000;
}
```

Обновление метрики:

```
$ nc 127.0.0.1 12411 </dev/null
```

Ожидаемое значение метрики в API:

```
{
  "KEY": {
    "0.1": 0,
    "0.2": 0,
    "0.5": 1,
    "1": 1,
    "2": 1,
    "inf": 1
  }
}
```

Дальнейшее обновление:

```
$ nc 127.0.0.1 12412 </dev/null
```

Ожидаемое значение метрики в API:

```
{
  "KEY": {
    "0.1": 0,
    "0.2": 0,
    "0.5": 1,
    "1": 1,
    "2": 2,
    "inf": 2
  }
}
```

Дальнейшее обновление:

```
$ nc 127.0.0.1 12413 </dev/null
```

Ожидаемое значение метрики в API:

```
{
  "KEY": {
    "0.1": 0,
    "0.2": 0,
    "0.5": 1,
    "1": 1,
    "2": 2,
    "inf": 3
  }
}
```

Встроенные переменные

Для каждой метрики создаются переменные:

- `$metric_<name>`
- `$metric_<name>_key`
- `$metric_<name>_value`

Для составной метрики добавляется переменная:

- `$metric_<name>_value_<metric>`

`$metric_<name>`

Аналогично директиве `metric`, можно использовать сеттер переменной `$metric_<name>` для подсчета метрики. Подсчет происходит в момент установки переменной — например, через директиву `set`, которая вычисляется на фазе `Pre-access` (той же, что и `on=connect`), либо программно из модуля Stream JS, например в обработчике `js_access`.

Сеттер принимает ключ с необязательным суффиксом `=значение`. Ключ от значения отделяет последний символ `=`; без суффикса используется значение 0. Ключ и значение могут состоять из текста, переменных или их комбинаций. Такие же правила разбора использует директива `metric`.

Пример использования:

```
stream {
    metric_zone hits:1m count;

    # В этот момент добавилась переменная $metric_hits

    server {
        listen 12345;

        metric hits client=1 on=connect;
    }

    server {
        listen 12346;

        set $metric_hits client=1;

        return "$metric_hits\n";
    }
}
```

После трех соединений с портом 12345 и одного соединения с портом 12346:

```
$ nc 127.0.0.1 12345 </dev/null
$ nc 127.0.0.1 12345 </dev/null
$ nc 127.0.0.1 12345 </dev/null
$ nc 127.0.0.1 12346 </dev/null
client=4
```

Примечание

Чтение переменной `$metric_<name>` возвращает ключ вместе с текущим *подсчитанным* значением метрики, в виде `ключ=значение`, а не ту строку, которая была присвоена буквально. В примере выше буквально присваивается значение `client=1`, но поскольку режим `count` игнорирует присвоенное значение и просто увеличивает счетчик при каждом обновлении, чтение

переменной `$metric_hits` после этого возвращает `client=4` — новое значение счетчика, уже учитывающее три предыдущих обновления с порта 12345.

Также изменится значение, хранящееся в переменной `$metric_<name>_key`, на указанный ключ.

`$metric_<name>_key` и `$metric_<name>_value`

Переменные `$metric_<name>_key` и `$metric_<name>_value` задают соответственно ключ и значение. Обновление метрики происходит в момент установки значения `$metric_<name>_value`, если ключ в `$metric_<name>_key` уже задан.

Примечание

Для составной метрики значения подметрик в переменной `$metric_<name>_value` объединены при помощи разделяющего символа `,`.

Пример использования:

```
stream {
    metric_zone level:1m gauge;

    # В этот момент добавились переменные $metric_level, $metric_level_key
    # и $metric_level_value

    metric_complex_zone stats:1m {
        count count;
        min min;
        avg average exp;
    }

    # В этот момент добавились переменные $metric_stats, $metric_stats_key
    # и $metric_stats_value

    server {
        listen 12345;

        metric level sensor=10 on=connect;
    }

    server {
        listen 12346;

        set $metric_level_key "sensor";
        set $metric_level_value 5;

        # Либо: set $metric_level sensor=5;

        return "Updated with '$metric_level'\nValue='$metric_level_value'\n";
    }

    server {
        listen 12347;

        set $metric_stats_key bar;
        set $metric_stats_value 9;
    }
}
```

```
# Любо: set $metric_stats bar=9;

return "Updated with '$metric_stats'\nValues='$metric_stats_value'\n";
}
}
```

Для такой конфигурации соединение с портом 12345 (устанавливающее начальное значение стрелки в 10), за которым следует соединение с портом 12346, дает:

```
$ nc 127.0.0.1 12345 </dev/null
$ nc 127.0.0.1 12346 </dev/null
Updated with 'sensor=15'
Value='15'
```

Соединение с портом 12347 дает:

```
$ nc 127.0.0.1 12347 </dev/null
Updated with 'bar=1, 9, 9'
Values='1, 9, 9'
```

Примечание

Если в качестве значения для переменной `$metric_<name>_value` указать пустую строку, значение будет распознано как 0. Если строка состоит из символов, которые невозможно преобразовать в число, она будет распознана как 1.

Подсчет метрики произойдет только после того, как заданы значения переменных `$metric_<name>_key` и `$metric_<name>_value`.

В таком случае значение, сохраненное в `$metric_<name>`, станет равным новой, *подсчитанной* паре **ключ=значение**, а не буквально присвоенным значениям.

Значение, которое хранится в `$metric_<name>_key`, является последним указанным через переменные ключом метрики.

Значение, которое хранится в `$metric_<name>_value`, является последним подсчитанным значением метрики с ключом, установленным в `$metric_<name>_key`.

`$metric_<name>_value_<metric>`

Для составной метрики значение конкретной подметрики можно получить при помощи переменной `$metric_<name>_value_<metric>`, указав имя подметрики в качестве `<metric>`.

Пример использования:

```
stream {
    metric_complex_zone foo:1m {
        count count;
        min    min;
        avg     average exp;
    }

    # В этот момент добавилась переменная $metric_foo, а также
    # $metric_foo_key, $metric_foo_value и $metric_foo_value_count,
    # $metric_foo_value_min, $metric_foo_value_avg

    server {
        listen 12345;
```

```

    set $metric_foo_key    bar;
    set $metric_foo_value 9;

    # Любо: set $metric_foo bar=9;

    return "Updated with '$metric_foo'\nValues='$metric_foo_value'\nCount='
↪$metric_foo_value_count'\n";
  }
}

```

Для такой конфигурации соединение с портом 12345 дает:

```

$ nc 127.0.0.1 12345 </dev/null
Updated with 'bar=1, 9, 9'
Values='1, 9, 9'
Count='1'

```

Дополнительные примеры

Мониторинг соединений по протоколу

```

metric_zone protocols:1m count;

server {
    listen 12345;
    listen 12346 udp;

    metric protocols $protocol;
}

```

Ответ:

```

{
    "TCP": 340,
    "UDP": 58
}

```

Распределение времени сессии upstream

```

metric_zone upstream_time:10m expire=on histogram
    0.05 0.1 0.3 0.5 1 2 5 10 inf;

server {
    listen 12345;

    proxy_pass backend;
    metric upstream_time $upstream_addr=$upstream_session_time on=end;
}

```

Ответ:

```

{
    "discarded": 0,
    "metrics": {
        "backend1:8080": {
            "0.05": 12,
            "0.1": 28,

```

```

        "0.3": 56,
        "0.5": 78,
        "1": 92,
        "2": 97,
        "5": 99,
        "10": 100,
        "inf": 100
    }
}

```

Активные подключения

```

stream {
    metric_zone active_connections:2m gauge;

    server {
        listen 12345;

        metric active_connections service_a=1 on=connect;
        metric active_connections service_a=-1 on=end;
    }

    server {
        listen 12346;

        metric active_connections service_b=1 on=connect;
        metric active_connections service_b=-1 on=end;
    }
}

http {
    server {
        listen 8080;

        location /connections/ {
            allow 127.0.0.1;
            deny all;
            api /status/stream/metric_zones/active_connections/metrics/;
        }
    }
}

```

Ответ:

```

{
    "service_a": 42,
    "service_b": 17
}

```

Поддержка Prometheus

В Angie есть встроенный модуль Prometheus для отображения метрик в [формате Prometheus](#), который поддерживает произвольные метрики, включая метрики, собранные на потоковой стороне. Поскольку экспорт в формате Prometheus — HTTP-функция, директивы `prometheus_template` и `prometheus` задаются в блоке `http`, обращаясь к потоковой зоне метрики через единое дерево API `/status/`.

В качестве примера интеграции рассмотрим следующую конфигурацию:

```
stream {
    # Создание метрики "upload"
    metric_complex_zone upload:1m discard_key="other" {
        stats    histogram 64 256 1024 4096 16384 +Inf;
        sum      gauge;
        count    count;
        avg_size average exp;
    }

    upstream backend {
        server 127.0.0.1:15001;
    }

    server {
        listen 12345;

        # Обновление метрики по завершении сессии общим количеством
        # байт, полученных от клиента
        proxy_pass backend;
        metric upload angie=$bytes_received on=end;
    }
}

http {
    # Описание шаблона Prometheus для метрики "upload"
    prometheus_template upload_metric {
        'stats{le="$1"}' $p8s_value
        path=~~/stream/metric_zones/upload/metrics/angie/stats/(.+)
        type=histogram;

        'stats_sum' $p8s_value
        path=/stream/metric_zones/upload/metrics/angie/sum;
        'stats_count' $p8s_value
        path=/stream/metric_zones/upload/metrics/angie/count;

        'avg_size' $p8s_value
        path=/stream/metric_zones/upload/metrics/angie/avg_size;
    }

    server {
        listen 80;

        # Таргет для сбора метрик
        location /prometheus/upload_metric/ {
            prometheus upload_metric;
        }
    }
}
```

После пяти соединений с портом 12345, при которых были отправлены данные размером 16384, 64448, 64, 1028 и 1028 байт:

```
$ head -c 16384 /dev/urandom | nc 127.0.0.1 12345
$ head -c 64448 /dev/urandom | nc 127.0.0.1 12345
$ head -c 64 /dev/urandom | nc 127.0.0.1 12345
$ head -c 1028 /dev/urandom | nc 127.0.0.1 12345
```

```
$ head -c 1028 /dev/urandom | nc 127.0.0.1 12345
```

Значения метрики будут следующими:

```
{
  "stats": {
    "64": 1,
    "256": 1,
    "1024": 1,
    "4096": 3,
    "16384": 4,
    "+Inf": 5
  },
  "sum": 82952,
  "count": 5,
  "avg_size": 1077.9376
}
```

В формате Prometheus метрика доступна на `/prometheus/upload_metric/`:

```
# Angie Prometheus template "upload_metric"
# TYPE stats histogram
stats{le="64"} 1
stats{le="256"} 1
stats{le="1024"} 1
stats{le="4096"} 3
stats{le="16384"} 4
stats{le="+Inf"} 5
stats_sum 82952
stats_count 5
avg_size 1077.9376
```

MQTT Preread

Позволяет извлекать идентификатор клиента и имя пользователя из пакетов CONNECT протокола Message Queuing Telemetry Transport (MQTT) версий 3.1.1 и 5.0.

Пример конфигурации

Выбор сервера в группе по идентификатору клиента:

```
stream {
    mqtt_preread on;

    upstream mqtt {
        hash $mqtt_preread_clientid;
        # ...
    }
}
```

Директивы

mqtt_preread

Синтаксис	mqtt_preread on off;
По умолчанию	mqtt_preread off;
Контекст	stream, server

Управляет извлечением информации из пакета **CONNECT** на этапе предварительного чтения. Если параметр включен (**on**), то в контексте, где он задан, заполняются перечисленные ниже переменные.

Встроенные переменные

Подробное описание семантики значений см. в спецификации протокола MQTT версий 3.1.1 и 5.0.

`$mqtt_preread_clientid`

Уникальный идентификатор клиента.

`$mqtt_preread_username`

Необязательное имя пользователя.

Pass

Позволяет передавать принятое соединение напрямую на любой настроенный слушающий сокет в модулях HTTP, Stream или Mail.

Модуль допускает выборочную SSL-терминацию на основе SNI.

Пример конфигурации

После того, как модуль **stream** завершит обработку SSL/TLS, он передает соединение в модуль **http**:

```
stream {
    server {
        listen 8000 default_server;
        ssl_preread on;
        # ...
    }

    server {
        listen 8000;
        server_name foo.example.com;
        pass 127.0.0.1:8001; # to HTTP
    }

    server {
        listen 8000;
        server_name bar.example.com;
        # ...
    }
}
```

```
http {
    server {
        listen 8001 ssl;
        # ...

        location / {
            root html;
        }
    }
}
```

Директивы

pass

Синтаксис	<code>pass <i>адрес</i>;</code>
По умолчанию	—
Контекст	server

Эта директива задает адрес сервера, на который должно быть передано клиентское соединение. *Адрес* можно указать как IP-адрес и порт:

```
pass 127.0.0.1:12345;
```

Или как путь к UNIX-сокету:

```
pass unix:/tmp/stream.socket;
```

Также *адрес* можно задать с помощью переменных:

```
pass $upstream;
```

Proxy

Позволяет проксировать потоки данных по TCP, UDP и UNIX-сокетах.

Пример конфигурации

```
server {
    listen 127.0.0.1:12345;
    proxy_pass 127.0.0.1:8080;
}

server {
    listen 12345;
    proxy_connect_timeout 1s;
    proxy_timeout 1m;
    proxy_pass example.com:12345;
}
```



```
server {
    listen 53 udp reuseport;
    proxy_timeout 20s;
    proxy_pass dns.example.com:53;
}

server {
    listen [::1]:12345;
    proxy_pass unix:/tmp/stream.socket;
}
```

Директивы

proxy_bind

Синтаксис	<code>proxy_bind <i>адрес</i> [transparent] off;</code>
По умолчанию	—
Контекст	stream, server

Задаёт локальный IP-адрес, который будет использоваться в исходящих соединениях с проксируемым сервером. В значении параметра можно использовать переменные. Специальное значение `off` отменяет действие унаследованной с предыдущего уровня конфигурации директивы `proxy_bind`, позволяя системе самостоятельно выбирать локальный IP-адрес.

По умолчанию это значение вычисляется один раз при инициализации сессии, до выбора проксируемого сервера. В Angie PRO вычисление происходит после каждого выбора проксируемого сервера (в том числе при повторных попытках), поэтому адрес привязки может зависеть от выбранного сервера; например, с помощью переменной `$upstream_current_addr`. Если вычисленное значение пусто, привязка не выполняется.

Параметр `transparent` позволяет задать нелокальный IP-адрес, который будет использоваться в исходящих соединениях с проксируемым сервером, например, реальный IP-адрес клиента:

```
proxy_bind $remote_addr transparent;
```

Для работы параметра обычно требуется запустить рабочие процессы Angie с привилегиями суперпользователя. В Linux этого не требуется, так как если указан параметр `transparent`, то рабочие процессы наследуют capability `CAP_NET_RAW` из главного процесса.

Примечание

Необходимо настроить таблицу маршрутизации ядра для перехвата сетевого трафика с проксируемого сервера.

proxy_buffer_size

Синтаксис	<code>proxy_buffer_size размер;</code>
По умолчанию	<code>proxy_buffer_size 16k;</code>
Контекст	stream, server

Задаёт размер буфера, в который будут читаться данные, получаемые от проксируемого сервера. Также задаёт размер буфера, в который будут читаться данные, получаемые от клиента.

proxy_connect_timeout

Синтаксис	proxy_connect_timeout время;
По умолчанию	proxy_connect_timeout 60s;
Контекст	stream, server

Задаёт таймаут для установления соединения с проксируемым сервером.

proxy_connection_drop

Синтаксис	proxy_connection_drop время on off;
По умолчанию	proxy_connection_drop off;
Контекст	stream, server

Настраивает завершение всех сессий с проксируемым сервером, если он был удален из группы или помечен как постоянно недоступный в результате процесса `eresolve` или команды `API DELETE`.

Сессия завершается, когда обрабатывается следующее событие чтения или записи для клиента или проксируемого сервера.

Установка *времени* включает таймаут до завершения сессии; при выборе значения `on` сессии завершаются немедленно.

proxy_download_rate

Синтаксис	proxy_download_rate <i>скорость</i> ;
По умолчанию	proxy_download_rate 0;
Контекст	stream, server

Ограничивает скорость чтения данных от проксируемого сервера; *скорость* задается в байтах в секунду.

0	отключает ограничение скорости
---	--------------------------------

Примечание

Ограничение устанавливается на соединение, поэтому, если Angie одновременно откроет два соединения к проксируемому серверу, суммарная скорость будет вдвое выше заданного ограничения.

В значении параметра можно использовать переменные. Это может быть полезно в случаях, когда скорость нужно ограничивать в зависимости от какого-либо условия:

```
map $slow $rate {
    1    4k;
    2    8k;
}

proxy_download_rate $rate
```

proxy_half_close

Синтаксис	proxy_half_close on off;
По умолчанию	proxy_half_close off;
Контекст	stream, server

Разрешает или запрещает независимое закрытие каждой из сторон проксируемого соединения TCP ("TCP half-close"). Если разрешено, то проксирование по TCP будет продолжаться, пока обе стороны не закроют соединение.

proxy_interface

Синтаксис	proxy_interface имя;
По умолчанию	—
Контекст	stream, server

Задаёт сетевой интерфейс, который будет использоваться в исходящих соединениях с проксируемым сервером; сокет привязывается к нему параметром сокета SO_BINDTODEVICE.

proxy_next_upstream

Синтаксис	proxy_next_upstream on off;
По умолчанию	proxy_next_upstream on;
Контекст	stream, server

При невозможности установить соединение с проксируемым сервером определяет, будет ли клиентское соединение передано следующему серверу.

Передача соединения следующему серверу может быть ограничена по количеству попыток и по времени.

proxy_next_upstream_timeout

Синтаксис	proxy_next_upstream_timeout время;
По умолчанию	proxy_next_upstream_timeout 0;
Контекст	stream, server

Ограничивает время, в течение которого возможна передача запроса следующему серверу.

0	отключает это ограничение
---	---------------------------

proxy_next_upstream_tries

Синтаксис	proxy_next_upstream_tries число;
По умолчанию	proxy_next_upstream_tries 0;
Контекст	stream, server

Ограничивает число допустимых попыток для передачи запроса следующему серверу.

0	отключает это ограничение
---	---------------------------

proxy_pass

Синтаксис	<code>proxy_pass адрес;</code>
По умолчанию	—
Контекст	server

Задаёт адрес проксируемого сервера; **адрес** может быть указан в виде доменного имени или IP-адреса, за которым следует порт:

```
proxy_pass localhost:12345;
```

или в виде пути UNIX-сокета:

```
proxy_pass unix:/tmp/stream.socket;
```

Если доменному имени соответствует несколько адресов, то все они будут использоваться по очереди (round-robin). Кроме того, в качестве адреса можно указать группу серверов.

Если указанная группа объявлена без портов, и ее серверы заданы без порта, то порт добавляется к имени группы:

```
proxy_pass backend:8443;
```

Указывать порт после имени группы обязательно для группы без портов и недопустимо для группы, серверы которой заданы с портами; нарушение любого из этих правил приводит к ошибке конфигурации.

Адрес можно также задать с помощью переменных (вместе с портом, если он допустим):

```
proxy_pass $upstream;
```

В этом случае имя сервера ищется среди описанных групп серверов и если не найдено, то определяется с помощью *resolver*'а.

proxy_protocol

Синтаксис	<code>proxy_protocol on off;</code>
По умолчанию	<code>proxy_protocol off;</code>
Контекст	stream, server

Включает протокол **PROXY** для соединений с проксируемым сервером.

proxy_protocol_tlv

Синтаксис	<code>proxy_protocol_tlv имя значение;</code>
По умолчанию	—
Контекст	stream, server

Добавляет TLV в заголовок протокола PROXY версии 2, отправляемый на проксируемый сервер. *Значение* может содержать переменные. *Имя* может быть именем типа TLV или его числовым значением; в последнем случае значение задается в шестнадцатеричном виде и должно начинаться с *0x*. Для TLV SSL используйте префикс `ssl_`; специальное имя `ssl_verify` задает поле `verify` структуры SSL TLV. Директива используется только при значении 2 в *proxy_protocol_version*.

proxy_protocol_version

Синтаксис	<code>proxy_protocol_version 1 2;</code>
По умолчанию	<code>proxy_protocol_version 1;</code>
Контекст	<code>stream, server</code>

Задаёт версию протокола PROXY, используемую при соединениях с проксируемым сервером. Настройка действует при включенной директиве *proxy_protocol*. Версия 2 позволяет отправлять TLV, заданные директивой *proxy_protocol_tlv*.

proxy_requests

Синтаксис	<code>proxy_requests число;</code>
По умолчанию	<code>proxy_requests 0;</code>
Контекст	<code>stream, server</code>

Задаёт число датаграмм, полученных от клиента, по достижении которого удаляется привязка между клиентом и существующей UDP-сессией. После получения указанного количества датаграмм следующая датаграмма, полученная от того же клиента, начинает новую сессию. Сессия завершится после отправки всех принятых датаграмм на проксируемый сервер и получения указанного количества ответов или после таймаута.

proxy_responses

Синтаксис	<code>proxy_responses число;</code>
По умолчанию	—
Контекст	<code>stream, server</code>

Задаёт количество датаграмм, ожидаемых от проксируемого сервера в ответ на датаграмму клиента в случае, если используется протокол UDP. Задаваемое число служит подсказкой для завершения сессии. По умолчанию количество датаграмм не ограничено.

Если указано нулевое значение, то ответ не ожидается. Однако если ответ получен и сессия еще не завершилась, то ответ будет обработан.

proxy_socket_keepalive

Синтаксис	<code>proxy_socket_keepalive on off;</code>
По умолчанию	<code>proxy_socket_keepalive off;</code>
Контекст	<code>stream, server</code>

Конфигурирует поведение "TCP keepalive" для исходящих соединений к проксируемому серверу.

""	По умолчанию для сокета действуют настройки операционной системы.
on	для сокета включается параметр <i>SO_KEEPALIVE</i>

proxy_ssl

Синтаксис	<code>proxy_ssl on off;</code>
По умолчанию	<code>proxy_ssl off;</code>
Контекст	stream, server

Включает протоколы SSL/TLS для соединений с проксируемым сервером.

proxy_ssl_alpn

Синтаксис	<code>proxy_ssl_alpn <i>протокол</i> ...;</code>
По умолчанию	—
Контекст	stream, server

Задаёт список протоколов, объявляемых через расширение [ALPN](#) при установлении соединения с проксируемым сервером, например:

```
proxy_ssl_alpn h2 http/1.1;
```

Значение параметра может содержать переменные.

proxy_ssl_certificate

Синтаксис	<code>proxy_ssl_certificate <i>файл</i> [<i>файл</i>];</code>
По умолчанию	—
Контекст	stream, server

Задаёт файл с сертификатом в формате PEM для аутентификации на проксируемом сервере. В имени файла можно использовать переменные.

При включенном *proxy_ssl_nls* директива принимает два аргумента вместо одного:

```
server {
    proxy_ssl_nls on;

    proxy_ssl_certificate      sign.crt enc.crt;
    proxy_ssl_certificate_key  sign.key enc.key;

    proxy_ssl_ciphers "ECC-SM2-WITH-SM4-SM3:ECDHE-SM2-WITH-SM4-SM3:RSA";

    proxy_pass backend:12345;
}
```

proxy_ssl_certificate_cache

Синтаксис	<code>proxy_ssl_certificate_cache off;</code> <code>proxy_ssl_certificate_cache max=<i>N</i> [inactive=<i>time</i>] [valid=<i>time</i>];</code>
Значение по умолчанию	<code>proxy_ssl_certificate_cache off;</code>
Контекст	stream, server

Определяет кэш для хранения SSL-сертификатов и секретных ключей, заданных через переменные.

Директива поддерживает следующие параметры:

- **max** — устанавливает максимальное количество элементов в кэше. При переполнении кэша удаляются наименее недавно использованные (LRU) элементы.
- **inactive** — определяет время, после которого элемент будет удален, если к нему не было обращений в течение этого времени, даже если он еще находится в пределах своего периода **valid**; каждое обращение сбрасывает этот таймер. Значение по умолчанию — 10 секунд.
- **valid** — определяет время, в течение которого элемент кэша считается действительным и может использоваться повторно. Значение по умолчанию — 60 секунд. По истечении этого времени сертификаты перезагружаются или проходят повторную проверку.
- **off** — отключает кэш.

Пример:

```
proxy_ssl_certificate      $proxy_ssl_server_name.crt;
proxy_ssl_certificate_key  $proxy_ssl_server_name.key;
proxy_ssl_certificate_cache max=1000 inactive=20s valid=1m;
```

proxy_ssl_certificate_key

Синтаксис	<code>proxy_ssl_certificate_key <i>файл</i> [<i>файл</i>];</code>
По умолчанию	—
Контекст	stream, server

Задаёт файл с секретным ключом в формате PEM для аутентификации на проксируемом сервере.

Вместо файла можно указать значение `engine:`name`:id`, которое загружает ключ с указанным *id* из OpenSSL engine с заданным именем.

Вместо файла можно указать значение `store:scheme:id`, которое используется для загрузки ключа с указанным *id* и URI-схемой *scheme*, зарегистрированной в OpenSSL provider, например `pkcs11`.

В имени файла можно использовать переменные.

При включенном `proxy_ssl_ntls` директива принимает два аргумента вместо одного:

```
server {
    proxy_ssl_ntls on;

    proxy_ssl_certificate      sign.crt enc.crt;
    proxy_ssl_certificate_key  sign.key enc.key;

    proxy_ssl_ciphers "ECC-SM2-WITH-SM4-SM3:ECDHE-SM2-WITH-SM4-SM3:RSA";

    proxy_pass backend:12345;
}
```

proxy_ssl_ciphers

Синтаксис	<code>proxy_ssl_ciphers <i>шифры</i>;</code>
По умолчанию	<code>proxy_ssl_ciphers DEFAULT;</code>
Контекст	stream, server

Описывает разрешенные шифры для запросов к проксируемому серверу. Шифры задаются в формате, поддерживаемом библиотекой OpenSSL.

Список шифров зависит от установленной версии OpenSSL. Полный список можно посмотреть с помощью команды `openssl ciphers`.

Предупреждение

Директива `proxy_ssl_ciphers` *не* настраивает шифры для TLS 1.3 при использовании OpenSSL. Для настройки шифров TLS 1.3 в OpenSSL используйте директиву `proxy_ssl_conf_command`, добавленную для расширенной конфигурации SSL.

- В LibreSSL шифры TLS 1.3 *можно* настраивать с помощью `proxy_ssl_ciphers`.
- В BoringSSL шифры TLS 1.3 настроить невозможно.

proxy_ssl_conf_command

Синтаксис	<code>proxy_ssl_conf_command <i>имя значение</i>;</code>
По умолчанию	—
Контекст	stream, server

Задаёт произвольные конфигурационные команды OpenSSL при установлении соединения с проксируемым сервером.

Примечание

Директива поддерживается при использовании OpenSSL 1.0.2 и выше. Чтобы настроить шифры TLS 1.3 в OpenSSL, используйте команду `ciphersuites`.

На одном уровне может быть указано несколько директив `proxy_ssl_conf_command`. Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы `proxy_ssl_conf_command`.

Предупреждение

Следует учитывать, что изменение настроек OpenSSL напрямую может привести к неожиданному поведению.

proxy_ssl_crl

Синтаксис	<code>proxy_ssl_crl <i>файл</i>;</code>
По умолчанию	—
Контекст	stream, server

Указывает файл с отозванными сертификатами (CRL) в формате PEM, используемыми при проверке сертификата проксируемого сервера.

proxy_ssl_keylog_file

Синтаксис	<code>proxy_ssl_keylog_file <i>файл</i>;</code>
По умолчанию	—
Контекст	stream, server

Задаёт файл для журналирования SSL/TLS-ключей соединений с проксируемыми серверами в данном контексте, использующий формат `SSLKEYLOGFILE`, который применяется, например, в Wireshark для расшифровки перехваченного TLS-трафика.

Директиву не следует использовать с LibreSSL.

Предупреждение

Файл содержит конфиденциальный ключевой материал TLS, с помощью которого можно расшифровать перехваченный трафик соответствующих соединений. Используйте директиву только для отладки или анализа протокола, ограничьте доступ к файлу и удалите его, когда он больше не нужен.

proxy_ssl_name

Синтаксис	<code>proxy_ssl_name <i>имя</i>;</code>
По умолчанию	<code>proxy_ssl_name <i>хост</i> из proxy_pass;</code>
Контекст	stream, server

Позволяет переопределить имя сервера, используемое при проверке сертификата проксируемого сервера, а также для передачи его через SNI при установлении соединения с проксируемым сервером. Имя сервера можно также задать с помощью переменных.

По умолчанию используется имя хоста из адреса, заданного директивой `proxy_pass`.

proxy_ssl_ntls

Синтаксис	<code>proxy_ssl_ntls on off;</code>
По умолчанию	<code>proxy_ssl_ntls off;</code>
Контекст	stream, server

Включает клиентскую поддержку NTLS при использовании TLS библиотеки `Tong suo`.

```
server {
    proxy_ssl_ntls on;

    proxy_ssl_certificate      sign.crt enc.crt;
    proxy_ssl_certificate_key  sign.key enc.key;

    proxy_ssl_ciphers "ECC-SM2-WITH-SM4-SM3:ECDHE-SM2-WITH-SM4-SM3:RSA";

    proxy_pass backend:12345;
}
```

proxy_ssl_password_file

Синтаксис	proxy_ssl_password_file <i>файл</i> ;
По умолчанию	—
Контекст	stream, server

Задаёт файл с паролями от секретных ключей, где каждый пароль указан на отдельной строке. Пароли применяются по очереди в момент загрузки ключа.

proxy_ssl_protocols

Синтаксис	proxy_ssl_protocols [SSLv2] [SSLv3] [TLSv1] [TLSv1.1] [TLSv1.2] [TLSv1.3];
По умолчанию	proxy_ssl_protocols TLSv1.2 TLSv1.3;
Контекст	stream, server

Разрешает указанные протоколы для соединений с проксируемым сервером.

proxy_ssl_server_name

Синтаксис	proxy_ssl_server_name on off;
По умолчанию	proxy_ssl_server_name off;
Контекст	stream, server

Разрешает или запрещает передачу имени сервера, заданного директивой *proxy_ssl_name*, через расширение [Server Name Indication](#) протокола TLS (SNI, [RFC 6066](#)) при установлении соединения с проксируемым сервером.

proxy_ssl_session_reuse

Синтаксис	proxy_ssl_session_reuse on off;
По умолчанию	proxy_ssl_session_reuse on;
Контекст	stream, server

Определяет, использовать ли повторно SSL-сессии при работе с проксируемым сервером. Если в логах появляются ошибки "*SSL3_GET_FINISHED:digest check failed*", то можно попробовать включить повторное использование сессий.

proxy_ssl_trusted_certificate

Синтаксис	proxy_ssl_trusted_certificate <i>файл</i> ;
По умолчанию	—
Контекст	stream, server

Задаёт файл с доверенными сертификатами СА в формате PEM, используемыми при проверке сертификата проксируемого HTTPS-сервера.

proxy_ssl_verify

Синтаксис	proxy_ssl_verify on off;
По умолчанию	proxy_ssl_verify off;
Контекст	stream, server

Разрешает или запрещает проверку сертификата проксируемого сервера.

proxy_ssl_verify_depth

Синтаксис	proxy_ssl_verify_depth <i>число</i> ;
По умолчанию	proxy_ssl_verify_depth 1;
Контекст	stream, server

Устанавливает глубину проверки в цепочке сертификатов проксируемого сервера.

proxy_tcp_congestion

Синтаксис	proxy_tcp_congestion <i>алгоритм</i> ;
По умолчанию	—
Контекст	stream, server

Задаёт алгоритм управления перегрузкой TCP (параметр сокета TCP_CONGESTION) для исходящих соединений с проксируемым сервером; к соединениям через UNIX-сокеты параметр не применяется.

Имя алгоритма не проверяется на этапе конфигурации: если алгоритм неизвестен или недоступен, при установке исходящего соединения в лог записывается сообщение уровня **alert**, а для соединения действует системный алгоритм по умолчанию.

proxy_timeout

Синтаксис	proxy_timeout время;
По умолчанию	proxy_timeout 10m;
Контекст	stream, server

Задаёт таймаут между двумя идущими подряд операциями чтения или записи на клиентском соединении или соединении с проксируемым сервером. Если по истечении этого времени данные не передавались, соединение закрывается.

proxy_upload_rate

Синтаксис	proxy_upload_rate <i>скорость</i> ;
По умолчанию	proxy_upload_rate 0;
Контекст	stream, server

Ограничивает скорость чтения данных от клиента. Скорость задается в байтах в секунду.

0	отключает ограничение скорости
---	--------------------------------

Примечание

Ограничение устанавливается на соединение, поэтому, если клиент одновременно откроет два соединения, суммарная скорость будет вдвое выше заданного ограничения.

В значении параметра можно использовать переменные. Это может быть полезно в случаях, когда скорость нужно ограничивать в зависимости от какого-либо условия:

```
map $slow $rate {
    1      4k;
    2      8k;
}

proxy_upload_rate $rate;
```

RDP Preread

При использовании протокола RDP позволяет извлекать cookie, которые используются для идентификации и управления сессиями, до момента принятия решения о балансировке.

Пример конфигурации

Привязка к выдавшему cookie серверу

Конфигурация использует режим `learn` директивы *sticky*:

```
stream {

    rdp_preread on;

    upstream rdp {

        server 127.0.0.1:3390 sid=a;
        server 127.0.0.1:3391 sid=b;

        sticky learn lookup=$rdp_cookie create=$rdp_cookie zone=sessions:1m;
    }
}
```

Директивы

rdp_preread

Синтаксис	<code>rdp_preread on off;</code>
По умолчанию	<code>rdp_preread off;</code>
Контекст	<code>stream, server</code>

Управляет извлечением информации из cookie протокола RDP на этапе предварительного чтения. Если параметр включен (`on`), то в контексте, где он задан, заполняются перечисленные ниже переменные.

Встроенные переменные

Семантика значений в составе cookie зависит от версии протокола RDP.

`$rdp_cookie`

Значение cookie целиком.

`$rdp_cookie_<имя>`

Значение поля cookie с заданным именем.

RealIP

Позволяет менять адрес и порт клиента на переданные в заголовке протокола PROXY. Протокол PROXY должен быть предварительно включен при помощи установки параметра `proxy_protocol` в директиве `listen`.

Пример конфигурации

```
listen 12345 proxy_protocol;

set_real_ip_from 192.168.1.0/24;
set_real_ip_from 192.168.2.1;
set_real_ip_from 2001:0db8::/32;
```

Директивы

set_real_ip_from

Синтаксис	<code>set_real_ip_from адрес CIDR unix;;</code>
По умолчанию	<code>—</code>
Контекст	<code>stream, server</code>

Задаёт доверенные адреса, которые передают верный адрес для замены. Если указано специальное значение `unix:`, доверенными будут считаться все UNIX-сокеты.

Встроенные переменные

`$realip_remote_addr`

хранит исходный адрес клиента

`$realip_remote_port`

хранит исходный порт клиента

Return

Позволяет отправить заданное значение клиенту и после этого закрыть соединение.

Пример конфигурации

```
server {
    listen 12345;
    return $time_iso8601;
}
```

Директивы

return

Синтаксис	<code>return <i>значение</i>;</code>
По умолчанию	—
Контекст	server

Задаёт значение, отправляемое клиенту. В качестве значения можно использовать переменные.

Set

Позволяет устанавливать значение переменной.

Пример конфигурации

```
server {
    listen 12345;
    set $true 1;
}
```

Директивы

set

Синтаксис	<code>set <i>\$переменная</i> <i>значение</i>;</code>
По умолчанию	—
Контекст	server

Устанавливает значение указанной переменной. В качестве значения можно использовать переменные.

Split Clients

Модуль генерирует переменные для А/В-тестирования, канареечных релизов и других сценариев, которые направляют определенный процент клиентов на один сервер или конфигурацию, а остальных — куда-то еще.

Пример конфигурации

```
stream {
    # ...
    split_clients "${remote_addr}AAA" $upstream {
        0.5%          feature_test1;
        2.0%          feature_test2;
        *             production;
    }

    server {
        # ...
        proxy_pass $upstream;
    }
}
```

Директивы

split_clients

Синтаксис	<code>split_clients строка \$переменная { ... }</code>
По умолчанию	—
Контекст	stream

Создает *\$переменную*, хэшируя *строку*; переменные в *строке* подставляются, результат хэшируется, затем по значению хэша выбирается строковое значение *\$переменной*.

Функция хэширования использует [MurmurHash2](#) (32 бит), и весь диапазон ее значений (с 0 по 4294967295) сопоставляется с корзинами в порядке появления; процентные величины определяют размер корзин. В конце может стоять метасимвол (*); хэши, не попавшие в другие корзины, сопоставляются с приданным ему значением.

Пример:

```
split_clients "${remote_addr}AAA" $variant {
    0.5%          .one;
    2.0%          .two;
    *             "";
}
```

Здесь после подстановки в строке `$remote_addrAAA` значения хэша распределяются следующим образом:

- значения от 0 до 21474835 (0,5%) дают `.one`;
- значения от 21474836 до 107374180 (2%) дают `.two`;
- значения от 107374181 до 4294967295 (все остальные) дают `"` (пустую строку).

SSL

Обеспечивает необходимую поддержку для работы прокси-сервера по протоколу SSL/TLS.

Примечание

Для этого модуля нужна библиотека OpenSSL.

Пример конфигурации

Для уменьшения загрузки процессора рекомендуется

- установить число рабочих процессов равным числу процессоров,
- включить разделяемый кэш сессий,
- выключить встроенный кэш сессий
- и, возможно, увеличить время жизни сессии (по умолчанию 5 минут):

```
worker_processes auto;

stream {

    #...

    server {
        listen          12345 ssl;

        ssl_protocols   TLSv1.2 TLSv1.3;
        ssl_ciphers      AES128-SHA:AES256-SHA:RC4-SHA:DES-CBC3-SHA:RC4-MD5;
        ssl_certificate  /usr/local/angie/conf/cert.pem;
        ssl_certificate_key /usr/local/angie/conf/cert.key;
        ssl_session_cache shared:SSL:10m;
        ssl_session_timeout 10m;

        #    ...
    }
}
```

Директивы

ssl_alpn

Синтаксис	<code>ssl_alpn <i>протокол ...</i>;</code>
По умолчанию	—
Контекст	stream, server

Задаёт список поддерживаемых протоколов [ALPN](#). Один из протоколов должен быть согласован, если клиент использует ALPN:

```
map $ssl_alpn_protocol $proxy {
    h2          127.0.0.1:8001;
    http/1.1    127.0.0.1:8002;
}

server {
    listen      12346;
```



```
proxy_pass $proxy;
ssl_alpn h2 http/1.1;
}
```

ssl_certificate

Синтаксис	ssl_certificate <i>файл</i> ;
По умолчанию	—
Контекст	stream, server

Указывает файл с сертификатом в формате PEM для данного сервера. Если вместе с основным сертификатом нужно указать промежуточные, то они должны находиться в этом же файле в следующем порядке — сначала основной сертификат, а затем промежуточные. В этом же файле может находиться секретный ключ в формате PEM.

Директива может быть указана несколько раз для загрузки сертификатов разных типов, например RSA и ECDSA:

```
server {
    listen          12345 ssl;

    ssl_certificate  example.com.rsa.crt;
    ssl_certificate_key example.com.rsa.key;

    ssl_certificate  example.com.ecdsa.crt;
    ssl_certificate_key example.com.ecdsa.key;

    # ...
}
```

Возможность задавать отдельные цепочки сертификатов для разных сертификатов есть только в OpenSSL 1.0.2 и выше. Для более старых версий следует указывать только одну цепочку сертификатов.

Примечание

В имени файла можно использовать переменные при использовании OpenSSL 1.0.2 и выше:

```
ssl_certificate $ssl_server_name.crt;
ssl_certificate_key $ssl_server_name.key;
```

При использовании переменных сертификат загружается при каждой операции SSL-рукопожатия, что может отрицательно влиять на производительность.

Вместо **файла** можно указать значение "data:*\$переменная*", при котором сертификат загружается из переменной без использования промежуточных файлов.

Ненадлежащее использование подобного синтаксиса может быть небезопасно, например данные секретного ключа могут попасть в лог ошибок.

ssl_certificate_cache

Синтаксис	<code>ssl_certificate_cache off;</code> <code>ssl_certificate_cache max=<i>N</i> [inactive=<i>time</i>] [valid=<i>time</i>];</code>
Значение по умолчанию	<code>ssl_certificate_cache off;</code>
Контекст	stream, server

Определяет кэш для хранения SSL-сертификатов и секретных ключей, заданных через переменные.

Директива поддерживает следующие параметры:

- **max** — устанавливает максимальное количество элементов в кэше. При переполнении кэша удаляются наименее недавно использованные (LRU) элементы.
- **inactive** — определяет время, после которого элемент будет удален, если к нему не было обращений в течение этого времени, даже если он еще находится в пределах своего периода **valid**; каждое обращение сбрасывает этот таймер. Значение по умолчанию — 10 секунд.
- **valid** — определяет время, в течение которого элемент кэша считается действительным и может использоваться повторно. Значение по умолчанию — 60 секунд. По истечении этого времени сертификаты перезагружаются или проходят повторную проверку.
- **off** — отключает кэш.

Пример:

```
ssl_certificate      $ssl_server_name.crt;
ssl_certificate_key  $ssl_server_name.key;
ssl_certificate_cache max=1000 inactive=20s valid=1m;
```

ssl_certificate_compression

Синтаксис	<code>ssl_certificate_compression on off;</code>
По умолчанию	<code>ssl_certificate_compression off;</code>
Контекст	stream, server

Разрешает сжатие TLS 1.3 сертификатов сервера.

Примечание

Директива поддерживается при использовании OpenSSL версии 3.2 и выше; список поддерживаемых алгоритмов сжатия предоставляется библиотекой.

Примечание

Директива поддерживается при использовании BoringSSL; список поддерживаемых алгоритмов сжатия включает `zlib`.

Если включен `ssl_stapling`, сжатие сертификатов отключается.

ssl_certificate_key

Синтаксис	<code>ssl_certificate_key <i>файл</i>;</code>
По умолчанию	—
Контекст	stream, server

Указывает файл с секретным ключом в формате PEM для данного виртуального сервера.

Примечание

В имени файла можно использовать переменные при использовании OpenSSL 1.0.2 и выше.

Вместо **файла** можно указать значение `"engine:имя:id"`, которое загружает ключ с указанным *id* из OpenSSL engine с заданным именем.

Вместо **файла** можно указать значение `"store:scheme:id"`, которое используется для загрузки ключа с указанным *id* и URI-схемой *scheme*, зарегистрированной в OpenSSL provider, например `pkcs11`.

Вместо **файла** также можно указать значение `"data:$переменная"`, при котором секретный ключ загружается из переменной без использования промежуточных файлов. При этом следует учитывать, что ненадлежащее использование подобного синтаксиса может быть небезопасно, например данные секретного ключа могут попасть в лог ошибок.

ssl_ciphers

Синтаксис	<code>ssl_ciphers <i>шифры</i>;</code>
По умолчанию	<code>ssl_ciphers HIGH:!aNULL:!MD5;</code>
Контекст	stream, server

Описывает разрешенные шифры. Шифры задаются в формате, поддерживаемом библиотекой OpenSSL, например:

```
ssl_ciphers ALL:!aNULL:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP;
```

Список шифров зависит от установленной версии OpenSSL. Полный список можно посмотреть с помощью команды `openssl ciphers`.

Предупреждение

Директива `ssl_ciphers` не настраивает шифры для TLS 1.3 при использовании OpenSSL. Для настройки шифров TLS 1.3 в OpenSSL используйте директиву `ssl_conf_command`, добавленную для расширенной конфигурации SSL.

- В LibreSSL шифры TLS 1.3 можно настраивать с помощью `ssl_ciphers`.
- В BoringSSL шифры TLS 1.3 настроить невозможно.

ssl_client_certificate

Синтаксис	<code>ssl_client_certificate <i>файл</i>;</code>
По умолчанию	—
Контекст	stream, server

Задаёт файл с доверенными сертификатами СА в формате PEM, которые используются для проверки клиентских сертификатов и ответов OCSP, если включен [ssl_stapling](#).

Список сертификатов будет отправляться клиентам. Если это нежелательно, можно воспользоваться директивой [ssl_trusted_certificate](#).

ssl_conf_command

Синтаксис	<code>ssl_conf_command <i>имя значение</i>;</code>
По умолчанию	—
Контекст	stream, server

Задаёт произвольные конфигурационные команды OpenSSL.

Примечание

Директива поддерживается при использовании OpenSSL 1.0.2 и выше. Чтобы настроить шифры TLS 1.3 в OpenSSL, используйте команду `ciphersuites`.

На одном уровне может быть указано несколько директив `ssl_conf_command`:

```
ssl_conf_command Options PrioritizeChaCha;
ssl_conf_command Ciphersuites TLS_CHACHA20_POLY1305_SHA256;
```

Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы `ssl_conf_command`.

Предупреждение

Изменение настроек OpenSSL напрямую может привести к неожиданному поведению.

ssl_crl

Синтаксис	<code>ssl_crl <i>файл</i>;</code>
По умолчанию	—
Контекст	stream, server

Указывает файл с отозванными сертификатами (CRL) в формате PEM, используемыми для проверки клиентских сертификатов.

ssl_dhparam

Синтаксис	<code>ssl_dhparam <i>файл</i>;</code>
По умолчанию	—
Контекст	stream, server

Указывает файл с параметрами для DHE-шифров.

Предупреждение

По умолчанию параметры не заданы, и соответственно DHE-шифры не будут использоваться.

ssl_early_data

Синтаксис	<code>ssl_early_data on off;</code>
По умолчанию	<code>ssl_early_data off;</code>
Контекст	stream, server

Разрешает или запрещает TLS 1.3 [early data](#).

Примечание

Директива поддерживается при использовании OpenSSL 1.1.1 и выше или [BoringSSL](#).

ssl_ecdh_curve

Синтаксис	<code>ssl_ecdh_curve <i>кривая</i>;</code>
По умолчанию	<code>ssl_ecdh_curve auto;</code>
Контекст	stream, server

Задаёт кривую для ECDHE-шифров.

Примечание

При использовании OpenSSL 1.0.2 и выше можно указывать несколько кривых, например:

```
ssl_ecdh_curve prime256v1:secp384r1;
```

Специальное значение `auto` соответствует встроенному в библиотеку OpenSSL списку кривых для OpenSSL 1.0.2 и выше, или `prime256v1` для более старых версий.

Примечание

При использовании OpenSSL 1.0.2 и выше директива задаёт список кривых, поддерживаемых сервером. Поэтому для работы ECDSA-сертификатов важно, чтобы список включал кривые, используемые в сертификатах.

ssl_encrypted_hello_key

Синтаксис	<code>ssl_encrypted_hello_key <i>файл</i>;</code>
По умолчанию	—
Контекст	stream, server

Указывает файл с секретным ключом и конфигурацией расширения TLS Encrypted Client Hello (ECH), которое скрывает имя сервера, запрашиваемое клиентом, от наблюдателей в сети. Переменная `ssl_encrypted_hello` показывает, использовалось ли расширение при рукопожатии.

Директива может быть указана несколько раз, что позволяет настроить ротацию ключей: для расшифровки применяются все ключи, но клиентам для повторных попыток предлагается конфигурация только первого из них.

```
ssl_encrypted_hello_key current.ech;
ssl_encrypted_hello_key previous.ech;
```

Файл содержит секретный ключ, за которым следует ECHConfigList; оба в формате PEM. В ветке OpenSSL с поддержкой ECH такой файл создается командой:

```
$ openssl ech -public_name public.example.com -out example.ech
```

В BoringSSL команда `bssl generate-ech` записывает ключ и ECHConfigList в отдельные двоичные файлы, и напрямую использовать их нельзя. Из них нужно собрать один файл PEM: необработанный ключ обернуть в структуру PKCS#8 и закодировать в base64 в виде блока PRIVATE KEY, а следом поместить ECHConfigList, а не отдельный ECHConfig, закодированный в base64, в виде блока ECHCONFIG. Сборки с BoringSSL поддерживают только ключи X25519. Сам формат не зависит от библиотеки TLS, поэтому файл, созданный приведенной выше командой, подходит и для сборки с BoringSSL.

Angie расшифровывает ClientHello до того, как узнает имя сервера, запрошенное через SNI, поэтому при нескольких виртуальных серверах ключи необходимо указывать для сервера, назначенного по умолчанию для соответствующего порта.

Примечание

Поддержка ECH обеспечивается библиотекой TLS, поэтому Angie необходимо собрать с библиотекой, которая ее реализует: с веткой OpenSSL с поддержкой ECH, которая подключается так:

```
$ ./configure --with-openssl=../openssl-ech
```

либо с BoringSSL, который подключается иначе. В противном случае директива игнорируется с предупреждением в журнале ошибок.

ssl_handshake_timeout

Синтаксис	<code>ssl_handshake_timeout время;</code>
По умолчанию	<code>ssl_handshake_timeout 60s;</code>
Контекст	stream, server

Задаёт таймаут для завершения операции SSL-рукопожатия.

ssl_keylog_file

Синтаксис	<code>ssl_keylog_file <i>файл</i>;</code>
По умолчанию	—
Контекст	stream, server

Задаёт файл для журналирования SSL/TLS-ключей соединений в данном контексте, использующий формат `SSLKEYLOGFILE`, который применяется, например, в Wireshark для расшифровки перехваченного TLS-трафика.

Директиву не следует использовать с LibreSSL.

Предупреждение

Файл содержит конфиденциальный ключевой материал TLS, с помощью которого можно расшифровать перехваченный трафик соответствующих соединений. Используйте директиву только для отладки или анализа протокола, ограничьте доступ к файлу и удалите его, когда он больше не нужен.

ssl_ntls

Синтаксис	<code>ssl_ntls on off;</code>
По умолчанию	<code>ssl_ntls off;</code>
Контекст	stream, server

Включает серверную поддержку NTLS при использовании TLS библиотеки [Tongtsuo](#)

```
listen ... ssl;
ssl_ntls on;
```

ssl_ocsp

Синтаксис	<code>ssl_ocsp on off leaf;</code>
По умолчанию	<code>ssl_ocsp off;</code>
Контекст	http, server

Включает проверку OCSP для цепочки клиентских сертификатов. Параметр `leaf` включает проверку только клиентского сертификата.

Для работы проверки OCSP необходимо дополнительно установить значение директивы `ssl_verify_client` в `on` или `optional`.

Для преобразования имени хоста OCSP-респондера в адрес необходимо дополнительно задать директиву `resolver`.

Пример:

```
ssl_verify_client on;
ssl_ocsp on;
resolver 127.0.0.53;
```

ssl_ocsp_cache

Синтаксис	<code>ssl_ocsp_cache off [shared:имя:размер];</code>
По умолчанию	<code>ssl_ocsp_cache off;</code>
Контекст	http, server

Задаёт имя и размер кэша, который хранит статус клиентских сертификатов для проверки OCSP-ответов. Кэш разделяется между всеми рабочими процессами. Кэш с одинаковым названием может использоваться в нескольких виртуальных серверах.

Параметр `off` запрещает использование кэша.

ssl_ocsp_responder

Синтаксис	<code>ssl_ocsp_responder uri;</code>
По умолчанию	—
Контекст	http, server

Переопределяет URI OCSP-респондера, указанный в расширении сертификата "Authority Information Access" для проверки клиентских сертификатов.

Поддерживаются только OCSP-респондеры на основе `http://`:

```
ssl_ocsp_responder http://ocsp.example.com/;
```

ssl_password_file

Синтаксис	<code>ssl_password_file файл;</code>
По умолчанию	—
Контекст	stream, server

Задаёт файл с паролями от секретных ключей, где каждый пароль указан на отдельной строке. Пароли применяются по очереди в момент загрузки ключа.

Пример:

```
stream {
    ssl_password_file /etc/keys/global.pass;
    ...

    server {
        listen 127.0.0.1:12345;
        ssl_certificate_key /etc/keys/first.key;
    }

    server {
        listen 127.0.0.1:12346;

        # вместо файла можно указать именованный канал
        ssl_password_file /etc/keys/fifo;
        ssl_certificate_key /etc/keys/second.key;
    }
}
```



```
}  
}
```

ssl_prefer_server_ciphers

Синтаксис	ssl_prefer_server_ciphers on off;
По умолчанию	ssl_prefer_server_ciphers off;
Контекст	stream, server

При использовании протоколов SSLv3 и TLS устанавливает приоритет серверных шифров над клиентскими.

ssl_protocols

Синтаксис	ssl_protocols [SSLv2] [SSLv3] [TLSv1] [TLSv1.1] [TLSv1.2] [TLSv1.3];
По умолчанию	ssl_protocols TLSv1.2 TLSv1.3;
Контекст	stream, server

Разрешает указанные протоколы.

Примечание

Параметры TLSv1.1 и TLSv1.2 работают только при использовании OpenSSL 1.0.1 и выше.
Параметр TLSv1.3 работает только при использовании OpenSSL 1.1.1 и выше.

ssl_session_cache

Синтаксис	ssl_session_cache off none [builtin[:размер]] [shared:название:размер];
По умолчанию	ssl_session_cache none;
Контекст	stream, server

Задаёт тип и размеры кэшей для хранения параметров сессий. Тип кэша может быть следующим:

off	жесткое запрещение использования кэша сессий: Angie явно сообщает клиенту, что сессии не могут использоваться повторно.
none	мягкое запрещение использования кэша сессий: Angie сообщает клиенту, что сессии могут использоваться повторно, но на самом деле не хранит параметры сессии в кэше.
builtin	встроенный в OpenSSL кэш, используется в рамках только одного рабочего процесса. Размер кэша задается в сессиях. Если размер не задан, то он равен 20480 сессиям. Использование встроенного кэша может вести к фрагментации памяти.
shared	кэш, разделяемый между всеми рабочими процессами. Размер кэша задается в байтах, в 1 мегабайт может поместиться около 4000 сессий. У каждого разделяемого кэша должно быть произвольное название. Кэш с одинаковым названием может использоваться в нескольких серверах. Также он используется для автоматического создания, хранения и периодического обновления ключей сессионных билетов TLS, если они не указаны явно с помощью директивы <code>ssl_session_ticket_key</code> .

Можно использовать одновременно оба типа кэша, например:

```
ssl_session_cache builtin:1000 shared:SSL:10m;
```

однако использование только разделяемого кэша без встроенного должно быть более эффективным.

ssl_session_ticket_key

Синтаксис	<code>ssl_session_ticket_key <i>файл</i>;</code>
По умолчанию	—
Контекст	stream, server

Задаёт файл с секретным ключом, применяемым при шифровании и расшифровке сессионных билетов TLS. Директива необходима, если один и тот же ключ нужно использовать на нескольких серверах. По умолчанию используется случайно сгенерированный ключ.

Если указано несколько ключей, то только первый ключ используется для шифрования сессионных билетов TLS. Это позволяет настроить ротацию ключей, например:

```
ssl_session_ticket_key current.key;
ssl_session_ticket_key previous.key;
```

Файл должен содержать 80 или 48 байт случайных данных и может быть создан следующей командой:

```
$ openssl rand 80 > ticket.key
```

В зависимости от размера файла для шифрования будет использоваться либо AES256 (для 80-байтных ключей), либо AES128 (для 48-байтных ключей).

ssl_session_tickets

Синтаксис	<code>ssl_session_tickets on off;</code>
По умолчанию	<code>ssl_session_tickets on;</code>
Контекст	stream, server

Разрешает или запрещает возобновление сессий при помощи [сессионных билетов TLS](#).

ssl_session_timeout

Синтаксис	ssl_session_timeout время;
По умолчанию	ssl_session_timeout 5m;
Контекст	stream, server

Задаёт время, в течение которого клиент может повторно использовать параметры сессии.

ssl_stapling

Синтаксис	ssl_stapling on off;
По умолчанию	ssl_stapling off;
Контекст	http, server

Разрешает или запрещает [прикрепление OCSP-ответов](#) сервером. Пример:

```
ssl_stapling on;
resolver 127.0.0.53;
```

Для работы OCSP-прикрепления должен быть известен сертификат издателя сертификата сервера. Если в заданном директивой [ssl_certificate](#) файле не содержится промежуточных сертификатов, то сертификат издателя сертификата сервера следует поместить в файл, заданный директивой [ssl_trusted_certificate](#).

Предупреждение

Для преобразования имени хоста OCSP-респондера в адрес необходимо дополнительно задать директиву [resolver](#).

ssl_stapling_file

Синтаксис	ssl_stapling_file <i>файл</i> ;
По умолчанию	—
Контекст	http, server

Если значение задано, то вместо опроса OCSP-респондера, указанного в сертификате сервера, ответ берётся из указанного файла.

Ответ должен быть в формате DER и может быть сгенерирован командой `openssl ocsp`.

ssl_stapling_responder

Синтаксис	ssl_stapling_responder uri;
По умолчанию	—
Контекст	http, server

Переопределяет URI OCSP-респондера, указанный в расширении сертификата "Authority Information Access".

Поддерживаются только OCSP-респондеры на основе `http://`:

```
ssl_stapling_responder http://ocsp.example.com/;
```

ssl_stapling_verify

Синтаксис	<code>ssl_stapling_verify on off;</code>
По умолчанию	<code>ssl_stapling_verify off;</code>
Контекст	<code>http, server</code>

Разрешает или запрещает проверку ответов OCSP сервером.

Для работоспособности проверки сертификат издателя сертификата сервера, корневой сертификат и все промежуточные сертификаты должны быть указаны как доверенные с помощью директивы `ssl_trusted_certificate`.

ssl_trusted_certificate

Синтаксис	<code>ssl_trusted_certificate <i>файл</i>;</code>
По умолчанию	—
Контекст	<code>stream, server</code>

Задаёт файл с доверенными сертификатами СА в формате PEM, которые используются для проверки клиентских сертификатов.

В отличие от `ssl_client_certificate`, список этих сертификатов не будет отправляться клиентам.

ssl_verify_client

Синтаксис	<code>ssl_verify_client on off optional optional_no_ca;</code>
По умолчанию	<code>ssl_verify_client off;</code>
Контекст	<code>stream, server</code>

Разрешает проверку клиентских сертификатов. Результат проверки доступен через переменную `$ssl_client_verify`. Если при проверке клиентского сертификата произошла ошибка или клиент не предоставил требуемый сертификат, соединение закрывается.

optional	запрашивает клиентский сертификат, и если сертификат был предоставлен, проверяет его
optional_no_ca	запрашивает сертификат клиента, но не требует, чтобы он был подписан доверенным сертификатом СА. Это предназначено для случаев, когда фактическая проверка сертификата осуществляется внешним по отношению к Angie сервисом.

ssl_verify_depth

Синтаксис	<code>ssl_verify_depth число;</code>
По умолчанию	<code>ssl_verify_depth 1;</code>
Контекст	<code>stream, server</code>

Устанавливает глубину проверки в цепочке клиентских сертификатов.

Встроенные переменные

Модуль `stream_ssl` поддерживает встроенные переменные:

`$ssl_alpn_protocol`

возвращает протокол, выбранный при помощи ALPN во время SSL-рукопожатия, либо пустую строку.

`$ssl_cipher`

возвращает название используемого шифра для установленного SSL-соединения.

`$ssl_ciphers`

возвращает список шифров, поддерживаемых клиентом. Известные шифры указаны по имени, неизвестные указаны в шестнадцатеричном виде, например:

`AES128-SHA:AES256-SHA:0x00ff`

i Примечание

Переменная полностью поддерживается при использовании OpenSSL версии 1.0.2 и выше. При использовании более старых версий переменная доступна только для новых сессий и может содержать только известные шифры.

`$ssl_client_cert`

возвращает клиентский сертификат для установленного SSL-соединения в формате PEM перед каждой строкой которого, кроме первой, вставляется символ табуляции.

`$ssl_client_fingerprint`

возвращает SHA1-отпечаток клиентского сертификата для установленного SSL-соединения.

`$ssl_client_i_dn`

возвращает строку "issuer DN" клиентского сертификата для установленного SSL-соединения согласно [RFC 2253](#).

`$ssl_client_raw_cert`

возвращает клиентский сертификат для установленного SSL-соединения в формате PEM.

`$ssl_client_s_dn`

возвращает строку "subject DN" клиентского сертификата для установленного SSL-соединения согласно RFC 2253.

`$ssl_client_serial`

возвращает серийный номер клиентского сертификата для установленного SSL-соединения.

`$ssl_client_sigalg`

возвращает алгоритм подписи для сертификата клиента в установленном SSL-соединении.

Примечание

Переменная поддерживается только при использовании OpenSSL версии 3.5 и выше. При использовании более старых версий значением переменной будет пустая строка.

Примечание

Переменная доступна только для новых сессий.

`$ssl_client_v_end`

возвращает дату окончания срока действия клиентского сертификата.

`$ssl_client_v_remain`

возвращает число дней, оставшихся до истечения срока действия клиентского сертификата.

`$ssl_client_v_start`

возвращает дату начала срока действия клиентского сертификата.

`$ssl_client_verify`

возвращает результат проверки клиентского сертификата: SUCCESS, FAILED:reason и, если сертификат не был предоставлен, NONE.

`$ssl_curve`

возвращает согласованную кривую, использованную для обмена ключами во время SSL-рукопожатия. Известные кривые указаны по имени, неизвестные указаны в шестнадцатеричном виде, например:

prime256v1

Примечание

Переменная поддерживается при использовании OpenSSL версии 3.0 и выше. При использовании более старых версий значением переменной будет пустая строка.

`$ssl_curves`

возвращает список кривых, поддерживаемых клиентом. Известные кривые указаны по имени, неизвестные указаны в шестнадцатеричном виде, например:

```
0x001d:prime256v1:secp521r1:secp384r1
```

Примечание

Переменная поддерживается при использовании OpenSSL версии 1.0.2 и выше. При использовании более старых версий значением переменной будет пустая строка.

Переменная доступна только для новых сессий.

`$ssl_early_data`

возвращает 1, если используется TLS 1.3 early data и операция SSL-рукопожатия не завершена, иначе "".

`$ssl_encrypted_hello`

возвращает 1, если ClientHello был расшифрован с помощью расширения TLS Encrypted Client Hello (ECH), иначе "".

`$ssl_protocol`

возвращает протокол установленного SSL-соединения.

`$ssl_server_cert_type`

принимает значения RSA, DSA, ECDSA, ED448, ED25519, SM2, RSA-PSS или `unknown` в зависимости от типа сертификата и ключа сервера.

`$ssl_server_name`

возвращает имя сервера, запрошенное через SNI.

`$ssl_session_id`

возвращает идентификатор сессии установленного SSL-соединения.

`$ssl_session_reused`

возвращает `r`, если сессия была использована повторно, иначе ".".

`$ssl_sigalg`

возвращает алгоритм подписи для сертификата сервера в установленном SSL-соединении.

Примечание

Переменная поддерживается только при использовании OpenSSL версии 3.5 и выше. При использовании более старых версий значением переменной будет пустая строка.

Примечание

Переменная доступна только для новых сессий.

`$ssl_sigalgs`

возвращает список алгоритмов подписи, предложенных клиентом в процессе SSL-рукопожатия, в виде списка, разделённого двоеточиями.

Примечание

При использовании OpenSSL версии 4.0 и выше алгоритмы возвращаются по именам; для алгоритма без имени используется соответствующий шестнадцатеричный код. SSL-библиотеки, предоставляющие прежний API алгоритмов подписи, возвращают все алгоритмы в виде шестнадцатеричных кодов. Если сборка не предоставляет поддерживаемого API алгоритмов подписи, значение переменной пустое.

SSL Preread

Позволяет извлекать информацию из сообщения `ClientHello` без терминации TLS, например имя сервера, запрошенное через `SNI`, протоколы, указанные в `ALPN`, или набор шифров (cipher suite).

Пример конфигурации

Выбор апстрима по имени сервера

```
map $ssl_preread_server_name $name {
    backend.example.com    backend;
    default                backend2;
}

upstream backend {
    server 192.168.0.1:12345;
    server 192.168.0.2:12345;
}

upstream backend2 {
    server 192.168.0.3:12345;
    server 192.168.0.4:12345;
}

server {
    listen      12346;
    proxy_pass  $name;
    ssl_preread on;
}
```

Выбор сервера по протоколу

```
map $ssl_preread_alpn_protocols $proxy {
    ~\bh2\b          127.0.0.1:8001;
    ~\bhttp/1.1\b     127.0.0.1:8002;
    ~\bxmlpp-client\b 127.0.0.1:8003;
}
```



```
server {
    listen      9000;
    proxy_pass  $proxy;
    ssl_preread on;
}
```

Выбор сервера по версии протокола SSL

```
map $ssl_preread_protocol $upstream {
    ""      ssh.example.com:22;
    "TLSv1.2" new.example.com:443;
    default  tls.example.com:443;
}

# ssh и https на одном порту
server {
    listen      192.168.0.1:443;
    proxy_pass  $upstream;
    ssl_preread on;
}
```

Выбор сервера по набору шифров

```
map $ssl_preread_ciphers $handler {
    "~TLS_GOST" 127.0.0.1:9443;
    default     127.0.0.1:8443;
}

server {
    listen      443;
    ssl_preread on;
    pass $handler;
}
```

Подробнее см. *Балансировка трафика на основе набора шифров*.

Директивы

ssl_preread

Синтаксис	ssl_preread on off;
По умолчанию	ssl_preread off;
Контекст	stream, server

Разрешает извлекать информацию из сообщения ClientHello на этапе предварительного чтения.

Встроенные переменные

`$ssl_preread_alpn_protocols`

Список протоколов, переданный клиентом через ALPN. Значения разделяются запятыми.

`$ssl_preread_ciphers`

Список шифров, предлагаемых в сообщении ClientHello, при установке TLS-соединения.

`$ssl_preread_protocol`

Максимальная версия протокола SSL, поддерживаемая клиентом.

`$ssl_preread_server_name`

Имя сервера, запрошенное через SNI.

Upstream

Предоставляет контекст для описания группы серверов, которые могут использоваться в директиве *proxy_pass*.

Пример конфигурации

```
upstream backend {
    hash $remote_addr consistent;
    zone backend 1m;

    server backend1.example.com:1935 weight=5;
    server unix:/tmp/backend3;
    server backend3.example.com      service=_example._tcp resolve;

    server backup1.example.com:1935  backup;
    server backup2.example.com:1935  backup;
}

resolver 127.0.0.53 status_zone=resolver;

server {
    listen 1936;
    proxy_pass backend;
}
```

Директивы

backup_switch

Синтаксис	<code>backup_switch permanent[=<i>время</i>];</code>
По умолчанию	—
Контекст	upstream

Директива включает возможность начать выбор серверов не с основной группы, а с *активной*, то есть той, где в предыдущий раз был успешно найден сервер. Если найти сервер в активной группе для очередного запроса не удастся, и поиск переходит к резервной группе, то уже эта группа становится активной, и последующие запросы сначала направляются на серверы этой группы.

Если параметр **permanent** определен без значения времени, группа остается активной после выбора, и автоматическая перепроверка групп с меньшим уровнем не происходит. Если *время* задано, то активный статус группы истекает через указанный интервал, и балансировщик снова проверяет группы с меньшим уровнем, возвращаясь к ним, если серверы работают нормально.

Пример:

```
upstream media_backend {
    zone media_backend 1m;
    server primary1.example.com:1935;
    server primary2.example.com:1935;

    server reserve1.example.com:1935 backup;
    server reserve2.example.com:1935 backup;

    backup_switch permanent=2m;
}
```

Если балансировщик переключается с основных серверов на резервную группу, все последующие запросы обрабатываются этой резервной группой в течение 2 минут. По истечении 2 минут балансировщик повторно проверяет основные серверы и снова делает их активными, если они работают нормально.

feedback

Синтаксис	<code>feedback</code> <i>переменная</i> <code>[inverse]</code> <code>[factor=число]</code> <code>[account=условная_переменная];</code>
По умолчанию	—
Контекст	upstream

Задаёт в `upstream` механизм балансировки нагрузки по обратной связи. Он динамически корректирует решения при балансировке, умножая вес каждого проксируемого сервера на среднее значение обратной связи, которое меняется с течением времени в зависимости от значения *переменной* и подчиняется необязательному условию.

Могут быть заданы следующие параметры:

переменная	Переменная, из которой берётся значение обратной связи. Она должна представлять собой метрику производительности или состояния; предполагается, что её передаёт сервер. Значение оценивается при каждом ответе от сервера и учитывается в скользящем среднем согласно настройкам inverse и factor .
inverse	Если параметр задан, значение обратной связи интерпретируется наоборот: более низкие значения указывают на лучшую производительность.
factor	Коэффициент, по которому значение обратной связи учитывается при расчёте среднего. Допустимы целые числа от 0 до 99. По умолчанию — 90. Среднее рассчитывается по формуле экспоненциального сглаживания . Чем больше коэффициент, тем меньше новые значения влияют на среднее; если указать 90, то будет взято 90 % от предыдущего значения и лишь 10 % от нового.
account	Указывает условную переменную, которая контролирует, как соединения учитываются при расчёте. Среднее значение обновляется с учётом значения обратной связи, только если условная переменная не равна "" или "0".

Примечание

По умолчанию трафик от активных проверок не включается в расчёт; комбинация переменной `$upstream_probe` с **account** позволяет включить и их или даже исключить все остальное.

Пример:

```
upstream backend {
    zone backend 1m;

    feedback $feedback_value factor=80 account=$condition_value;

    server backend1.example.com:1935 weight=1;
    server backend2.example.com:1935 weight=2;
}

map $protocol $feedback_value {
    "TCP"                100;
    "UDP"                75;
    default              10;
}

map $upstream_probe $condition_value {
    "high_priority"      "1";
    "low_priority"       "0";
    default              "1";
}
```

Эта конфигурация категоризирует серверы по уровням обратной связи на основе протоколов, используемых в отдельных сессиях, а также добавляет условие на `$upstream_probe`, чтобы учитывать только активную проверку `high_priority` или обычные клиентские сессии.

hash

Синтаксис	<code>hash <i>ключ</i> [consistent];</code>
По умолчанию	—
Контекст	upstream

Задаёт метод балансировки нагрузки для группы, при котором соответствие клиента серверу определяется при помощи хэшированного значения ключа. В качестве ключа можно использовать переменные. Следует отметить, что любое добавление или удаление серверов в группе может привести к перераспределению большинства ключей на другие серверы. Метод совместим с библиотекой Perl `Cache::Memcached`.

Пример использования:

```
hash $remote_addr;
```

При использовании доменных имен, разрешающихся в несколько IP-адресов (например, с параметром `resolve`), сервер не сортирует полученные адреса, поэтому их порядок может различаться на разных серверах, что влияет на распределение клиентов. Чтобы обеспечить одинаковое распределение, используйте параметр `consistent`.

Если задан параметр `consistent`, то вместо вышеописанного метода будет использоваться метод консистентного хэширования `ketama`. Метод гарантирует, что при добавлении сервера в группу или его удалении на другие серверы будет перераспределено минимальное число ключей. Применение метода для кэширующих серверов обеспечивает больший процент попаданий в кэш. Метод совместим с библиотекой Perl `Cache::Memcached::Fast` при значении параметра `ketama_points`, равном 160.

least_bandwidth

<i>Синтаксис</i>	<code>least_bandwidth both upstream downstream [factor=number];</code>
По умолчанию	—
<i>Контекст</i>	upstream

Модуль балансировки нагрузки **least_bandwidth** в потоковом модуле выбирает проксируемый сервер, который использует наименьший объем пропускной способности, стремясь равномерно распределить сетевой трафик, направляя новые сессии на сервер с наименьшим потреблением пропускной способности. Соответственно, чем меньше трафика, тем сервер считается менее загруженным и тем больше запросов будет на него отправляться.

upstream	Директива учитывает только пропускную способность, используемую от клиента к серверу .
downstream	Директива учитывает только пропускную способность, используемую от сервера к клиенту .
both	Директива учитывает сумму пропускной способности, используемой вверх (от клиента к серверу) и вниз по потоку (от сервера к клиенту).
factor	Необязательный коэффициент сглаживания для расчета скользящего среднего, в диапазоне от 0 до 99 (по умолчанию 90). Значение задается в процентах: при 90 берется 90 % от предыдущего среднего и 10 % от нового измерения. Более высокий коэффициент означает более медленную адаптацию к изменениям.

Модуль периодически вычисляет среднее использование пропускной способности для каждого проксируемого сервера, используя формулу скользящего среднего для сглаживания колебаний со временем. Когда начинается новая сессия, он выбирает проксируемый сервер с наименьшим средним использованием пропускной способности, скорректированным с учетом веса сервера, если он указан. Если несколько серверов имеют одинаковое минимальное среднее использование пропускной способности, применяется взвешенный метод round-robin.

least_conn

<i>Синтаксис</i>	<code>least_conn;</code>
По умолчанию	—
<i>Контекст</i>	upstream

Задаёт для группы метод балансировки нагрузки, при котором соединение передается серверу с наименьшим числом активных соединений, с учетом весов серверов. Если подходит сразу несколько серверов, они выбираются циклически (в режиме round-robin) с учетом их весов.

least_packets

<i>Синтаксис</i>	<code>least_packets both upstream downstream [factor=number];</code>
По умолчанию	—
<i>Контекст</i>	upstream

Модуль балансировки нагрузки **least_packets** в потоковом модуле выбирает проксируемый сервер, обрабатывающий наименьшее количество трафика на основе количества переданных пакетов,

стремясь равномерно распределить сетевую нагрузку, направляя новые сеансы на сервер с наименьшей скоростью передачи пакетов.

upstream	Директива учитывает только пакеты, отправленные от клиента к серверу .
downstream	Директива учитывает только пакеты, отправленные от сервера к клиенту .
both	Директива учитывает сумму пакетов, отправленных вверх (от клиента к серверу) и вниз по потоку (от сервера к клиенту).
factor	Необязательный коэффициент сглаживания для расчета скользящего среднего, в диапазоне от 0 до 99 (по умолчанию 90). Значение задается в процентах: при 90 берется 90 % от предыдущего среднего и 10 % от нового измерения. Более высокий коэффициент означает более медленную адаптацию к изменениям.

Модуль периодически обновляет среднюю скорость пакетов для каждого проксируемого сервера, используя формулу скользящего среднего для сглаживания колебаний со временем. Когда инициируется новый сеанс, он выбирает проксируемый сервер с наименьшей средней скоростью передачи пакетов, скорректированной с учетом веса сервера. Если несколько серверов имеют одно и то же минимальное значение, применяется взвешенный метод round-robin.

least_time

Синтаксис	<code>least_time connect first_byte last_byte [factor=number] [account=condition_variable];</code>
По умолчанию	—
Контекст	upstream

Задаёт для группы метод балансировки нагрузки, при котором вероятность передачи соединения активному серверу обратно пропорциональна среднему времени его ответа; чем оно меньше, тем больше соединений будет получать сервер.

connect	Директива учитывает среднее время установки соединения.
first_byte	Директива использует среднее время получения первого байта ответа.
last_byte	Директива использует среднее время получения полного ответа.

factor	Выполняет ту же функцию, что и <i>response_time_factor</i> , и переопределяет его, если параметр задан.
account	Указывает условную переменную, которая контролирует, какие соединения учитываются при расчете. Среднее значение обновляется, только если условная переменная соединения не равна "" или "0".

Примечание

По умолчанию активные проверки не включаются в расчет; комбинация переменной `$upstream_probe` с **account** позволяет включить их или даже исключить все остальное.

Текущие значения представлены как `connect_time`, `first_byte_time` и `last_byte_time` в объекте `health` сервера среди метрик апстрима в API.

no_port

Синтаксис	<code>no_port;</code>
По умолчанию	—
Контекст	upstream

Явно помечает группу серверов как не использующую порты: адреса ее серверов задаются без порта, а порт соединения указывается в каждой ссылающейся на группу директиве *proxy_pass*.

Директива предназначена для групп, в которых серверы не перечислены статически, а добавляются динамически через API конфигурации. Такие группы нуждаются в зонах разделяемой памяти и по умолчанию считаются использующими порты. Одну группу без портов можно затем использовать для проксирования на разные порты без дублирования списка серверов:

```
upstream backend {
    zone backend 1m;
    no_port;
}

server {
    listen 8443;
    proxy_pass backend:8443;
}

server {
    listen 8080;
    proxy_pass backend:8080;
}
```

Группа считается не использующей порты и без этой директивы, если адреса ее серверов заданы доменными именами или IP-адресами без портов. Директива `no_port` явно фиксирует это свойство группы и запрещает адреса с указанным портом.

В группе без портов допустимы только доменные имена и IP-адреса без портов; пути UNIX-сокетов и серверы с параметром `service=` для разрешения записей DNS SRV не допускаются. Доменные имена и IP-адреса в одной группе должны быть либо все с портами, либо все без них; смешивать эти варианты нельзя.

random

Синтаксис	<code>random [two];</code>
По умолчанию	—
Контекст	upstream

Задаёт для группы метод балансировки нагрузки, при котором соединение передается случайно выбранному серверу, с учетом весов серверов.

Если указан необязательный параметр `two`, Angie случайным образом выбирает два сервера, из которых выбирает сервер, используя указанный метод. Методом по умолчанию является *least_conn*, при котором соединение передается на сервер с наименьшим количеством активных соединений.

response_time_factor

Синтаксис	<code>response_time_factor число;</code>
По умолчанию	<code>response_time_factor 90;</code>
Контекст	<code>upstream</code>

Задаёт для метода балансировки нагрузки *least_time* коэффициент сглаживания **предыдущего** значения при вычислении среднего времени ответа по формуле **экспоненциально взвешенного скользящего среднего**.

Чем больше указанное *число*, тем меньше новые значения влияют на среднее; если указать 90, то будет взято 90 % от предыдущего значения и лишь 10 % от нового. Допустимые значения — от 0 до 99 включительно.

Текущие результаты вычислений представлены как `connect_time` (время установления соединения), `first_byte_time` (время получения первого байта ответа) и `last_byte_time` (время получения ответа целиком) в объекте `health` сервера среди метрик апстрима в API.

Примечание

При подсчете учитываются только успешные ответы; что считать неуспешным ответом, определяют директивы *proxy_next_upstream*.

server

Синтаксис	<code>server адрес [параметры];</code>
По умолчанию	—
Контекст	<code>upstream</code>

Задаёт адрес и другие параметры сервера. Адрес может быть указан в виде доменного имени или IP-адреса либо в виде пути UNIX-сокета после префикса `unix:.` Доменное имя, которому соответствует несколько IP-адресов, задаёт сразу несколько серверов. Для доменного имени или IP-адреса порт обязателен, кроме группы без портов, где он опускается и задаётся в директиве *proxy_pass*.

Могут быть заданы следующие параметры:

<code>weight=число</code>	Задаёт вес сервера. По умолчанию — 1.
<code>max_conns=число</code>	Ограничивает максимальное число одновременных активных соединений к проксируемому серверу. Значение по умолчанию равно 0 и означает, что ограничения нет. Если группа не находится в зоне разделяемой памяти, то ограничение работает отдельно для каждого рабочего процесса.

`max_fails=число` — задаёт число неудачных попыток связи с сервером, которые должны произойти в течение заданного `fail_timeout` времени для того, чтобы сервер считался недоступным; после этого он будет повторно проверен через то же самое время.

В данном случае неудачной попыткой считается ошибка или таймаут при установке соединения с сервером.

Примечание

Если директива **server** в группе разрешается в несколько серверов, ее настройка **max_fails** применяется к каждому серверу отдельно.

Если после разрешения всех директив **server** в апстриме остается только один сервер, настройка **max_fails** не действует и будет проигнорирована.

max_fails=1	Число попыток по умолчанию.
max_fails=0	Отключает учет попыток.

fail_timeout=время — задает период времени, в течение которого должно произойти определенное число неудачных попыток связи с сервером (**max_fails**), чтобы сервер считался недоступным. Затем сервер остается недоступным в течение того же самого времени, прежде чем будет проверен повторно.

Значение по умолчанию — 10 секунд.

Примечание

Если директива **server** в группе разрешается в несколько серверов, ее настройка **fail_timeout** применяется к каждому серверу отдельно.

Если после разрешения всех директив **server** в апстриме остается только один сервер, настройка **fail_timeout** не действует и будет проигнорирована.

backup	Помечает сервер как запасной. На него будут передаваться запросы в случае, если не работают основные серверы. Можно создать несколько групп резервирования серверов и задать уровень для каждой группы с помощью <code>backup=<уровень_группы></code>. Например, <code>backup=1</code> (или просто <code>backup</code>) — группа резервных серверов первого уровня, (в API отображается как <code>true</code>), <code>backup=2</code> — группа резервных серверов второго уровня, (в API отображается как соответствующее число). Если директива <code>server</code> использует SRV-записи DNS для получения адресов проксируемых серверов, то уровень группы резервирования можно задать с помощью параметра <code>backup=prio</code>. В этом случае абсолютное значение приоритета <code>priority=n</code> из SRV-записи будет использовано в качестве уровня резервной группы: <code>backup=n</code>. Если параметр <code>backup=prio</code> не задан, уровень резервирования будет определяться через относительное значение приоритета <code>priority=n</code> из SRV-записи. В этом случае директива <code>backup=n</code> задает базовый уровень отсчета (например <code>backup=3</code>), а относительные уровни резервирования отсчитываются от этого уровня. Сервер с наименьшим значением <code>priority</code> получит уровень <code>backup=3</code>, следующий за ним — <code>backup=4</code> и так далее. Если уровень отсчета <code>backup=n</code> не задан, отсчет будет идти от нуля: сервер с наименьшим значением <code>priority</code> получит <code>backup=0</code> и станет основным, за ним — <code>backup=1</code>, резервный, и так далее. Если задана директива <code>backup_switch</code>, также применяется ее логика активного резервирования.
down	Помечает сервер как постоянно недоступный.
drain	Помечает сервер как разгружаемый (<code>draining</code>); это значит, что он получает только запросы сессий, привязанных ранее через <code>sticky</code> . В остальном поведение такое же, как в режиме <code>down</code> .
spare	Помечает сервер как запасной (<code>spare</code>) в рамках одной группы серверов. Запасные серверы простаивают, пока основные серверы группы справляются с нагрузкой; когда основные серверы становятся недоступны, запасные вводятся в работу (в порядке убывания веса), пока мощность группы не восстановится, а после восстановления основных серверов снова переводятся в режим простоя. Можно добавить несколько запасных серверов. Подробнее см. запасные серверы.

⚠ Предупреждение

Параметр `backup` нельзя использовать совместно с методами балансировки нагрузки `hash` и `random`.

Параметры `down` и `drain` взаимно исключают.

resolve	Позволяет отслеживать изменения списка IP-адресов, соответствующего доменному имени, и обновлять его без перезагрузки конфигурации. При этом группа должна находиться в зоне разделяемой памяти; также должен быть определен преобразователь имен в адреса.
service=имя	Включает преобразование SRV-записей DNS и задает имя сервиса. При указании этого параметра необходимо также задать параметр <i>resolve</i>, не указывая порт сервера при имени хоста. Если в имени службы нет точек, формируется имя по стандарту RFC: к имени службы добавляется префикс <i>_</i>, затем через точку добавляется <i>_tcp</i>. Так, имя службы <i>http</i> даст в результате <i>_http._tcp</i>. Angie разрешает SRV-записи, объединяя нормализованное имя службы и имя хоста и получая список серверов для полученной комбинации через DNS, вместе с их приоритетами и весами. • SRV-записи с наивысшим приоритетом (те, которые имеют минимальное значение приоритета) разрешаются как основные серверы, а прочие записи становятся запасными серверами. Если <i>backup</i> установлено с <i>server</i>, SRV-записи с наивысшим приоритетом разрешаются как запасные серверы, а прочие записи игнорируются. • Вес аналогичен параметру <i>weight</i> директивы <i>server</i>. Если вес задан как в самой директиве, так и в SRV-записи, используется вес, установленный в директиве.

В этом примере выполняется поиск записи *_http._tcp.backend.example.com*:

```
server backend.example.com service=http resolve;
```

sid=id	Задаёт ID сервера в группе. Если параметр не задан, то ID задается как шестнадцатеричный MD5-хэш IP-адреса и порта или пути UNIX-сокета.
slow_start=время	Задаёт время восстановления веса сервера, возвращающегося к работе при балансировке нагрузки методом round-robin или <i>least_conn</i>. Если параметр задан и сервер после сбоя снова считается работающим с точки зрения <i>max_fails</i> и <i>upstream_probe</i>, то такой сервер равномерно набирает указанный для него вес в течение заданного времени. Если параметр не задан, то в аналогичной ситуации сервер сразу начинает работу с указанным для него весом.

Примечание

Если в апстриме задан только один *server*, *slow_start* не работает и будет игнорироваться.

state

Синтаксис	state <i>файл</i> ;
По умолчанию	—
Контекст	upstream

Указывает *файл*, где постоянно хранится список серверов апстрима. Для хранения таких файлов специально создается каталог */var/lib/angie/state/* с соответствующими правами доступа, и в

конфигурации остается добавить лишь имя файла:

```
upstream backend {
    zone backend 1m;
    state /var/lib/angie/state/<ИМЯ ФАЙЛА>;
}
```

Список серверов здесь имеет формат, аналогичный `server`. Содержимое файла изменяется при любом изменении серверов в разделе `/config/stream/upstreams/` через API конфигурации.

Предупреждение

Чтобы использовать директиву `state` в блоке `upstream`, в нем не должно быть директив `server`, но нужна зона разделяемой памяти (`zone`).

sticky

Синтаксис	<code>sticky route значение...;</code> <code>sticky learn zone=zone create=\$create_var1... lookup=\$lookup_var1...</code> <code>[connect] [norefresh] [timeout=time];</code> <code>sticky learn lookup=\$lookup_var1... remote_action=uri</code> <code>remote_result=\$remote_var [remote_uri=uri];</code>
По умолчанию	—
Контекст	upstream

Настраивает привязку клиентских сессий к проксируемым серверам в режиме, заданном первым параметром; для разгрузки серверов, у которых задана директива `sticky`, можно использовать опцию `drain` в блоке `server`.

Предупреждение

Директива `sticky` должна использоваться после всех директив, задающих тот или иной метод балансировки нагрузки, иначе она не будет работать.

Режим route

Этот режим использует predetermined идентификаторы маршрутов, которые могут быть встроены в свойства соединения, доступные Angie. Он менее гибок, так как зависит от predetermined значений, но лучше подходит, если такие идентификаторы уже используются.

Здесь при установлении соединения проксируемый сервер может назначить клиенту маршрут и вернуть его идентификатор способом, известным им обоим. В качестве идентификатора маршрута должно использоваться значение параметра `sid` директивы `server`. Учтите, что параметр дополнительно хэшируется, если задана директива `sticky_secret`.

Последующие соединения от клиентов, желающих использовать этот маршрут, должны содержать выданный сервером идентификатор, причем так, чтобы он попал в переменные Angie.

В параметрах директивы указываются строки, которые могут содержать переменные для маршрутизации. Чтобы выбрать сервер, куда направляется входящее соединение, используется первое непустое значение; она затем сравнивается с параметром `sid` директивы `server`. Если выбрать сервер не удастся или выбранный сервер не может принять соединение, то будет выбран другой сервер согласно настроенному методу балансировки.

Здесь Angie ищет идентификатор маршрута в переменной `$route`, получающей значение на основе `$ssl_preread_server_name` (обратите внимание, что нужно включить `ssl_preread`):

```
stream {

    map $ssl_preread_server_name $route {

        a.example.com      a;
        b.example.com      b;
        default            "";
    }

    upstream backend {

        zone backend 1m;

        server 127.0.0.1:8081 sid=a;
        server 127.0.0.1:8082 sid=b;

        sticky route $route;
    }

    server {

        listen 127.0.0.1:8080;

        ssl_preread on;

        proxy_pass backend;
    }
}
```

Режим `learn`

В этом режиме для привязки клиента к конкретному проксируемому серверу используется динамически генерируемый ключ; этот режим более гибок, так как назначает серверы на ходу, хранит сеансы в зоне разделяемой памяти и поддерживает различные способы передачи идентификаторов сессий.

Здесь сессия создается на основе свойств соединения, идущих от проксируемого сервера. С параметрами `create` и `lookup` перечисляются переменные, указывающие, как создаются новые и ищутся существующие сессии. Оба параметра можно использовать по несколько раз.

Идентификатором сессии служит значение первой непустой переменной, указанной с `create`; например, это может быть имя проксируемого сервера.

Сессии хранятся в зоне разделяемой памяти; ее имя и размер задаются параметром `zone`. Если к сессии не было обращений в течение времени `timeout`, она удаляется. Значение по умолчанию — 1 час.

По умолчанию Angie продлевает срок действия сессии, обновляя метку времени последнего обращения при каждом ее использовании. Параметр `norefresh` меняет это поведение: сессия истекает строго по таймауту, даже если используется.

Последующие соединения от клиентов, желающих использовать сессию, должны содержать ее идентификатор. Параметр `lookup` ищет идентификатор сессии в соединении по заданному для него списку переменных, останавливаясь на первой непустой. Если ничего не найдено — запрос считается новым. Значение найденного идентификатора сопоставляется с сессиями в разделяемой памяти. Если выбрать сервер не удастся или выбранный сервер не может обработать соединение, то будет выбран другой сервер согласно настроенному методу балансировки.

Параметр `connect` позволяет создать сессию сразу после установления соединения с проксируемым

сервером. Без него сессия создается только после завершения обработки соединения. (В случае UDP-соединений сессии создаются сразу после выбора сервера.)

В примере Angie создает и ищет сессии, используя переменную `$rdp_cookie`:

```
stream {
    upstream backend {
        zone backend 1m;

        server 127.0.0.1:3390;
        server 127.0.0.1:3391;

        sticky learn lookup=$rdp_cookie create=$rdp_cookie zone=sessions:1m;
    }

    server {
        listen 127.0.0.1:3389;

        rdp_preread on;

        proxy_pass backend;
    }
}
```

Режим `learn` с `remote_action`

Параметры `remote_action` и `remote_result` позволяют динамически назначать идентификаторы сессий и управлять ими с использованием удаленного хранилища сессий.

В отличие от режима `learn` с `zone`, данный режим не кэширует сессии локально и обращается к удаленному хранилищу при каждом соединении.

Параметр `remote_action` должен указывать на `location` в контексте *client*. Параметр `remote_uri` задает URI клиентского HTTP-запроса к указанному `location`. По умолчанию он равен `/`. Значение `remote_uri` может содержать переменные.

Общий принцип работы режима таков: если идентификатор сессии не найден локально, Angie отправляет синхронный подзапрос в некое удаленное хранилище, заданное параметром `remote_action`.

При поступлении нового соединения Angie выполняет следующие действия:

- Сначала извлекается идентификатор сессии из первой непустой переменной в списке `lookup`. Если все переменные пустые, используется обычный алгоритм балансировки нагрузки без привязки.
- Затем Angie отправляет в удаленное хранилище, заданное параметром `remote_action`, синхронный HTTP-подзапрос, который должен содержать в понятном хранилищу виде:
 - идентификатор *сессии* из параметра `lookup` (в конфигурации это переменная `$sticky_sessid`);
 - идентификатор предварительно выбранного *сервера*: значение параметра `sid=` из директивы `server`, если оно задано, либо MD5-хэш имени сервера (в конфигурации это переменная `$sticky_sid`).

Переменные `$sticky_sessid` и `$sticky_sid` автоматически экспортируются в HTTP-контекст с префиксом `stream_`: `$stream_sticky_sessid`, `$stream_sticky_sid`. Это позволяет напрямую использовать их в HTTP-директивах, например через HTTP-заголовки с помощью *`proxy_set_header`*.

- Удаленное хранилище обрабатывает запрос и возвращает HTTP-ответ:

Ответ с кодом 200, 201 или 204 подтверждает выбранный сервер. Удаленное хранилище может одновременно с этим вернуть альтернативный идентификатор сервера в HTTP-заголовке или в теле ответа; извлечь его можно через `remote_result`.

При получении от хранилища любого другого HTTP-кода (включая ошибки сети и таймауты) либо несуществующего идентификатора сервера Angie использует изначально выбранный сервер.

Идентификатор сервера извлекается из ответа удаленного хранилища через параметр `remote_result`: в нем можно указывать переменные с префиксом `upstream_http_`, которые создаются Angie автоматически для доступа к заголовкам HTTP-ответов от удаленного хранилища, или `$sent_body` для использования тела ответа. Например, заголовок `X-Sid: server1` в таком ответе становится доступным в переменной `$upstream_http_x_sid` со значением `server1`.

Ниже показан упрощенный пример конфигурации. Удаленное хранилище возвращает идентификатор сервера в заголовке `X-Sticky-Sid` и таким образом подтверждает или переопределяет выбор Angie:

```
http {
    client {
        location @sticky_client1 {
            # используем переменные из потокового upstream;
            # он добавляет эти переменные в HTTP-контекст с префиксом stream_*
            proxy_set_header X-Sticky-Sessid $stream_sticky_sessid;
            proxy_set_header X-Sticky-Sid $stream_sticky_sid;
            proxy_set_header X-Sticky-Last $msec;
            proxy_pass http://127.0.0.1:8080;

            proxy_cache remote;
            proxy_cache_valid 200 1d;
            proxy_cache_key $scheme$proxy_host$request_uri$stream_sticky_sessid;
        }
    }
}

stream {
    upstream u {
        zone u 1m;

        server 127.0.0.1:8081 sid=backend-01;
        server 127.0.0.1:8082 sid=backend-02;

        sticky learn lookup=$remote_addr           # переменная stream
        remote_action=@sticky_client1               # location из блока client
        remote_result=$upstream_http_x_sticky_sid   # HTTP-переменная
        remote_uri=/foo;                            # по умолчанию - /
    }

    server {
        listen 127.0.0.1:8080;
        proxy_pass u;
    }
}
```

```
}

```

Здесь при следующем ответе от удаленного хранилища:

```
HTTP/1.1 200 OK
...
X-Sticky-Sid: backend-01
X-Session-Info: active

```

Становятся доступными две переменные:

- `$upstream_http_x_sticky_sid`, со значением `backend-01`;
- `$upstream_http_x_session_info`, со значением `active`.

Так как переменная `$upstream_http_x_sticky_sid` указана в параметре `remote_result`, то ее значение будет использовано для выбора сервера с `sid=backend-01`.

Директива `sticky` учитывает состояние серверов в *upstream*:

- Серверы, помеченные как `down` или временно недоступные из-за сбоев, исключаются из выбора.
- Серверы, которые достигли максимального количества соединений (при использовании `max_conns`), временно пропускаются.
- Серверы с опцией `drain` могут быть выбраны для создания новых сессий в режиме `sticky` при совпадении идентификаторов.
- Если ранее недоступный сервер восстанавливается, `sticky` автоматически возобновляет его использование.

Поведение `sticky` можно дополнительно настроить директивами *sticky_secret* и *sticky_strict*. Если в ходе работы `sticky` выбрать сервер не удастся или он недоступен, запрос будет обработан согласно выбранному методу балансировки нагрузки, если только не включена директива `sticky_strict`. В режиме `sticky_strict on`; запрос отклоняется с ошибкой.

Зоны разделяемой памяти, указываемые в параметре `zone` директивы `sticky`, не могут использоваться совместно различными *upstream*; каждая группа должна использовать свою собственную зону.

sticky_secret

Синтаксис	<code>sticky_secret строка;</code>
По умолчанию	—
Контекст	<code>upstream</code>

Добавляет *строку* как соль в функцию MD5-хэширования для директивы `sticky` в режиме `route`. *Строка* может содержать переменные, например `$remote_addr`:

```
upstream backend {
    zone backend 1m;
    server 127.0.0.1:8081 sid=a;
    server 127.0.0.1:8082 sid=b;

    sticky route $route;
    sticky_secret my_secret.$remote_addr;
}

```

Соль добавляется после хэшируемого значения; чтобы независимо проверить механизм хэширования:


```
$ echo -n "<VALUE><SALT>" | md5sum
```

sticky_strict

Синтаксис	<code>sticky_strict on off;</code>
По умолчанию	<code>sticky_strict off;</code>
Контекст	<code>upstream</code>

При включении Angie будет возвращать клиенту ошибку соединения, если желаемый сервер недоступен, вместо использования любого другого доступного сервера, как это происходит, когда в группе нет доступных серверов.

upstream

Синтаксис	<code>upstream <i>имя</i> { ... }</code>
По умолчанию	—
Контекст	<code>stream</code>

Описывает группу серверов. Серверы могут слушать на разных портах. Кроме того, можно одновременно использовать серверы, слушающие на TCP- и UNIX-сокетах.

Пример:

```
upstream backend {
    zone backend 1m;
    server backend1.example.com:1935 weight=5;
    server 127.0.0.1:1935 max_fails=3 fail_timeout=30s;
    server unix:/tmp/backend2;
    server backend3.example.com:1935 resolve;

    server backup1.example.com:1935 backup;
}
```

По умолчанию соединения распределяются по серверам циклически (в режиме round-robin) с учетом весов серверов. В вышеприведенном примере каждые 7 соединений будут распределены так: 5 соединений на backend1.example.com:1935 и по одному соединению на второй и третий серверы. Распределение является *равномерным*: соединения к серверу с большим весом распределяются по всему циклу, а не отправляются подряд одной группой.

Если при попытке работы с сервером происходит ошибка, то соединение передается следующему серверу, и так далее до тех пор, пока не будут опробованы все работающие серверы. Если связь с серверами не удалась, соединение будет закрыто.

Примечание

По умолчанию сервер, у которого изредка происходят неудачные попытки, не достигающие порога `max_fails`, временно получает меньшую долю соединений и восстанавливает свою полную долю в течение последующих соединений. Это отличается от `slow_start`, который плавно возвращает сервер к работе только после того, как сервер был помечен недоступным и затем восстановился.

zone

Синтаксис	<code>zone имя [размер];</code>
По умолчанию	—
Контекст	upstream

Задаёт имя и размер зоны разделяемой памяти, в которой хранятся конфигурация группы и её рабочее состояние, разделяемые между рабочими процессами. В одной и той же зоне могут быть сразу несколько групп. В этом случае достаточно указать размер только один раз.

Примечание

Содержимое зоны сохраняется при перезагрузке только в том случае, если настроенный **размер** не изменился. Любое изменение — увеличение или уменьшение — приводит к пересозданию зоны с потерей всех данных.

Примечание

Метрики группы собираются, только если настроена эта зона. Без нее группа не отображается в `/status/stream/upstreams/<upstream>`, в виджете «TCP/UDP Upstreams» и в выводе Prometheus, и предупреждение при этом не выводится.

Встроенные переменные

Модуль `stream_upstream` поддерживает следующие встроенные переменные:

`$sticky_sessid`

Используется с `remote_action` в *sticky*; хранит начальный идентификатор сессии, взятый из `lookup`.

`$sticky_sid`

Используется с `remote_action` в *sticky*; хранит идентификатор сервера, предварительно связанный с сессией.

В `sticky_sid` содержится значение параметра `sid=` из блока *upstream* директивы `server`, если оно задано, либо MD5-хэш имени сервера.

`$upstream_addr`

хранит IP-адрес и порт или путь к UNIX-сокету сервера группы. Если при проксировании были сделаны обращения к нескольким серверам, то их адреса разделяются запятой, например:

```
192.168.1.1:1935, 192.168.1.2:1935, unix:/tmp/sock"
```

Если сервер не может быть выбран, то переменная хранит *имя* группы серверов.

`$upstream_bytes_received`

число байт, полученных от сервера группы. Значения нескольких соединений разделяются запятыми и двоеточиями подобно адресам в переменной `$upstream_addr`.

`$upstream_bytes_sent`

число байт, переданных на сервер группы. Значения нескольких соединений разделяются запятыми и двоеточиями подобно адресам в переменной `$upstream_addr`.

`$upstream_connect_time`

хранит время, затраченное на установление соединения с сервером группы; время хранится в секундах с точностью до миллисекунд. Времена нескольких соединений разделяются запятыми и двоеточиями подобно адресам в переменной `$upstream_addr`.

`$upstream_current_addr`

адрес проксируемого сервера, выбранного для текущей попытки соединения; в отличие от `$upstream_addr`, где накапливается список адресов всех серверов, к которым были обращения при повторных попытках, эта переменная всегда содержит только один адрес и становится доступна сразу после выбора сервера, в том числе во время вычисления переменных в директиве `proxy_bind`.

`$upstream_first_byte_time`

время получения первого байта данных; время хранится в секундах с точностью до миллисекунд. Времена нескольких соединений разделяются запятыми подобно адресам в переменной `$upstream_addr`.

`$upstream_session_time`

длительность сессии в секундах с точностью до миллисекунд. Времена нескольких соединений разделяются запятыми подобно адресам в переменной `$upstream_addr`.

`$upstream_sticky_status`

Статус соединений с привязкой.

""	Соединение направлено в группу серверов, где привязка не используется.
NEW	Соединение не содержит информации о привязке к серверу.
HIT	Соединение с привязкой направлено на желаемый сервер.
MISS	Соединение с привязкой направлено на сервер, выбранный по алгоритму балансировки.

Статусы из нескольких соединений разделяются запятыми и двоеточиями аналогично адресам в переменной `$upstream_addr`.

Upstream Probe

Реализует активные проверки работоспособности (health probes) для Upstream.

Пример конфигурации

```
server {
    listen ...;

    # ...
    proxy_pass backend;
    upstream_probe_timeout 1s;

    upstream_probe backend_probe
        port=12345
```

```
interval=5s
test=$good
essential
fails=3
passes=3
max_response=512k
mode=onfail
"send=data:GET / HTTP/1.0\r\n\r\n";
}
```

Примечание

Согласно спецификации RFC 2616 (HTTP/1.1) и RFC 9110 (HTTP Semantics), заголовки HTTP должны разделяться последовательностью CRLF (\r\n), а не просто \n.

Директивы

upstream_probe

<i>Синтаксис</i>	<code>upstream_probe</code> <i>имя</i> [<code>port=число</code>] [<code>interval=время</code>] [<code>test=условие</code>] [<code>ping</code> [<code>ping_timeout=время</code>]] [<code>essential</code> [<code>persistent</code>]] [<code>fails=число</code>] [<code>passes=число</code>] [<code>max_response=размер</code>] [<code>mode=always idle onfail</code>] [<code>udp</code>] [<code>send=строка</code>];
По умолчанию	—
<i>Контекст</i>	server

Задаёт активную проверку работоспособности серверов апстрима, указанного в директиве *proxy_pass* в том же контексте **server**, где находится директива **upstream_probe**.

Сервер проходит проверку, если запрос к нему успешно выполняется с учетом всех параметров самой директивы **upstream_probe** и всех параметров, влияющих на использование апстримов тем контекстом **server**, где она задана, в том числе директивы *proxy_next_upstream*.

Чтобы использовать проверки, в апстриме необходима зона разделяемой памяти (zone). Для одного апстрима можно определить несколько проверок.

Могут быть заданы следующие параметры:

имя	Обязательное имя проверки.
port	Альтернативный порт для запроса.
interval	<i>Интервал</i> между проверками. По умолчанию — 5s.
ping	Использовать проверку методом ICMP Echo вместо TCP- или UDP-пакета. При выборе этого режима Angie ADC отправляет эхо-запрос (ICMP Echo Request) на IP-адрес сервера и ожидает ответ (Echo Reply). Проверка осуществляется по IP-адресу, а номер порта из upstream игнорируется.
 Примечание Параметры test, port, udp и send несовместимы с ping и игнорируются. Для отправки ICMP-пакетов необходимы соответствующие права: на Linux — запуск процессов от имени группы, указанной в <code>/proc/sys/net/ipv4/ping_group_range</code>; на FreeBSD — права суперпользователя или наличие <i>capability</i> на использование raw-сокетов. При использовании ICMP-проверок переменные, основанные на данных ответа сервера или отправленных пакетах, всегда пусты; результат определяется только фактом получения ICMP Echo Reply.	
ping_timeout	<i>Время</i> ожидания ICMP-ответа; по умолчанию — 1s. Используется только вместе с ping .
test	Проверяемое при запросе условие; задается строкой из переменных. Если результат подстановки всех переменных — "" или "0", проверка не пройдена.
essential	Если параметр задан, то изначально состояние сервера подлежит уточнению и клиентские запросы не передаются ему, пока проверка не будет пройдена.
persistent	Установка этого параметра требует сначала включить essential ; серверы с persistent , работавшие до <i>перезагрузки конфигурации</i> , начинают получать запросы без необходимости сначала пройти эту проверку.
fails	Число последовательных неуспешных запросов, при котором проверка считает сервер неработающим. По умолчанию — 1.
passes	Число последовательных успешных запросов, при котором проверка считает сервер работающим. По умолчанию — 1.
max_response	Максимальный объем памяти для ответа. Если задано нулевое <i>значение</i> , ожидание ответа отключается. По умолчанию — 256k.
mode	Режим проверки в зависимости от работоспособности серверов: • always — серверы проверяются независимо от состояния; • idle — проверяются неработающие серверы, а также серверы, где с последнего клиентского запроса прошло время interval. • onfail — проверяются серверы только в неработающем состоянии. По умолчанию — always .
udp	Если параметр указан, используется протокол UDP. По умолчанию для проверок используется TCP.
send	Отправляемые для проверки данные: встроенные данные с префиксом data: или путь к файлу (абсолютный или относительно <code>/usr/local/angie/</code>). При использовании файла: • <i>Рабочий процесс</i> открывает и читает файл при каждом обращении; содержимое не сохраняется в памяти. • Перезагрузка конфигурации при изменении файла не требуется; новое содержимое будет прочитано при следующем обращении. • Необходимые права доступа: 644 для файла, 755 для директории. • Обновляйте файлы командой перемещения (mv), а не прямым редактированием.

Пример:

```
upstream backend {
    zone backend 1m;

    server a.example.com;
    server b.example.com;
}

map $upstream_probe_response $good {
    ~200      "1";
    default   "";
}

server {
    listen ...;

    # ...
    proxy_pass backend;
    upstream_probe_timeout 1s;

    upstream_probe backend_probe
        port=12345
        interval=5s
        test=$good
        essential
        persistent
        fails=3
        passes=3
        max_response=512k
        mode=onfail
        "send=data:GET / HTTP/1.0\r\n\r\n";
}
```

Детали работы:

- Изначально сервер не получает клиентские запросы, пока не пройдет *все* заданные для него проверки с параметром **essential** (пропуская помеченные как **persistent**, если конфигурация перезагружена и до этого сервер считался работающим). Если таких проверок нет, сервер считается работающим.
- Сервер считается неработающим и не получает клиентские запросы, если *какая-либо* заданная для него проверка достигает своего порога **fails** или сам сервер достигает порога **max_fails**.
- Чтобы неработающий сервер снова мог считаться работающим, *все* заданные для него проверки должны достичь своего порога **passes**; после этого учитывается порог **max_fails**.

upstream_probe_timeout

Синтаксис	upstream_probe_timeout время;
По умолчанию	upstream_probe_timeout 50s;
Контекст	server

Задаёт максимальное время бездействия установленного с сервером соединения для проверок, настроенных с помощью директивы *upstream_probe*; при превышении этого предела соединение будет закрыто.

Встроенные переменные

Модуль `stream_upstream` поддерживает следующие встроенные переменные:

`$upstream_probe`

Имя активной сейчас проверки `upstream_probe`.

`$upstream_probe_response`

Содержимое ответа, полученного в ходе активной проверки `upstream_probe`.

Базовый потоковый модуль реализует основную функциональность для обработки TCP- и UDP-соединений: это определение серверных блоков, маршрутизация трафика, настройка проксирования, поддержка SSL/TLS и управление подключениями для потоковых сервисов, таких как базы данных, DNS и другие протоколы, работающие на основе TCP и UDP.

Остальные модули этого раздела расширяют эту функциональность, позволяя гибко настраивать и оптимизировать работу потокового сервера под различные сценарии и требования.

Пример конфигурации

```
worker_processes auto;

error_log /var/log/angie/error.log info;

events {
    worker_connections 1024;
}

stream {
    upstream backend {
        hash $remote_addr consistent;

        server backend1.example.com:12345 weight=5;
        server 127.0.0.1:12345 max_fails=3 fail_timeout=30s;
        server unix:/tmp/backend3;
    }

    upstream dns {
        server 192.168.0.1:53535;
        server dns.example.com:53;
    }

    server {
        listen 12345;
        proxy_connect_timeout 1s;
        proxy_timeout 3s;
        proxy_pass backend;
    }

    server {
        listen 127.0.0.1:53 udp reuseport;
        proxy_timeout 20s;
        proxy_pass dns;
    }

    server {
        listen [::1]:12345;
```

```
    proxy_pass unix:/tmp/stream.socket;
}
}
```

Директивы

error_log_user_tag

Синтаксис	error_log_user_tag <i>значение</i> ;
По умолчанию	—
Контекст	stream, server

Добавляет тег, зависящий от сессии, в записи *error_log*. *Значение* может содержать переменные. Директива может задаваться несколько раз для добавления нескольких тегов. Теги используются в фильтрах *filter=tag*: директивы *error_log*.

listen

Синтаксис	listen <i>адрес[:порт]</i> [<i>default_server</i>] [<i>ssl</i>] [<i>udp</i>] [<i>proxy_protocol</i>] [<i>setfib=число</i>] [<i>fastopen=число</i>] [<i>backlog=число</i>] [<i>rcvbuf=размер</i>] [<i>sndbuf=размер</i>] [<i>accept_filter=фильтр</i>] [<i>deferred</i>] [<i>bind</i>] [<i>ipv6only=on off</i>] [<i>reuseport</i>] [<i>multipath</i>] [<i>so_keepalive=on off</i>][<i>keepidle</i>]:[<i>keepintvl</i>]:[<i>keepcnt</i>];
По умолчанию	—
Контекст	server

Задаёт *адрес* и *порт* для сокета, на котором сервер будет принимать соединения. Можно указать только *порт*, и тогда Angie будет слушать на всех доступных IPv4-интерфейсах (и IPv6, если он включен). Кроме того, *адрес* может быть именем хоста, например:

```
listen 127.0.0.1:12345;
listen *:12345;
listen 12345;      # то же, что и *:12345
listen localhost:12345;
```

IPv6-адреса задаются в квадратных скобках:

```
listen [::1]:12345;
listen [::]:12345;
```

Диапазоны портов задаются при помощи указания первого и последнего порта через дефис:

```
listen 127.0.0.1:12345-12399;
listen 12345-12399;
```

UNIX-сокеты задаются префиксом *unix*:

```
listen unix:/var/run/angie.sock;
```

Примечание

Несколько серверов могут слушать одну и ту же пару *адрес:порт*; в этом случае сервер, обрабатывающий соединение, выбирается по `server_name`, а это имя Angie узнает только из SNI. См. порядок выбора сервера для сессии.

<code>default_server</code>	делает этот сервер сервером по умолчанию для данной пары <i>адрес:порт</i> : он обрабатывает соединения, имя сервера в которых не совпадает ни с одним другим сервером, слушающим эту пару. Если этот параметр не указан ни в одной директиве <code>listen</code> , сервером по умолчанию становится первый в конфигурации сервер, слушающий эту пару.
<code>ssl</code>	указывает на то, что все соединения, принимаемые на данном слушающем сокете, должны работать в режиме SSL.
<code>udp</code>	конфигурирует слушающий сокет для работы с датаграммами. Для обработки пакетов с одного адреса и порта в рамках одной сессии необходимо также указывать параметр <i>reuseport</i> . Параметр несовместим с <code>ssl</code> , <code>proxy_protocol</code> , <code>fastopen</code> , <code>backlog</code> , <code>accept_filter</code> , <code>deferred</code> , <code>multipath</code> и <code>so_keepalive</code> ; Angie не принимает конфигурацию, в которой они указаны вместе.
<code>proxy_protocol</code>	указывает на то, что все соединения, принимаемые на данном порту, должны использовать протокол PROXY . Клиент должен передать заголовок целиком за время, заданное директивой <code>proxy_protocol_timeout</code> , иначе Angie закрывает соединение. Переданные в заголовке значения заполняют семейство переменных <code>\$proxy_protocol_addr</code> и с помощью модуля RealIP могут заменить адрес клиента.

В директиве `listen` можно также указать несколько дополнительных параметров, специфичных для связанных с сокетами системных вызовов.

<code>setfib=число</code>	устанавливает связанную таблицу маршрутизации, FIB (параметр <code>SO_SETFIB</code>) для слушающего сокета. Пока это работает только на FreeBSD.
<code>fastopen=число</code>	включает "TCP Fast Open" для слушающего сокета и ограничивает максимальную длину очереди соединений, которые еще не завершили процесс трехстороннего рукопожатия.
 Предупреждение Не включайте "TCP Fast Open", не убедившись, что сервер может адекватно обрабатывать многократное получение одного и того же SYN-пакета с данными.	
<code>backlog=число</code>	задает параметр <code>backlog</code> в вызове <code>listen()</code> , который ограничивает максимальный размер очереди ожидающих приема соединений. По умолчанию <code>backlog</code> устанавливается равным <code>-1</code> для FreeBSD, DragonFly BSD и macOS, и <code>511</code> для других платформ.
<code>rcvbuf=размер</code>	задает размер буфера приема (параметр <code>SO_RCVBUF</code>) для слушающего сокета.
<code>sndbuf=размер</code>	задает размер буфера передачи (параметр <code>SO_SNDBUF</code>) для слушающего сокета.
<code>accept_filter=фи</code>	задает имя принимающего фильтра (параметр <code>SO_ACCEPTFILTER</code>) для слушающего сокета, который фильтрует входящие соединения перед их передачей в <code>accept()</code> . Работает только на FreeBSD и NetBSD 5.0+. Допустимые значения: <code>dataready</code> и <code>httpready</code> .
<code>deferred</code>	указывает использовать отложенный <code>accept()</code> (параметр <code>TCP_DEFER_ACCEPT</code>) на Linux.
<code>bind</code>	указывает, что для данного слушающего сокета нужно делать <code>bind()</code> отдельно. Это нужно потому, что если описаны несколько директив <code>listen</code> с одинаковым портом, но разными адресами, и одна из директив <code>listen</code> слушает на всех адресах для данного <i>порта</i> (<code>*:порт</code>), то Angie сделает <code>bind()</code> только на <code>*:порт</code> . Необходимо заметить, что в этом случае для определения адреса, на который пришло соединение, делается системный вызов <code>getsockname()</code> . Если же используются параметры <code>setfib</code> , <code>fastopen</code> , <code>backlog</code> , <code>rcvbuf</code> , <code>sndbuf</code> , <code>accept_filter</code> , <code>deferred</code> , <code>ipv6only</code> , <code>reuseport</code> , <code>multipath</code> или <code>so_keepalive</code> , то для данной пары <i>адрес:порт</i> всегда делается отдельный вызов <code>bind()</code> .
<code>ipv6only=on off</code>	определяет (через параметр сокета <code>IPV6_V6ONLY</code>), будет ли слушающий на wildcard-адресе <code>::</code> IPv6-сокеты принимать только IPv6-соединения, или же одновременно IPv6- и IPv4-соединения. По умолчанию параметр включен. Установить его можно только один раз на старте.
<code>reuseport</code>	указывает, что нужно создавать отдельный слушающий сокет для каждого рабочего процесса (через параметр сокета <code>SO_REUSEPORT</code> для Linux 3.9+ и DragonFly BSD или <code>SO_REUSEPORT_LB</code> для FreeBSD 12+), позволяя ядру распределять входящие соединения между рабочими процессами. В настоящий момент это работает только на Linux 3.9+, DragonFly BSD и FreeBSD 12+.
 Предупреждение Ненадлежащее использование параметра <code>reuseport</code> может быть небезопасно.	
<code>multipath</code>	включает прием соединений по протоколу Multipath TCP (MPTCP), поддерживаемому в ядре Linux с версии 5.6.

`so_keepalive=on | off | [keepidle]:[keepintvl]:[keepcnt]`

Конфигурирует для слушающего сокета поведение "TCP keepalive".

<code>''</code>	если параметр опущен, для сокета будут действовать настройки операционной системы
<code>on</code>	для сокета включается параметр <code>SO_KEEPALIVE</code>
<code>off</code>	для сокета параметр <code>SO_KEEPALIVE</code> выключается

Некоторые операционные системы поддерживают настройку параметров "TCP keepalive" на уровне сокета посредством параметров `TCP_KEEPIDLE`, `TCP_KEEPINTVL` и `TCP_KEEPCNT`. На таких системах их можно сконфигурировать с помощью параметров `keepidle`, `keepintvl` и `keepcnt`. Один или два параметра могут быть опущены, в таком случае для соответствующего параметра сокета будут действовать стандартные системные настройки.

Например,

```
so_keepalive=30m:10
```

установит таймаут бездействия (`TCP_KEEPIDLE`) в 30 минут, для интервала проб (`TCP_KEEPINTVL`) будет действовать стандартная системная настройка, а счетчик проб (`TCP_KEEPCNT`) будет равен 10.

Дополнительные параметры привязывают слушающий сокет к сетевому интерфейсу и задают для этого сокета алгоритм управления перегрузкой TCP:

<code>interface=имя</code>	привязывает слушающий сокет к указанному сетевому интерфейсу (параметр сокета <code>SO_BINDTODEVICE</code>)
<code>congestion=алгор</code>	задает алгоритм управления перегрузкой TCP (параметр сокета <code>TCP_CONGESTION</code>) для слушающего сокета; принимаемые соединения наследуют этот алгоритм. Параметр несовместим с <code>udp</code> и с UNIX-сокетами

При использовании любого из этих параметров для данной пары *адрес:порт* всегда выполняется отдельный вызов `bind()`.

Имя алгоритма управления перегрузкой не проверяется на этапе конфигурации: если алгоритм неизвестен или недоступен, при настройке слушающего сокета в лог однократно записывается сообщение уровня `alert`, а для сокета действует системный алгоритм по умолчанию.

preread_buffer_size

Синтаксис	<code>preread_buffer_size</code> размер;
По умолчанию	<code>preread_buffer_size 16k</code> ;
Контекст	<code>stream, server</code>

Задаёт размер буфера предварительного чтения.

preread_timeout

Синтаксис	<code>preread_timeout</code> время;
По умолчанию	<code>preread_timeout 30s</code> ;
Контекст	<code>stream, server</code>

Задаёт время фазы предварительного чтения.

proxy_protocol_timeout

Синтаксис	<code>proxy_protocol_timeout время;</code>
По умолчанию	<code>proxy_protocol_timeout 30s;</code>
Контекст	stream, server

Задаёт время для завершения операции чтения заголовка протокола PROXY. Если по истечении этого времени заголовок полностью не получен, соединение закрывается.

resolver

Синтаксис	<code>resolver [адрес ...] [conf [=файл]] [valid=время] [ipv4=on off] [ipv6=on off] [status_zone=зона];</code> <code>resolver off;</code>
По умолчанию	<code>resolver conf;</code>
Контекст	stream, server, upstream

Задаёт серверы DNS, используемые для преобразования имен вышестоящих серверов в адреса, например:

```
resolver 127.0.0.53 [::1]:5353;
```

Адрес может быть указан в виде доменного имени или IP-адреса, и необязательного порта. Если порт не указан, используется порт 53. Серверы DNS опрашиваются циклически.

Необходимо указать хотя бы один адрес DNS-сервера или параметр `conf`, кроме формы `resolver off;`.

Примечание

Рекомендуется использовать локальный доверенный резолвер, например 127.0.0.53 (systemd-resolved), а не публичный (например, 8.8.8.8). Публичные резолверы раскрывают DNS-запросы третьим сторонам и повышают риск атак с подменой кэша.

Примечание

Значение директивы наследуется вложенными блоками и может быть переопределено в них при необходимости. В пределах одного блока директиву допустимо указывать только один раз; повторное указание является ошибкой конфигурации.

Включает разрешение DNS-имен для вышестоящих серверов, задаваемых через переменные или настроенных с параметром `resolve`, а также для прикрепления OCSP-ответов.

Помимо явно заданных адресов серверов директива распознаёт:

<code>off</code>	явно отключает разрешение DNS-имен
<code>conf [=файл]</code>	считывает адреса DNS-серверов из указанного файла или из <code>/etc/resolv.conf</code> , если файл не указан

Файл может содержать строки `nameserver` в том же формате, что использует системный резолвер:

```
nameserver 127.0.1.4
nameserver 127.0.1.5
```

Адреса, считанные из файла, добавляются к адресам, заданным явно в директиве, и все они затем опрашиваются совместно циклическим образом. Если файл отсутствует, недоступен для чтения или не содержит адресов, а явно ни один сервер не задан, используется сервер по умолчанию 127.0.0.1:53.

В процессе работы Angie перечитывает файл при его изменении, обновляя набор DNS-серверов.

По умолчанию Angie кэширует каждый ответ на время, заданное значением TTL в DNS-ответе. Необязательный параметр `valid` позволяет это переопределить:

valid	<i>необязательный</i> параметр, позволяет переопределить срок кэширования ответа
--------------	--

```
resolver 127.0.0.53 [::1]:5353 valid=30s;
```

По умолчанию Angie будет искать как IPv4-, так и IPv6-адреса при преобразовании имен в адреса.

ipv4=off	запрещает поиск IPv4-адресов
ipv6=off	запрещает поиск IPv6-адресов

status_zone	<i>необязательный</i> параметр; включает сбор метрик запросов и ответов DNS-сервера (<code>/status/resolvers/<зона></code>) в указанной зоне
--------------------	--

Совет

Для предотвращения DNS-спуфинга рекомендуется использовать DNS-серверы в защищенной доверенной локальной сети.

Совет

При запуске в Docker используйте соответствующий внутренний адрес DNS-сервера, например 127.0.0.11.

resolver_timeout

Синтаксис	<code>resolver_timeout время;</code>
По умолчанию	<code>resolver_timeout 30s;</code>
Контекст	<code>stream, server, upstream</code>

Задаёт таймаут для преобразования имени в адрес, например:

```
resolver_timeout 5s;
```

server

Синтаксис	server { ... }
По умолчанию	—
Контекст	stream

Задаёт конфигурацию для сервера.

server_name

Синтаксис	server_name <i>имя</i> ...;
По умолчанию	server_name "";
Контекст	server

Задаёт имена виртуального сервера.

Предупреждение

В модуле **stream** директива **server_name** основана на Server Name Indication (SNI) и работает только с TLS-соединениями. Для использования необходимо настроить TLS-терминацию или включить TLS pread в соответствующем блоке **server**.

Пример конфигурации:

```
server {
    listen 443 ssl;
    server_name example.com www.example.com;
    ssl_certificate /etc/angie/cert.pem;
    ssl_certificate_key /etc/angie/key.pem;
}
```

Первое указанное имя становится основным именем сервера.

Имена серверов могут включать звездочку (*), заменяющую первую или последнюю часть имени:

```
server {
    server_name example.com *.example.com www.example.*;
}
```

Такие имена называются подстановочными (wildcard).

Также можно использовать регулярные выражения в именах серверов, предваряя имя тильдой (~):

```
server {
    server_name www.example.com ~^www\d+\.example\.com$;
}
```

Регулярные выражения могут включать захватываемые группы, которые можно использовать в других директивах:

```
server {
    server_name ~^(www\.)?(.+)$;
```

```
proxy_pass www.$2:12345;
}
```

Именованные захваты в регулярных выражениях создают переменные, которые можно использовать в других директивах:

```
server {
    server_name ~(www\.)?(?<domain>.+)$;

    proxy_pass www.$domain:12345;
}
```

Если параметр директивы установлен на `$hostname`, будет вставлено имя хоста машины.

При поиске виртуального сервера по имени, если имя совпадает с более чем одним из указанных вариантов (например, совпадают и шаблонное имя, и регулярное выражение), будет выбрано первое совпавшее имя в следующем порядке приоритета:

- Точное имя
- Самое длинное шаблонное имя, начинающееся с звездочки, например, `*.example.com`
- Самое длинное шаблонное имя, заканчивающееся звездочкой, например, `mail.*`
- Первое совпавшее регулярное выражение (в порядке появления в конфигурационном файле)

`server_names_hash_bucket_size`

Синтаксис	<code>server_names_hash_bucket_size</code> размер;
По умолчанию	<code>server_names_hash_bucket_size</code> 32 64 128;
Контекст	stream

Задаёт размер корзины для хэш-таблиц имен серверов. Значение по умолчанию зависит от размера кэш-линии процессора.

`server_names_hash_max_size`

Синтаксис	<code>server_names_hash_max_size</code> размер;
По умолчанию	<code>server_names_hash_max_size</code> 512;
Контекст	stream

Задаёт максимальный размер хэш-таблиц имен серверов.

`status_zone`

Синтаксис	<code>status_zone зона ключ zone=зона[:число];</code>
По умолчанию	—
Контекст	<code>server</code>

Выделяет зону разделяемой памяти для сбора метрик `/status/stream/server_zones/<зона>`.

Несколько контекстов `server` могут совместно использовать одну и ту же зону для сбора данных.

Синтаксис с одним значением *зоны* объединяет все метрики для текущего контекста в одну зону разделяемой памяти:

```
server {  
  
    listen 80;  
    server_name *.example.com;  
  
    status_zone single;  
    # ...  
}
```

Альтернативный синтаксис позволяет задавать следующие параметры:

<i>значение</i>	Строка с переменными, значение которой определяет группировку подключений в зоне. Все подключения, дающие одинаковые значения после подстановки, объединяются в одну группу. Если подстановка возвращает пустое значение, метрики не обновляются.
<i>зона</i>	Имя зоны разделяемой памяти.
<i>число</i> (необязательный)	Максимальное количество отдельных групп для сбора метрик. Если новые значения <i>ключа</i> превышают этот лимит, они объединяются в группу <i>zone</i> . Значение по умолчанию — 1.

В следующем примере все соединения с одинаковым значением `$server_addr` группируются в `host_zone`. Метрики собираются отдельно для каждого уникального значения `$server_addr` до тех пор, пока количество групп метрик не достигнет 10. После этого любые новые значения `$server_addr` будут добавляться в группу `server_zone`:

```
stream {  
  
    upstream backend {  
        server 192.168.0.1:3306;  
        server 192.168.0.2:3306;  
        # ...  
    }  
  
    server {  
  
        listen 3306;  
        proxy_pass backend;  
  
        status_zone $server_addr zone=server_zone:10;  
    }  
}
```

Результирующие метрики разделяются по отдельным серверам в выводе API.

Примечание

Эти метрики собираются, только если задан `status_zone`. Без него сервер не отображается в `/status/stream/server_zones/<зона>`, в виджете «TCP/UDP Zones» и в выводе Prometheus, и предупреждение при этом не выводится.

stream

Синтаксис	<code>stream { ... }</code>
По умолчанию	—
Контекст	main

Предоставляет контекст конфигурационного файла, в котором указываются директивы stream-сервера.

tcp_nodelay

Синтаксис	<code>tcp_nodelay on off;</code>
По умолчанию	<code>tcp_nodelay on;</code>
Контекст	stream, server

Разрешает или запрещает использование параметра TCP_NODELAY. Параметр включается как для клиентских соединений, так и для соединений с проксируемыми серверами.

time_format

Синтаксис	<code>time_format \$переменная формат;</code>
По умолчанию	—
Контекст	stream

Определяет *\$переменную*, содержащую текущее локальное время, отформатированное в соответствии со строкой *формата*, которая поддерживает спецификаторы в стиле `strftime`, а также `%L` для миллисекунд. Директива полезна, когда возможностей `$time_local` и `$time_iso8601` недостаточно — например, если требуется точность до миллисекунд:

```
time_format $time_ms "%Y-%m-%dT%H:%M:%S.%L%Z";
log_format tf '$time_ms $remote_addr $status';
```

При таком формате переменная `$time_ms` принимает значение вида `2024-01-15T14:34:56.789+03:00`.

Попытка переопределить встроенную переменную, например `$time_local`, или повторно объявить `time_format` с тем же именем приводит к ошибке загрузки конфигурации вида: "the duplicate "... variable".

Строка *формата* поддерживает следующие спецификаторы; все они вычисляются в том же часовом поясе, что и `$time_local`:

%Y	год, 4 цифры
%y	год, 2 цифры
%m	месяц, 01-12
%d	день месяца, 01-31
%e	день месяца, дополненный пробелом, 1-31
%H	час, 00-23
%I	час, 01-12
%M	минута, 00-59
%S	секунда, 00-59
%L	миллисекунды, 000-999
%p	обозначение AM/PM в верхнем регистре, AM / PM
%P	то же самое в нижнем регистре, am / pm
%a	сокращенное название дня недели, Sun-Sat
%A	полное название дня недели, Sunday-Saturday
%b, %h	сокращенное название месяца, Jan-Dec
%B	полное название месяца, January-December
%z	смещение часового пояса, +0300
%Z	смещение часового пояса в формате ISO 8601, +03:00
%n	перевод строки
%t	табуляция
%%	символ %

Примечание

Нераспознанный спецификатор копируется в вывод без изменений, включая ведущий символ %; одиночный символ % в самом конце строки формата отбрасывается.

variables_hash_bucket_size

Синтаксис	variables_hash_bucket_size размер;
По умолчанию	variables_hash_bucket_size 64;
Контекст	stream

Задаёт размер корзины в хэш-таблице переменных. Подробнее настройка хэш-таблиц обсуждается отдельно.

variables_hash_max_size

Синтаксис	variables_hash_max_size размер;
По умолчанию	variables_hash_max_size 1024;
Контекст	stream

Задаёт максимальный размер хэш-таблиц переменных. Подробнее настройка хэш-таблиц обсуждается отдельно.

Встроенные переменные

Базовый потоковый модуль поддерживает встроенные переменные:

`$angie_version`

версия Angie

`$binary_remote_addr`

адрес клиента в бинарном виде, длина значения всегда 4 байта для IPv4-адресов или 16 байт для IPv6-адресов

`$bytes_received`

число байт, полученных от клиента

`$bytes_sent`

число байт, переданных клиенту

`$connection`

порядковый номер соединения

`$hostname`

имя хоста

`$msec`

текущее время в секундах с точностью до миллисекунд

`$nginx_version`

версия nginx

`$pid`

номер (PID) рабочего процесса

`$protocol`

протокол, используемый для работы с клиентом: *TCP* или *UDP*

`$proxy_protocol_addr`

адрес клиента, полученный из заголовка протокола PROXY. Протокол PROXY должен быть предварительно включен при помощи установки параметра `proxy_protocol` в директиве *listen*.

`$proxy_protocol_port`

порт клиента, полученный из заголовка протокола PROXY. Протокол PROXY должен быть предварительно включен при помощи установки параметра `proxy_protocol` в директиве *listen*.

`$proxy_protocol_server_addr`

адрес сервера, полученный из заголовка протокола PROXY. Протокол PROXY должен быть предварительно включен при помощи установки параметра `proxy_protocol` в директиве *listen*.

`$proxy_protocol_server_port`

порт сервера, полученный из заголовка протокола PROXY. Протокол PROXY должен быть предварительно включен при помощи установки параметра `proxy_protocol` в директиве *listen*.

`$proxy_protocol_tlv_<имя>`

TLV, полученный из заголовка протокола PROXY. *Имя* может быть именем типа TLV или его числовым значением. В последнем случае значение задается в шестнадцатеричном виде и должно начинаться с *0x*:

```
$proxy_protocol_tlv_alpn
$proxy_protocol_tlv_0x01
```

SSL TLV могут также быть доступны как по имени типа TLV, так и по его числовому значению, оба должны начинаться с `ssl_`:

```
$proxy_protocol_tlv_ssl_version
$proxy_protocol_tlv_ssl_0x21
```

Поддерживаются следующие имена типов TLV:

- `alpn` (0x01) - протокол более высокого уровня, используемый поверх соединения
- `authority` (0x02) - значение имени хоста, передаваемое клиентом
- `unique_id` (0x05) - уникальный идентификатор соединения
- `netns` (0x30) - имя пространства имен
- `ssl` (0x20) - структура SSL TLV в бинарном виде

Поддерживаются следующие имена типов SSL TLV:

- `ssl_version` (0x21) - версия SSL, используемая в клиентском соединении
- `ssl_cn` (0x22) - Common Name сертификата
- `ssl_cipher` (0x23) - имя используемого шифра
- `ssl_sig_alg` (0x24) - алгоритм, используемый для подписи сертификата
- `ssl_key_alg` (0x25) - алгоритм публичного ключа

Также поддерживается следующее специальное имя типа SSL TLV:

- `ssl_verify` - результат проверки клиентского сертификата: 0, если клиент предоставил сертификат и он был успешно верифицирован, либо ненулевое значение

Протокол PROXY должен быть предварительно включен при помощи установки параметра `proxy_protocol` в директиве *listen*.

`$remote_addr`

адрес клиента

`$remote_port`

порт клиента

`$server_addr`

адрес сервера, принявшего соединение. Получение значения этой переменной обычно требует одного системного вызова. Чтобы избежать системного вызова, в директивах *listen* следует указывать адреса и использовать параметр *bind*.

`$server_port`

порт сервера, принявшего соединение

`$session_time`

длительность сессии в секундах с точностью до миллисекунд

`$status`

статус сессии, может принимать одно из следующих значений:

200	сессия завершена успешно
400	невозможно разобрать данные, полученные от клиента, например заголовок протокола PROXY
403	доступ запрещен, например при ограничении доступа для определенных адресов клиентов
500	внутренняя ошибка сервера
502	плохой шлюз, например если невозможно выбрать сервер группы или сервер недоступен
503	сервис недоступен, например при ограничении по числу соединений

`$time_iso8601`

локальное время в формате по стандарту ISO 8601

`$time_local`

локальное время в Common Log Format

4.10.8.6 Почтовые модули

<i>Mail</i>	Основная функциональность почтового прокси-сервера.
<i>Auth</i>	Аутентификация пользователей и выбор сервера для последующего проксирования с помощью HTTP-запросов к внешнему серверу.
<i>HTTP</i>	Поддержка протокола HTTP.
<i>IMAP</i>	Поддержка протокола IMAP.
<i>POP3</i>	Поддержка протокола POP3.
<i>Proxy</i>	Настройка проксирования к другим серверам.
<i>RealIP</i>	Определение адреса и порта клиента при работе за другим прокси-сервером.
<i>SMTP</i>	Поддержка протокола SMTP.
<i>SSL</i>	Поддержка протоколов SSL/TLS и StartTLS.

Почтовый модуль

Базовый почтовый модуль реализует основную функциональность почтового прокси-сервера: это поддержка протоколов SMTP, IMAP и POP3, настройка серверных блоков, маршрутизация почтовых запросов, аутентификация пользователей и поддержка SSL/TLS для защиты почтовых соединений.

Остальные модули этого раздела расширяют эту функциональность, позволяя гибко настраивать и оптимизировать работу почтового сервера под различные сценарии и требования.

В образах из наших репозиториях модуль уже включен в сборку.

Пример конфигурации

```
worker_processes auto;

error_log /var/log/angie/error.log info;

events {
    worker_connections 1024;
}

mail {
    server_name      mail.example.com;
    auth_http        localhost:9000/cgi-bin/auth.cgi;

    imap_capabilities IMAP4rev1 UIDPLUS IDLE LITERAL+ QUOTA;

    pop3_auth        plain apop cram-md5;
    pop3_capabilities LAST TOP USER PIPELINING UIDL;

    smtp_auth        login plain cram-md5;
    smtp_capabilities "SIZE 10485760" ENHANCEDSTATUSCODES 8BITIME DSN;
    xclient          off;

    server {
        listen 25;
        protocol smtp;
    }
    server {
        listen 110;
        protocol pop3;
        proxy_pass_error_message on;
    }
    server {
        listen 143;
        protocol imap;
    }
    server {
        listen 587;
        protocol smtp;
    }
}
```

Директивы

listen

<i>Синтаксис</i>	<code>listen</code> <i>адрес[:порт]</i> [<i>ssl</i>] [<i>proxy_protocol</i>] [<i>backlog=число</i>] [<i>rcvbuf=размер</i>] [<i>sndbuf=размер</i>] [<i>bind</i>] [<i>ipv6only=on off</i>] [<i>reuseport</i>] [<i>so_keepalive=on off</i>][<i>keepidle</i>]:[<i>keepintvl</i>]:[<i>keepcnt</i>];
По умолчанию	—
<i>Контекст</i>	server

Задаёт *адрес* и *порт* для сокета, на котором сервер будет принимать соединения. Можно указать только *порт*, и тогда Angie будет слушать на всех доступных IPv4-интерфейсах (и IPv6, если он включен). Кроме того, *адрес* может быть именем хоста, например:

```
listen 127.0.0.1:110;
listen *:110;
listen 110;      # то же, что и *:110
listen localhost:110;
```

IPv6-адреса задаются в квадратных скобках:

```
listen [::1]:110;
listen [::]:110;
```

UNIX-сокеты задаются префиксом `unix:`

```
listen unix:/var/run/angie.sock;
```

Примечание

Разные серверы должны слушать на разных парах *адрес:порт*.

ssl	указывает на то, что все соединения, принимаемые на данном слушающем сокете, должны работать в режиме SSL.
proxy_protocol	указывает на то, что все соединения, принимаемые на данном порту, должны использовать протокол PROXY. Полученная информация передается <i>серверу аутентификации</i> и может быть использована для <i>изменения адреса клиента</i> .

В директиве `listen` можно также указать несколько дополнительных параметров, специфичных для связанных с сокетами системных вызовов.

backlog=число	задает параметр <i>backlog</i> в вызове <i>listen()</i> , который ограничивает максимальный размер очереди ожидающих приема соединений. По умолчанию <i>backlog</i> устанавливается равным -1 для FreeBSD, DragonFly BSD и macOS, и 511 для других платформ.
rcvbuf=размер	задает размер буфера приема (параметр <i>SO_RCVBUF</i>) для слушающего сокета.
sndbuf=размер	задает размер буфера передачи (параметр <i>SO_SNDBUF</i>) для слушающего сокета.
bind	указывает, что для данного слушающего сокета нужно делать <i>bind()</i> отдельно. Это нужно потому, что если описаны несколько директив <i>listen</i> с одинаковым портом, но разными адресами, и одна из директив <i>listen</i> слушает на всех адресах для данного <i>порта</i> (<i>*:порт</i>), то Angie сделает <i>bind()</i> только на <i>*:порт</i> . Необходимо заметить, что в этом случае для определения адреса, на который пришло соединение, делается системный вызов <i>getsockname()</i> . Если же используются параметры <i>backlog</i> , <i>rcvbuf</i> , <i>sndbuf</i> , <i>ipv6only</i> , <i>reuseport</i> или <i>so_keepalive</i> , то для данной пары <i>адрес:порт</i> всегда делается отдельный вызов <i>bind()</i> .
ipv6only=on off	определяет (через параметр сокета <i>IPV6_V6ONLY</i>), будет ли слушающий на wildcard-адресе <i>[:]</i> IPv6-сокеты принимать только IPv6-соединения, или же одновременно IPv6- и IPv4-соединения. По умолчанию параметр включен. Установить его можно только один раз на старте.
multipath	включает прием соединений по протоколу Multipath TCP (MPTCP), поддерживаемому в ядре Linux с версии 5.6.

so_keepalive=on | off | [keepidle]:[keepintvl]:[keepcnt]

Конфигурирует для слушающего сокета поведение "TCP keepalive".

''	если параметр опущен, для сокета будут действовать настройки операционной системы
on	для сокета включается параметр <i>SO_KEEPALIVE</i>
off	для сокета параметр <i>SO_KEEPALIVE</i> выключается

Некоторые операционные системы поддерживают настройку параметров "TCP keepalive" на уровне сокета посредством параметров *TCP_KEEPIDLE*, *TCP_KEEPINTVL* и *TCP_KEEPCNT*. На таких системах их можно сконфигурировать с помощью параметров *keepidle*, *keepintvl* и *keepcnt*. Один или два параметра могут быть опущены, в таком случае для соответствующего параметра сокета будут действовать стандартные системные настройки.

Например,

```
so_keepalive=30m:10
```

установит таймаут бездействия (*TCP_KEEPIDLE*) в 30 минут, для интервала проб (*TCP_KEEPINTVL*) будет действовать стандартная системная настройка, а счетчик проб (*TCP_KEEPCNT*) будет равен 10.

Дополнительные параметры привязывают слушающий сокет к сетевому интерфейсу и задают для этого сокета алгоритм управления перегрузкой TCP:

interface=имя	привязывает слушающий сокет к указанному сетевому интерфейсу (параметр сокета <i>SO_BINDTODEVICE</i>)
congestion=алгор	задает алгоритм управления перегрузкой TCP (параметр сокета <i>TCP_CONGESTION</i>) для слушающего сокета; принимаемые соединения наследуют этот алгоритм. Параметр несовместим с UNIX-сокетами

При использовании любого из этих параметров для данной пары *адрес:порт* всегда выполняется отдельный вызов *bind()*.

Имя алгоритма управления перегрузкой не проверяется на этапе конфигурации: если алгоритм неизвестен или недоступен, при настройке слушающего сокета в лог однократно записывается сообщение уровня **alert**, а для сокета действует системный алгоритм по умолчанию.

Разные серверы должны слушать на разных парах *адрес:порт*.

mail

<i>Синтаксис</i>	mail { ... }
По умолчанию	—
<i>Контекст</i>	main

Предоставляет контекст конфигурационного файла, в котором указываются директивы почтового сервера.

max_commands

<i>Синтаксис</i>	max_commands <i>число</i> ;
По умолчанию	max_commands 1000;
<i>Контекст</i>	mail, server

Задаёт максимальное количество команд, отправляемых во время аутентификации, для усиления защиты от DoS-атак.

max_errors

<i>Синтаксис</i>	max_errors <i>число</i> ;
По умолчанию	max_errors 5;
<i>Контекст</i>	mail, server

Задаёт число ошибок протокола, по достижении которого соединение закрывается.

protocol

<i>Синтаксис</i>	protocol <i>imap</i> <i>pop3</i> <i>smtp</i> ;
По умолчанию	—
<i>Контекст</i>	server

Задаёт протокол проксируемого сервера. Поддерживаются протоколы *IMAP*, *POP3* и *SMTP*.

Если директива не указана, то протокол может быть определен автоматически по общеизвестному порту, указанному в директиве **listen**:

```
imap: 143, 993
pop3: 110, 995
smtp: 25, 587, 465
```

resolver

<i>Синтаксис</i>	<code>resolver адрес ... [valid=время] [ipv4=on off] [ipv6=on off] [status_zone=зона] off;</code>
По умолчанию	<code>resolver off;</code>
<i>Контекст</i>	mail, server

Задаёт серверы DNS, используемые для преобразования имени хоста клиента для передачи его на сервер аутентификации и в команде *XCLIENT* при проксировании SMTP, например:

```
resolver 127.0.0.53 [::1]:5353;
```

Специальное значение **off** отключает преобразование имени хоста клиента и отменяет унаследованное значение директивы.

Адрес может быть указан в виде доменного имени или IP-адреса, и необязательного порта. Если порт не указан, используется порт 53. Серверы DNS опрашиваются циклически.

Примечание

Рекомендуется использовать локальный доверенный резолвер, например 127.0.0.53 (systemd-resolved), а не публичный (например, 8.8.8.8). Публичные резолверы раскрывают DNS-запросы третьим сторонам и повышают риск атак с подменой кэша.

Примечание

Значение директивы наследуется вложенными блоками и может быть переопределено в них при необходимости. В пределах одного блока директиву допустимо указывать только один раз; повторное указание является ошибкой конфигурации.

При поиске имени хоста клиента Angie кэширует каждый DNS-ответ на время, заданное его TTL. Необязательный параметр **valid** позволяет переопределить этот срок:

valid	необязательный параметр, позволяет переопределить срок кэширования ответа
--------------	---

```
resolver 127.0.0.53 [::1]:5353 valid=30s;
```

По умолчанию Angie будет искать как IPv4-, так и IPv6-адреса при преобразовании имен в адреса.

ipv4=off	запрещает поиск IPv4-адресов
ipv6=off	запрещает поиск IPv6-адресов
status_zone	необязательный параметр, включает сбор информации о запросах и ответах сервера DNS в указанной зоне

Совет

Для предотвращения DNS-спуфинга рекомендуется использовать DNS-серверы в защищенной доверенной локальной сети.

resolver_timeout

<i>Синтаксис</i>	<code>resolver_timeout время;</code>
По умолчанию	<code>resolver_timeout 30s;</code>
<i>Контекст</i>	mail, server

Задаёт таймаут для преобразования имени в адрес, например:

```
resolver_timeout 5s;
```

server

<i>Синтаксис</i>	<code>server { ... }</code>
По умолчанию	—
<i>Контекст</i>	mail

Задаёт конфигурацию для сервера.

server_name

<i>Синтаксис</i>	<code>server_name имя;</code>
По умолчанию	<code>server_name hostname;</code>
<i>Контекст</i>	mail, server

Задаёт имя сервера, используемое:

- в начальном приветствии POP3/SMTP-сервера;
- в salt при аутентификации SASL-методом CRAM-MD5;
- в команде EHLO при подключении к SMTP-бэкенду, если разрешена передача команды *XCLIENT*.

Если директива не указана, используется имя хоста (*hostname*) машины.

timeout

<i>Синтаксис</i>	<code>timeout время;</code>
По умолчанию	<code>timeout 60s;</code>
<i>Контекст</i>	mail, server

Задаёт таймаут, который используется до начала проксирования на бэкенд.

Auth HTTP

Модуль позволяет выполнять аутентификацию на основе дополнительного HTTP-запроса перед обработкой основного запроса. Если подзапрос возвращает статус 2xx, основной запрос продолжается; если возвращается 401 или 403, пользователю отправляется соответствующая ошибка, а при любом другом статусе возвращается ошибка 500. Такой подход обычно используется для передачи аутентификации внешним сервисам, объединения аутентификации в разных приложениях или интеграции со сторонними системами, такими как OAuth или LDAP.

Директивы

auth_http

Синтаксис	<code>auth_http uri;</code>
По умолчанию	—
Контекст	mail, server

Задаёт URL HTTP-сервера аутентификации. Протокол описан ниже.

auth_http_header

Синтаксис	<code>auth_http_header <i>заголовок значение</i>;</code>
По умолчанию	—
Контекст	mail, server

Добавляет указанный заголовок к запросам, посылаемым на сервер аутентификации. Заголовок можно использовать в качестве shared secret для проверки, что запрос поступил от Angie. Например:

```
auth_http_header X-Auth-Key "secret_string";
```

auth_http_pass_client_cert

Синтаксис	<code>auth_http_pass_client_cert on off;</code>
По умолчанию	<code>auth_http_pass_client_cert off;</code>
Контекст	mail, server

Добавляет заголовок **Auth-SSL-Cert** с клиентским сертификатом в формате PEM (закодирован в формате *urlencode*) к запросам, посылаемым на сервер аутентификации.

auth_http_timeout

Синтаксис	<code>auth_http_timeout время;</code>
По умолчанию	<code>auth_http_timeout 60s;</code>
Контекст	mail, server

Задаёт таймаут общения с сервером аутентификации.

Протокол

Для общения с сервером аутентификации используется протокол HTTP. Данные в теле ответа игнорируются, информация передается только в заголовках.

Примеры запросов и ответов:

Запрос:

```
GET /auth HTTP/1.0
Host: localhost
Auth-Method: plain # plain/apop/cram-md5/external/soauth2/oauthbearer/none
Auth-User: user
Auth-Pass: password
Auth-Protocol: imap # imap/pop3/smtp
Auth-Login-Attempt: 1
Client-IP: 192.0.2.42
Client-Host: client.example.org
```

Хороший ответ:

```
HTTP/1.0 200 OK
Auth-Status: OK
Auth-Server: 198.51.100.1
Auth-Port: 143
```

Плохой ответ:

```
HTTP/1.0 200 OK
Auth-Status: Invalid login or password
Auth-Wait: 3
```

Если заголовок **Auth-Wait** нет, то после выдачи ошибки соединение будет закрыто. В текущей реализации на каждую попытку аутентификации выделяется память, которая освобождается только при завершении сессии. Поэтому число неудачных попыток аутентификации в рамках одной сессии должно быть ограничено — после 10-20 попыток (номер попытки передается в заголовке **Auth-Login-Attempt**) сервер должен выдать ответ без заголовка **Auth-Wait**.

При использовании APOP или CRAM-MD5 запрос и ответ будут выглядеть так:

```
GET /auth HTTP/1.0
Host: localhost
Auth-Method: apop
Auth-User: user
Auth-Salt: <238188073.1163692009@mail.example.com>
Auth-Pass: auth_response
Auth-Protocol: imap
Auth-Login-Attempt: 1
Client-IP: 192.0.2.42
Client-Host: client.example.org
```

Хороший ответ:

```
HTTP/1.0 200 OK
Auth-Status: OK
Auth-Server: 198.51.100.1
Auth-Port: 143
Auth-Pass: plain-text-pass
```

При использовании XOAUTH2 или OAUTHBEARER заголовки **Auth-User** и **Auth-Pass** содержат имя пользователя и токен bearer, извлеченные из начального SASL-ответа.

Если в ответе есть заголовок **Auth-User**, то он переопределяет имя пользователя, используемое для аутентификации с бэкендом.

Для SMTP в ответе дополнительно учитывается заголовок **Auth-Error-Code** — если он есть, то используется как код ответа в случае ошибки. Если его нет, то по умолчанию к **Auth-Status** будет добавлен код *535 5.7.0*.

Для XOAUTH2 и OAUTHBEARER в ответе с ошибкой также может присутствовать заголовок **Auth-Error-SASL**. Его значение отправляется клиенту как дополнительный SASL-вызов (SMTP: *334*, IMAP/POP3: *+*). После ответа клиента (пустой ответ для XOAUTH2 или **AQ==** для OAUTHBEARER) будет возвращена ошибка из **Auth-Status**.

Например, если от сервера аутентификации будет получен ответ:

```
HTTP/1.0 200 OK
Auth-Status: Temporary server problem, try again later
Auth-Error-Code: 451 4.3.0
Auth-Wait: 3
```

то по SMTP клиенту будет выдана ошибка

```
451 4.3.0 Temporary server problem, try again later
```

Если при проксировании SMTP не требуется аутентификация, запрос будет выглядеть так:

```
GET /auth HTTP/1.0
Host: localhost
Auth-Method: none
Auth-User:
Auth-Pass:
Auth-Protocol: smtp
Auth-Login-Attempt: 1
Client-IP: 192.0.2.42
Client-Host: client.example.org
Auth-SMTP-Helo: client.example.org
Auth-SMTP-From: MAIL FROM: <>
Auth-SMTP-To: RCPT TO: <postmaster@mail.example.com>
```

Для клиентского соединения по протоколу SSL/TLS добавляется заголовок **Auth-SSL**, и если директива `ssl_verify_client` включена, заголовок **Auth-SSL-Verify** содержит результат проверки клиентского сертификата: **SUCCESS**, **FAILED:reason** и, если сертификат не был предоставлен, **NONE**.

Если клиентский сертификат был предоставлен, информация о нем передается в следующих заголовках запроса: **Auth-SSL-Subject**, **Auth-SSL-Issuer**, **Auth-SSL-Serial** и **Auth-SSL-Fingerprint**. Если директива `auth_http_pass_client_cert` включена, сам сертификат передается в заголовке **Auth-SSL-Cert**. Протокол и шифр установленного соединения передаются в заголовках **Auth-SSL-Protocol** и **Auth-SSL-Cipher**. Запрос будет выглядеть так:

```
GET /auth HTTP/1.0
Host: localhost
Auth-Method: plain
Auth-User: user
Auth-Pass: password
Auth-Protocol: imap
Auth-Login-Attempt: 1
Client-IP: 192.0.2.42
Auth-SSL: on
Auth-SSL-Protocol: TLSv1.3
Auth-SSL-Cipher: TLS_AES_256_GCM_SHA384
Auth-SSL-Verify: SUCCESS
Auth-SSL-Subject: /CN=example.com
Auth-SSL-Issuer: /CN=example.com
Auth-SSL-Serial: C07AD56B846B5BFF
```

Auth-SSL-Fingerprint: 29d6a80a123d13355ed16b4b04605e29cb55a5ad

При использовании протокола PROXY, информация о нем передается в следующих заголовках запроса: Proxy-Protocol-Addr, Proxy-Protocol-Port, Proxy-Protocol-Server-Addr и Proxy-Protocol-Server-Port.

IMAP

Модуль обеспечивает поддержку почтового протокола IMAP, позволяя серверу взаимодействовать с системами хранения почты. Он устанавливает соединения с серверами IMAP, обрабатывает основные команды, такие как список папок и получение сообщений, а также обеспечивает безопасную аутентификацию и управление статусами сообщений.

Директивы

imap_auth

Синтаксис	imap_auth <i>метод</i> ...;
По умолчанию	imap_auth plain;
Контекст	mail, server

Задаёт разрешенные методы аутентификации IMAP-клиентов. Поддерживаемые методы:

plain	LOGIN, AUTH=PLAIN
login	AUTH=LOGIN
cram-md5	AUTH=CRAM-MD5. Для работы этого метода пароль должен храниться в незашифрованном виде.
external	AUTH=EXTERNAL
xoauth2	AUTH=XOAUTH2
oauthbearer	AUTH=OAUTHBEARER

Методы аутентификации с передачей пароля открытым текстом (команда *LOGIN, AUTH=PLAIN* и *AUTH=LOGIN*) включены всегда, однако если методы *plain* и *login* не указаны, то *AUTH=PLAIN* и *AUTH=LOGIN* не будут автоматически добавляться в *imap_capabilities*.

imap_capabilities

Синтаксис	imap_capabilities <i>расширение</i> ...;
По умолчанию	imap_capabilities IMAP4 IMAP4rev1 UIDPLUS;
Контекст	mail, server

Позволяет указать список расширений протокола IMAP, выдаваемый клиенту по команде CAPABILITY. В зависимости от значения директивы *starttls* к этому списку автоматически добавляются методы аутентификации, указанные в директиве *imap_auth*, и STARTTLS.

В данной директиве имеет смысл указать расширения, поддерживаемые IMAP-серверами, на которые проксируются клиенты (если эти расширения относятся к командам, используемым после аутентификации, когда Angie прозрачно проксирует подключение клиента на бэкенд).

imap_client_buffer

Синтаксис	imap_client_buffer размер;
По умолчанию	imap_client_buffer 4k 8k;
Контекст	mail, server

Задаёт размер буфера для чтения IMAP-команд. По умолчанию размер одного буфера равен размеру страницы. В зависимости от платформы это или 4К, или 8К.

POP3

Модуль обеспечивает поддержку почтового протокола POP3, позволяя серверу загружать сообщения с почтовых серверов. Он подключается к серверам POP3, получает заголовки и содержимое сообщений, обеспечивает безопасную аутентификацию и управляет статусами сообщений, такими как загружено или удалено.

Директивы

pop3_auth

Синтаксис	pop3_auth <i>метод</i> ...;
По умолчанию	pop3_auth plain;
Контекст	mail, server

Задаёт разрешённые методы аутентификации POP3-клиентов. Поддерживаемые методы:

plain	USER/PASS, AUTH PLAIN, AUTH LOGIN
apop	APOP. Для работы этого метода пароль должен храниться в незашифрованном виде.
cram-md5	AUTH=CRAM-MD5. Для работы этого метода пароль должен храниться в незашифрованном виде.
external	AUTH=EXTERNAL
xoauth2	AUTH=XOAUTH2
oauthbearer	AUTH=OAUTHBEARER

Методы аутентификации с передачей пароля открытым текстом (USER/PASS, AUTH PLAIN и AUTH LOGIN) включены всегда, однако если метод **plain** не указан, то *AUTH PLAIN* и *AUTH LOGIN* не будут автоматически добавляться в *pop3_capabilities*.

pop3_capabilities

Синтаксис	pop3_capabilities <i>расширение</i> ...;
По умолчанию	pop3_capabilities TOP USER UIDL;
Контекст	mail, server

Позволяет указать список расширений протокола POP3, выдаваемый клиенту по команде CAPA. В зависимости от значения директивы *starttls* к этому списку автоматически добавляются методы аутентификации, указанные в директиве *pop3_auth* (расширение SASL), и STLS.

В данной директиве имеет смысл указать расширения, поддерживаемые POP3-серверами, на которые проксируются клиенты (если эти расширения относятся к командам, используемым после аутентификации, когда Angie прозрачно проксирует подключение клиента на бэкенд).

Proxy

Модуль обеспечивает поддержку почтовых протоколов (POP3, IMAP, SMTP), позволяя серверу работать в качестве прокси между клиентами и почтовыми серверами. Он устанавливает соединения с серверами, выполняет безопасную аутентификацию с использованием открытого текста, SSL/TLS или STARTTLS, правильно маршрутизирует трафик клиентов и поддерживает гибкий выбор методов аутентификации и серверов.

Директивы

proxy_buffer

Синтаксис	proxy_buffer размер;
По умолчанию	proxy_buffer 4k 8k;
Контекст	mail, server

Задаёт размер буфера, используемого при проксировании. По умолчанию размер одного буфера равен размеру страницы. В зависимости от платформы это или 4К, или 8К.

proxy_pass_error_message

Синтаксис	proxy_pass_error_message on off;
По умолчанию	proxy_pass_error_message off;
Контекст	mail, server

Определяет, передавать ли клиенту сообщение об ошибке, полученное при аутентификации на бэкенде.

Обычно, если аутентификация в Angie прошла успешно, бэкенд не может вернуть ошибку. Если же он все-таки возвращает ошибку, это значит, что произошла ошибка внутри системы. В таких случаях сообщение бэкенда может содержать информацию, которую нельзя показывать клиенту. Однако для некоторых POP3-серверов ошибка в ответ на правильный пароль является штатным поведением. В этом случае директиву стоит включить.

proxy_protocol

Синтаксис	proxy_protocol on off;
По умолчанию	proxy_protocol off;
Контекст	mail, server

Включает протокол **PROXY** для соединений с бэкендом.

proxy_smtp_auth

Синтаксис	proxy_smtp_auth on off;
По умолчанию	proxy_smtp_auth off;
Контекст	mail, server

Разрешает или запрещает аутентификацию пользователей на SMTP-бэкенде при помощи команды AUTH.

Если также включен XCLIENT, то команда XCLIENT не будет отправлять параметр LOGIN.

proxy_tcp_congestion

Синтаксис	proxy_tcp_congestion алгоритм;
По умолчанию	—
Контекст	mail, server

Задаёт алгоритм управления перегрузкой TCP (параметр сокета TCP_CONGESTION) для исходящих соединений с проксируемым сервером; к соединениям через UNIX-сокеты параметр не применяется.

Имя алгоритма не проверяется на этапе конфигурации: если алгоритм неизвестен или недоступен, при установке исходящего соединения в лог записывается сообщение уровня **alert**, а для соединения действует системный алгоритм по умолчанию.

proxy_timeout

Синтаксис	proxy_timeout время;
По умолчанию	proxy_timeout 24h;
Контекст	mail, server

Задаёт таймаут между двумя идущими подряд операциями чтения или записи на клиентском соединении или соединении с проксируемым сервером. Если по истечении этого времени данные не передавались, соединение закрывается.

xclient

Синтаксис	xclient on off;
По умолчанию	xclient on;
Контекст	mail, server

Разрешает или запрещает передачу команды **XCLIENT** с параметрами клиента при подключении к SMTP-бэкенду.

При помощи XCLIENT МТА может писать в лог информацию о клиенте и применять различные ограничения на основе этих данных.

Если команда XCLIENT разрешена, то при подключении к бэкенду Angie посылает ему следующие команды:

- EHLO с именем сервера
- XCLIENT

- EHLO или HELO, как ее передал клиент

Если найденное по IP-адресу клиента имя указывает на тот же адрес, оно передается в параметре NAME команды XCLIENT. Если имя не может быть найдено, указывает на другой адрес, или не задан resolver, то в параметре NAME передается [UNAVAILABLE]. Если же в процессе поиска имени или адреса произошла ошибка, передается [TEMPUNAVAIL].

Если команда XCLIENT запрещена, то при подключении к бэкенду Angie передает команду EHLO с именем сервера, если клиент передал EHLO, иначе HELO с именем сервера.

RealIP

Позволяет менять адрес и необязательный порт клиента на переданные в указанном поле заголовка адрес и порт клиента на переданные в заголовке протокола PROXY. Протокол PROXY должен быть предварительно включен при помощи установки параметра proxy_protocol в директиве [listen](#).

Пример конфигурации

```
listen 110 proxy_protocol;

set_real_ip_from 192.168.1.0/24;
set_real_ip_from 192.168.2.1;
set_real_ip_from 2001:0db8::/32;
```

Директивы

set_real_ip_from

Синтаксис	<code>set_real_ip_from <i>адрес</i> <i>CIDR</i> <i>unix</i>::;</code>
По умолчанию	—
Контекст	mail, server

Задаёт доверенные адреса, которые передают верный адрес для замены. Если указано специальное значение `unix::`, доверенными будут считаться все UNIX-сокеты.

SMTP

Модуль обеспечивает поддержку почтового протокола SMTP, позволяя серверу проксировать исходящий почтовый трафик между клиентами и почтовыми серверами. Он устанавливает соединения с серверами SMTP, поддерживает безопасную аутентификацию с помощью методов LOGIN или PLAIN, обеспечивает шифрование через STARTTLS и SSL/TLS и маршрутизирует клиентские запросы на основе результатов аутентификации.

Директивы

smtp_auth

Синтаксис	<code>smtp_auth <i>method</i> ...;</code>
По умолчанию	<code>smtp_auth plain login;</code>
Контекст	mail, server

Задаёт разрешенные методы [SASL-аутентификации](#) SMTP-клиентов. Поддерживаемые методы:

plain	AUTH PLAIN
login	AUTH LOGIN
cram-md5	AUTH CRAM-MD5. Для работы этого метода пароль должен храниться в незашифрованном виде.
external	AUTH EXTERNAL
oauth2	AUTH XOAUTH2
oauthbearer	AUTH OAUTHBEARER
none	Аутентификация не требуется

Методы аутентификации с передачей пароля открытым текстом (AUTH PLAIN и AUTH LOGIN) включены всегда, однако если методы **plain** и **login** не указаны, то AUTH PLAIN и AUTH LOGIN не будут автоматически добавляться в *smtp_capabilities*.

smtp_capabilities

Синтаксис	smtp_capabilities <i>расширение ...</i> ;
По умолчанию	—
Контекст	mail, server

Позволяет указать список расширений протокола SMTP, выдаваемый клиенту в ответе на команду EHLO. В зависимости от значения директивы *starttls* к этому списку автоматически добавляются методы аутентификации, указанные в директиве *smtp_auth*, и STARTTLS.

В данной директиве имеет смысл указать расширения, поддерживаемые МТА, на который проксируются клиенты (если эти расширения относятся к командам, используемым после аутентификации, когда Angie прозрачно проксирует подключение клиента на бэкенд).

smtp_client_buffer

Синтаксис	smtp_client_buffer размер;
По умолчанию	smtp_client_buffer 4k 8k;
Контекст	mail, server

Задаёт размер буфера для чтения SMTP-команд. По умолчанию размер одного буфера равен размеру страницы. В зависимости от платформы это или 4К, или 8К.

smtp_greeting_delay

Синтаксис	smtp_greeting_delay размер;
По умолчанию	smtp_greeting_delay 0;
Контекст	mail, server

Позволяет задать задержку перед отправкой SMTP-приветствия, чтобы отклонить клиентов, не дожидаящихся приветствия до начала отправки SMTP-команд.

SSL

Модуль обеспечивает поддержку шифрования SSL/TLS для почтовых прокси-протоколов (POP3, IMAP, SMTP), позволяя устанавливать защищённые соединения между клиентами и сервером. Он

обеспечивает шифрование SSL/TLS для входящих подключений, поддерживает обновление соединений через STARTTLS, управляет сертификатами и ключами, а также контролирует настройки SSL, такие как выбор шифров и версий протоколов.

Примечание

Для этого модуля нужна библиотека OpenSSL.

Пример конфигурации

Для уменьшения загрузки процессора рекомендуется

- установить число рабочих процессов равным числу процессоров,
- включить разделяемый кэш сессий,
- выключить встроенный кэш сессий
- и, возможно, увеличить время жизни сессии (по умолчанию 5 минут):

```
worker_processes auto;

mail {
    ...

    server {
        listen          993 ssl;

        ssl_protocols   TLSv1.2 TLSv1.3;
        ssl_ciphers      AES128-SHA:AES256-SHA:RC4-SHA:DES-CBC3-SHA:RC4-MD5;
        ssl_certificate   /usr/local/angie/conf/cert.pem;
        ssl_certificate_key /usr/local/angie/conf/cert.key;
        ssl_session_cache shared:SSL:10m;
        ssl_session_timeout 10m;

        # ...
    }
}
```

Директивы

ssl_certificate

Синтаксис	ssl_certificate <i>файл</i> ;
По умолчанию	—
Контекст	mail, server

Указывает файл с сертификатом в формате PEM для данного сервера. Если вместе с основным сертификатом нужно указать промежуточные, то они должны находиться в этом же файле в следующем порядке — сначала основной сертификат, а затем промежуточные. В этом же файле может находиться секретный ключ в формате PEM.

Директива может быть указана несколько раз для загрузки сертификатов разных типов, например RSA и ECDSA:

```
server {
    listen          993 ssl;
```

```
ssl_certificate      example.com.rsa.crt;
ssl_certificate_key  example.com.rsa.key;

ssl_certificate      example.com.ecdsa.crt;
ssl_certificate_key  example.com.ecdsa.key;

#    ...
}
```

Возможность задавать отдельные цепочки сертификатов для разных сертификатов есть только в OpenSSL 1.0.2 и выше. Для более старых версий следует указывать только одну цепочку сертификатов.

Вместо **файла** можно указать значение "data:*сертификат*", при котором сертификат загружается без использования промежуточных файлов.

Ненадлежащее использование подобного синтаксиса может быть небезопасно, например данные секретного ключа могут попасть в лог ошибок.

ssl_certificate_compression

Синтаксис	ssl_certificate_compression on off;
По умолчанию	ssl_certificate_compression off;
Контекст	mail, server

Разрешает сжатие TLS 1.3 **сертификатов сервера**.

i Примечание

Директива поддерживается при использовании OpenSSL версии 3.2 и выше; список поддерживаемых алгоритмов сжатия предоставляется библиотекой.

i Примечание

Директива поддерживается при использовании **BoringSSL**; список поддерживаемых алгоритмов сжатия включает **zlib**.

ssl_certificate_key

Синтаксис	ssl_certificate_key <i>файл</i> ;
По умолчанию	—
Контекст	mail, server

Указывает файл с секретным ключом в формате PEM для данного виртуального сервера.

Вместо **файла** можно указать значение "engine:*имя*:*id*", которое загружает ключ с указанным *id* из OpenSSL engine с заданным именем.

Вместо **файла** можно указать значение "store:*scheme*:*id*", которое используется для загрузки ключа с указанным *id* и URI-схемой *scheme*, зарегистрированной в OpenSSL provider, например **pkcs11**.

Вместо файла также можно указать значение "data:ключ", при котором секретный ключ загружается без использования промежуточных файлов. При этом следует учитывать, что ненадлежащее использование подобного синтаксиса может быть небезопасно, например данные секретного ключа могут попасть в лог ошибок.

ssl_ciphers

Синтаксис	ssl_ciphers <i>шифры</i> ;
По умолчанию	ssl_ciphers HIGH:!aNULL:!MD5;
Контекст	mail, server

Описывает разрешенные шифры. Шифры задаются в формате, поддерживаемом библиотекой OpenSSL, например:

```
ssl_ciphers ALL:!aNULL:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP;
```

Список шифров зависит от установленной версии OpenSSL. Полный список можно посмотреть с помощью команды `openssl ciphers`.

⚠ Предупреждение

Директива `ssl_ciphers` не настраивает шифры для TLS 1.3 при использовании OpenSSL. Для настройки шифров TLS 1.3 в OpenSSL используйте директиву `ssl_conf_command`, добавленную для расширенной конфигурации SSL.

- В LibreSSL шифры TLS 1.3 *можно* настраивать с помощью `ssl_ciphers`.
- В BoringSSL шифры TLS 1.3 настроить невозможно.

ssl_client_certificate

Синтаксис	ssl_client_certificate <i>файл</i> ;
По умолчанию	—
Контекст	mail, server

Указывает файл с доверенными сертификатами СА в формате PEM, которые используются для проверки клиентских сертификатов.

Список сертификатов будет отправляться клиентам. Если это нежелательно, можно воспользоваться директивой `ssl_trusted_certificate`.

ssl_conf_command

Синтаксис	ssl_conf_command <i>имя значение</i> ;
По умолчанию	—
Контекст	mail, server

Задаёт произвольные конфигурационные команды OpenSSL.

Примечание

Директива поддерживается при использовании OpenSSL 1.0.2 и выше. Чтобы настроить шифры TLS 1.3 в OpenSSL, используйте команду `ciphersuites`.

На одном уровне может быть указано несколько директив `ssl_conf_command`:

```
ssl_conf_command Options PrioritizeChaCha;
ssl_conf_command Ciphersuites TLS_CHACHA20_POLY1305_SHA256;
```

Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы `ssl_conf_command`.

Предупреждение

Изменение настроек OpenSSL напрямую может привести к неожиданному поведению.

ssl_crl

Синтаксис	<code>ssl_crl файл;</code>
По умолчанию	—
Контекст	mail, server

Указывает файл с отозванными сертификатами (CRL) в формате PEM, используемыми для проверки клиентских сертификатов.

ssl_dhparam

Синтаксис	<code>ssl_dhparam файл;</code>
По умолчанию	—
Контекст	mail, server

Указывает файл с параметрами для DHE-шифров.

Предупреждение

По умолчанию параметры не заданы, и соответственно DHE-шифры не будут использоваться.

ssl_ecdh_curve

Синтаксис	<code>ssl_ecdh_curve кривая;</code>
По умолчанию	<code>ssl_ecdh_curve auto;</code>
Контекст	mail, server

Задаёт кривую для ECDHE-шифров.

i Примечание

При использовании OpenSSL 1.0.2 и выше можно указывать несколько кривых, например:

```
ssl_ecdh_curve prime256v1:secp384r1;
```

Специальное значение `auto` соответствует встроенному в библиотеку OpenSSL списку кривых для OpenSSL 1.0.2 и выше, или `prime256v1` для более старых версий.

i Примечание

При использовании OpenSSL 1.0.2 и выше директива задает список кривых, поддерживаемых сервером. Поэтому для работы ECDSA-сертификатов важно, чтобы список включал кривые, используемые в сертификатах.

ssl_password_file

Синтаксис	<code>ssl_password_file <i>файл</i>;</code>
По умолчанию	—
Контекст	mail, server

Задаёт файл с паролями от секретных ключей, где каждый пароль указан на отдельной строке. Пароли применяются по очереди в момент загрузки ключа.

Пример:

```
mail {
    ssl_password_file /etc/keys/global.pass;
    ...

    server {
        server_name mail1.example.com;
        ssl_certificate_key /etc/keys/first.key;
    }

    server {
        server_name mail2.example.com;

        # вместо файла можно указать именованный канал
        ssl_password_file /etc/keys/fifo;
        ssl_certificate_key /etc/keys/second.key;
    }
}
```

ssl_prefer_server_ciphers

Синтаксис	<code>ssl_prefer_server_ciphers on off;</code>
По умолчанию	<code>ssl_prefer_server_ciphers off;</code>
Контекст	mail, server

При использовании протоколов SSLv3 и TLS устанавливает приоритет серверных шифров над

клиентскими.

ssl_protocols

Синтаксис	<code>ssl_protocols [SSLv2] [SSLv3] [TLSv1] [TLSv1.1] [TLSv1.2] [TLSv1.3];</code>
По умолчанию	<code>ssl_protocols TLSv1.2 TLSv1.3;</code>
Контекст	<code>mail, server</code>

Разрешает указанные протоколы.

Примечание

Параметры TLSv1.1 и TLSv1.2 работают только при использовании OpenSSL 1.0.1 и выше.
Параметр TLSv1.3 работает только при использовании OpenSSL 1.1.1 и выше.

ssl_session_cache

Синтаксис	<code>ssl_session_cache off none [builtin[:размер]] [shared:название:размер];</code>
По умолчанию	<code>ssl_session_cache none;</code>
Контекст	<code>mail, server</code>

Задаёт тип и размеры кэшей для хранения параметров сессий. Тип кэша может быть следующим:

off	жёсткое запрещение использования кэша сессий: Angie явно сообщает клиенту, что сессии не могут использоваться повторно.
none	мягкое запрещение использования кэша сессий: Angie сообщает клиенту, что сессии могут использоваться повторно, но на самом деле не хранит параметры сессии в кэше.
builtin	встроенный в OpenSSL кэш, используется в рамках только одного рабочего процесса. Размер кэша задается в сессиях. Если размер не задан, то он равен 20480 сессиям. Использование встроенного кэша может вести к фрагментации памяти.
shared	кэш, разделяемый между всеми рабочими процессами. Размер кэша задается в байтах, в 1 мегабайт может поместиться около 4000 сессий. У каждого разделяемого кэша должно быть произвольное название. Кэш с одинаковым названием может использоваться в нескольких серверах. Также он используется для автоматического создания, хранения и периодического обновления ключей TLS session tickets, если они не указаны явно с помощью директивы <code>ssl_session_ticket_key</code> .

Можно использовать одновременно оба типа кэша, например:

```
ssl_session_cache builtin:1000 shared:SSL:10m;
```

однако использование только разделяемого кэша без встроенного должно быть более эффективным.

ssl_session_ticket_key

Синтаксис	<code>ssl_session_ticket_key <i>файл</i>;</code>
По умолчанию	—
Контекст	mail, server

Задаёт файл с секретным ключом, применяемым при шифровании и расшифровании TLS session tickets. Директива необходима, если один и тот же ключ нужно использовать на нескольких серверах. По умолчанию используется случайно сгенерированный ключ.

Если указано несколько ключей, то только первый ключ используется для шифрования TLS session tickets. Это позволяет настроить ротацию ключей, например:

```
ssl_session_ticket_key current.key;
ssl_session_ticket_key previous.key;
```

Файл должен содержать 80 или 48 байт случайных данных и может быть создан следующей командой:

```
$ openssl rand 80 > ticket.key
```

В зависимости от размера файла для шифрования будет использоваться либо AES256 (для 80-байтных ключей), либо AES128 (для 48-байтных ключей).

ssl_session_tickets

Синтаксис	<code>ssl_session_tickets on off;</code>
По умолчанию	<code>ssl_session_tickets on;</code>
Контекст	mail, server

Разрешает или запрещает возобновление сессий при помощи TLS session tickets.

ssl_session_timeout

Синтаксис	<code>ssl_session_timeout время;</code>
По умолчанию	<code>ssl_session_timeout 5m;</code>
Контекст	mail, server

Задаёт время, в течение которого клиент может повторно использовать параметры сессии.

ssl_trusted_certificate

Синтаксис	<code>ssl_trusted_certificate <i>файл</i>;</code>
По умолчанию	—
Контекст	mail, server

Задаёт файл с доверенными сертификатами СА в формате PEM, которые используются для проверки клиентских сертификатов.

В отличие от `ssl_client_certificate`, список этих сертификатов не будет отправляться клиентам.

ssl_verify_client

Синтаксис	<code>ssl_verify_client on off optional optional_no_ca;</code>
По умолчанию	<code>ssl_verify_client off;</code>
Контекст	mail, server

Разрешает проверку клиентских сертификатов. Результат проверки передается в заголовке **Auth-SSL-Verify** в запросе аутентификации. Если при проверке клиентского сертификата произошла ошибка или клиент не предоставил требуемый сертификат, соединение закрывается.

optional	запрашивает клиентский сертификат, и если сертификат был предоставлен, проверяет его
optional_no_ca	запрашивает сертификат клиента, но не требует, чтобы он был подписан доверенным сертификатом СА. Это предназначено для случаев, когда фактическая проверка сертификата осуществляется внешним по отношению к Angie сервисом. Содержимое сертификата доступно в запросах, посылаемых на сервер аутентификации.

ssl_verify_depth

Синтаксис	<code>ssl_verify_depth <i>число</i>;</code>
По умолчанию	<code>ssl_verify_depth 1;</code>
Контекст	mail, server

Устанавливает глубину проверки в цепочке клиентских сертификатов.

starttls

Синтаксис	<code>starttls on off only;</code>
По умолчанию	<code>starttls off;</code>
Контекст	mail, server

on	разрешить использование команд STLS для POP3 и STARTTLS для IMAP и SMTP;
off	запретить использование команд STLS и STARTTLS;
only	требовать предварительного перехода на TLS.

4.10.8.7 Сторонние модули

Мы собираем пакеты для популярных сторонних nginx-совместимых модулей, разработанных за пределами нашей компании.

Примечание

Мы не проверяем исходный код этих модулей и не отвечаем за последствия их установки; пакеты собираются на основе многочисленных запросов *исключительно* для удобства пользователей.

Перечень модулей

Модуль	Версия	Описание
<i>GeoIP2</i>	3.4	Добавляет поиск по геоданным в базах MaxMind GeoIP2.
<i>NJS:</i> • <i>HTTP JS</i> • <i>Stream JS</i>	0.9.5	Позволяют использовать njs, подмножество языка JavaScript, в конфигурации Angie ADC соответственно в контекстах <code>http</code> и <code>stream</code> .

GeoIP2

Модуль GeoIP2 реализует поиск в базах MaxMind GeoIP2 по IP-адресу клиента (по умолчанию) или по значению определенной переменной. Поддерживает как IPv4, так и IPv6.

Установка

Модуль включен в сборку.

Загрузка модуля

Для работы с модулем необходимо загрузить его в контексте `main{}`:

```
load_module modules/nginx_http_geoip2_module.so;    # для использования в блоке http{}
load_module modules/nginx_stream_geoip2_module.so;  # для использования в контексте
↪stream{}
```

Пример конфигурации

```
http {
    geoip2 /usr/share/GeoIP/GeoLite2-Country.mmdb {
        auto_reload 1h;
        $geoip2_country_code default=RU source=$http_x_forwarded_for country iso_code;
        $geoip2_country_name source=$http_x_forwarded_for country names ru;
    }

    log_format with_geoip '$server_port $remote_addr - $remote_user [$time_local] '
↪'$request' '
                                '$status $body_bytes_sent "$http_referer' '
                                '"$http_user_agent" "$http_x_forwarded_for" "$http_host' '
                                'country="$geoip2_country_code"';

    map $geoip2_country_code $denied {
        PL "1";
        QA "1";
    }

    server {
        listen 80;
        root /usr/share/angie/html;
        index index.html index.htm;
        access_log /var/log/angie/geoip_access.log with_geoip;

        if ($denied) {
            return 403;
        }
    }
}
```

```
}  
}
```

Примеры выполнения запросов

В примерах ниже 192.0.2.0 это адрес Angie ADC.

```
$ curl -H'X-Forwarded-For: 51.68.138.153' http://192.0.2.0  
  
<html>  
<head><title>403 Forbidden</title></head>  
<body>  
<center><h1>403 Forbidden</h1></center>  
<hr><center>Angie/1.12.2</center>  
</body>  
</html>
```

```
$ curl -H'X-Forwarded-For: 8.8.8.8' http://192.0.2.0  
  
ip = 8.8.8.8  
code = US  
name = США
```

```
$ curl -H'X-Forwarded-For: 77.88.44.242' http://192.0.2.0  
  
ip = 77.88.44.242  
code = RU  
name = Россия
```

Дополнительная информация

Подробная документация и исходный код доступны по ссылке: https://github.com/leev/nginx_http_geoip2_module

NJS

Модуль обеспечивает интеграцию языка программирования JavaScript в модель обработки событий Angie ADC и позволяет расширять функциональность сервера с помощью скриптов на JavaScript. Состоит из двух модулей:

- *HTTP JS* — для обработки HTTP-трафика;
- *Stream JS* — для обработки TCP/UDP-трафика.

Возможности

Модуль расширяет функциональность сервера с помощью скриптов на языке njs, подмножестве JavaScript, позволяя реализовывать пользовательскую логику на стороне сервера и многое другое:

- Сложные проверки контроля доступа и безопасности до того, как запрос достигнет проксируемого сервера.
- Манипулирование заголовками ответа.
- Написание гибких асинхронных обработчиков и фильтров контента.

Также доступна автономная утилита командной строки, которая может использоваться независимо от сервера для разработки и отладки njs-скриптов.

Загрузка модуля

Загрузка модулей в контексте `main{}`:

```
load_module modules/nginx_http_js_module.so;    # для HTTP
load_module modules/nginx_stream_js_module.so;   # для Stream
```

Использование

Подробная документация доступна в разделах отдельных модулей:

- *HTTP JS*
- *Stream JS*

Безопасность

Модуль не выполняет динамический код, в особенности код, полученный из сети. Единственный способ выполнить такой код с помощью `njs` — это настроить директиву `js_import` в конфигурации сервера. JavaScript-код загружается однократно при запуске сервера.

В модели угроз модуля JavaScript-код считается доверенным источником так же, как конфигурационный файл и сертификаты сайтов. На практике это означает следующее:

- раскрытие содержимого памяти и другие проблемы безопасности, вызванные модификацией JavaScript-кода, не считаются проблемами безопасности, а рассматриваются как обычные ошибки;
- необходимо принимать меры по защите JavaScript-кода, используемого модулем;
- если в конфигурационном файле отсутствуют директивы `js_import`, сервер защищен от уязвимостей, связанных с JavaScript.

Предзагружаемые объекты

Для каждого входящего запроса модуль создает отдельную виртуальную машину. Это дает множество преимуществ, таких как предсказуемое потребление памяти и изоляция запросов. Однако, поскольку все запросы изолированы, если обработчику запроса необходимо получить доступ к каким-либо данным, он должен прочитать их самостоятельно. Это неэффективно, особенно когда объем данных велик.

Для решения этой проблемы был введен механизм предзагружаемых общих объектов. Такие объекты создаются неизменяемыми и не имеют цепочек прототипов: их значения не могут быть изменены, свойства не могут быть добавлены или удалены.

Вот несколько примеров работы с предзагружаемыми объектами в `njs`:

- Доступ к свойствам по имени:

```
preloaded_object.prop_name
preloaded_object[prop_name]
```

- Перечисление свойств:

```
for (i in preloaded_object_name) {
    // ...
}
```

- Применение немодифицирующих встроенных методов с помощью `call()`:

```
Array.prototype.filter.call(preloaded_object_name, ...)
```

Справочник API

Полный справочник всех объектов, методов и свойств njs:

- *Справочник API NJS*

Дополнительная информация

- Официальный сайт: <https://nginx.org/ru/docs/njs/>
- Примеры использования: <https://github.com/nginx/njs-examples/>

HTTP JS

Позволяет задавать обработчики на njs — подмножестве языка JavaScript.

Пример конфигурации

```
http {
    js_import http.js;

    js_set $foo      http.foo;
    js_set $summary  http.summary;
    js_set $hash     http.hash;

    resolver 10.0.0.1;

    server {
        listen 8000;

        location / {
            add_header X-Foo $foo;
            js_content http.baz;
        }

        location = /summary {
            return 200 $summary;
        }

        location = /hello {
            js_content http.hello;
        }

        location = /fetch {
            js_content          http.fetch;
            js_fetch_trusted_certificate /path/to/ISRG_Root_X1.pem;
        }

        location = /crypto {
            add_header Hash $hash;
            return 200;
        }
    }
}
```

Файл http.js:

```
function foo(r) {
    r.log("hello from foo() handler");
}
```



```

    return "foo";
}

function summary(r) {
    var a, s, h;

    s = "JS summary\n\n";

    s += "Method: " + r.method + "\n";
    s += "HTTP version: " + r.httpVersion + "\n";
    s += "Host: " + r.headersIn.host + "\n";
    s += "Remote Address: " + r.remoteAddress + "\n";
    s += "URI: " + r.uri + "\n";

    s += "Headers:\n";
    for (h in r.headersIn) {
        s += "  header '" + h + "' is '" + r.headersIn[h] + "'\n";
    }

    s += "Args:\n";
    for (a in r.args) {
        s += "  arg '" + a + "' is '" + r.args[a] + "'\n";
    }

    return s;
}

function baz(r) {
    r.status = 200;
    r.headersOut.foo = 1234;
    r.headersOut['Content-Type'] = "text/plain; charset=utf-8";
    r.headersOut['Content-Length'] = 15;
    r.sendHeader();
    r.send("nginx");
    r.send("java");
    r.send("script");

    r.finish();
}

function hello(r) {
    r.return(200, "Hello world!");
}

async function fetch(r) {
    let results = await Promise.all([ngx.fetch('https://example.com/'),
                                     ngx.fetch('https://example.org/')]);

    r.return(200, JSON.stringify(results, undefined, 4));
}

async function hash(r) {
    let hash = await crypto.subtle.digest('SHA-512', r.headersIn.host);
    r.setReturnValue(Buffer.from(hash).toString('hex'));
}

export default {foo, summary, baz, hello, fetch, hash};

```

Директивы

js_body_filter

<i>Синтаксис</i>	<code>js_body_filter функция модуль.функция [buffer_type=строка буфер];</code>
По умолчанию	—
<i>Контекст</i>	location, if in location, limit_except

Задаёт функцию `пjs` в качестве фильтра тела ответа. Функция фильтра вызывается для каждого блока данных тела ответа со следующими аргументами:

r	объект <i>HTTP request</i>
data	входящий блок данных может быть строкой или буфером в зависимости от значения <i>buffer_type</i> , по умолчанию является строкой.
flags	объект со следующими свойствами: - <i>last</i> — логическое значение; <i>true</i> , если данные являются последним буфером.

Функция фильтра может передавать свою модифицированную версию входящего блока данных следующему фильтру тела ответа при помощи вызова *r.sendBuffer()*. Пример преобразования букв в нижний регистр в теле ответа:

```
function filter(r, data, flags) {
    r.sendBuffer(data.toLowerCase(), flags);
}
```

Для отмены фильтра (блоки данных будут передаваться клиенту без вызова *js_body_filter*), можно использовать *r.done()*.

Если функция фильтра изменяет длину тела ответа, то необходимо очистить заголовок ответа *Content-Length* (если присутствует) в *js_header_filter*, чтобы применить поблочное кодирование.

Примечание

Так как обработчик *js_body_filter* должен сразу возвращать результат, то поддерживаются только синхронные операции. Таким образом, асинхронные операции, например *r.subrequest()* или *setTimeout()*, не поддерживаются.

js_content

<i>Синтаксис</i>	<code>js_content функция модуль.функция;</code>
По умолчанию	—
<i>Контекст</i>	location, if in location, limit_except

Задаёт функцию `пjs` в качестве обработчика содержимого *location*. Можно ссылаться на функцию модуля.

js_context_reuse

<i>Синтаксис</i>	<code>js_context_reuse число;</code>
По умолчанию	<code>js_context_reuse 128;</code>
<i>Контекст</i>	http, server, location

Устанавливает максимальное количество контекстов JS для повторного использования движком QuickJS. Каждый контекст используется для одного запроса. Завершенный контекст помещается в пул повторно используемых контекстов. Если пул заполнен, контекст уничтожается.

js_engine

<i>Синтаксис</i>	<code>js_engine njs qjs;</code>
По умолчанию	<code>js_engine njs;</code>
<i>Контекст</i>	http, server, location

Устанавливает движок JavaScript для использования в скриптах njs. Параметр njs задает движок njs, также используемый по умолчанию. Параметр qjs задает движок QuickJS.

js_fetch_buffer_size

<i>Синтаксис</i>	<code>js_fetch_buffer_size размер;</code>
По умолчанию	<code>js_fetch_buffer_size 16k;</code>
<i>Контекст</i>	http, server, location

Задаёт размер буфера, который будет использоваться для чтения и записи для *Fetch API*.

js_fetch_ciphers

<i>Синтаксис</i>	<code>js_fetch_ciphers шифры;</code>
По умолчанию	<code>js_fetch_ciphers HIGH:!aNULL:!MD5;</code>
<i>Контекст</i>	http, server, location

Описывает разрешенные шифры для HTTPS-соединений при помощи *Fetch API*. Шифры задаются в формате, поддерживаемом библиотекой OpenSSL.

Список шифров зависит от установленной версии OpenSSL. Полный список можно посмотреть с помощью команды `openssl ciphers`.

js_fetch_max_response_buffer_size

<i>Синтаксис</i>	<code>js_fetch_max_response_buffer_size размер;</code>
По умолчанию	<code>js_fetch_max_response_buffer_size 1m;</code>
<i>Контекст</i>	http, server, location

Задаёт максимальный размер ответа, полученного при помощи *Fetch API*.

js_fetch_protocols

<i>Синтаксис</i>	<code>js_fetch_protocols [TLSv1] [TLSv1.1] [TLSv1.2] [TLSv1.3];</code>
По умолчанию	<code>js_fetch_protocols TLSv1 TLSv1.1 TLSv1.2;</code>
<i>Контекст</i>	http, server, location

Разрешает указанные протоколы для HTTPS-соединений при помощи *Fetch API*.

js_fetch_timeout

<i>Синтаксис</i>	<code>js_fetch_timeout время;</code>
По умолчанию	<code>js_fetch_timeout 60s;</code>
<i>Контекст</i>	http, server, location

Задаёт таймаут при чтении и записи при помощи *Fetch API*. Таймаут устанавливается не на всю передачу ответа, а только между двумя операциями чтения. Если по истечении этого времени данные не передавались, соединение закрывается.

js_fetch_trusted_certificate

<i>Синтаксис</i>	<code>js_fetch_trusted_certificate файл;</code>
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт файл с доверенными сертификатами СА в формате PEM, используемыми при проверке HTTPS-сертификата при помощи *Fetch API*.

js_fetch_verify

<i>Синтаксис</i>	<code>js_fetch_verify on off;</code>
По умолчанию	<code>js_fetch_verify on;</code>
<i>Контекст</i>	http, server, location

Разрешает или запрещает проверку сертификата HTTPS-сервера при помощи *Fetch API*.

js_fetch_verify_depth

<i>Синтаксис</i>	<code>js_fetch_verify_depth число;</code>
По умолчанию	<code>js_fetch_verify_depth 100;</code>
<i>Контекст</i>	http, server, location

Устанавливает глубину проверки в цепочке HTTPS-сертификатов при помощи *Fetch API*.

js_fetch_keepalive

<i>Синтаксис</i>	<code>js_fetch_keepalive</code> <i>соединения</i> ;
По умолчанию	<code>js_fetch_keepalive 0</code> ;
<i>Контекст</i>	http, server, location

Активирует кэш соединений к серверам назначения. Когда значение больше 0, включает keepalive-соединения для *Fetch API*.

Параметр *соединения* задает максимальное количество неактивных keepalive-соединений к серверам назначения, которые сохраняются в кэше каждого рабочего процесса. Когда это число превышено, закрываются наименее недавно использованные соединения.

Пример:

```
location /fetch {
    js_fetch_keepalive 32;
    js_fetch_trusted_certificate /path/to/ISRG_Root_X1.pem;
    js_content main.fetch_handler;
}
```

js_fetch_keepalive_requests

<i>Синтаксис</i>	<code>js_fetch_keepalive_requests</code> <i>число</i> ;
По умолчанию	<code>js_fetch_keepalive_requests 1000</code> ;
<i>Контекст</i>	http, server, location

Задаёт максимальное количество запросов, которые могут обрабатываться через одно keepalive-соединение с *Fetch API*. После выполнения максимального количества запросов соединение закрывается.

Периодическое закрытие соединений необходимо для освобождения выделенной памяти для каждого соединения. Поэтому использование слишком большого максимального количества запросов может привести к чрезмерному использованию памяти и не рекомендуется.

js_fetch_keepalive_time

<i>Синтаксис</i>	<code>js_fetch_keepalive_time</code> <i>время</i> ;
По умолчанию	<code>js_fetch_keepalive_time 1h</code> ;
<i>Контекст</i>	http, server, location

Ограничивает максимальное время, в течение которого запросы могут обрабатываться через одно keepalive-соединение с *Fetch API*. По достижении этого времени соединение закрывается после обработки следующего запроса.

js_fetch_keepalive_timeout

<i>Синтаксис</i>	<code>js_fetch_keepalive_timeout</code> <i>время</i> ;
По умолчанию	<code>js_fetch_keepalive_timeout 60s</code> ;
<i>Контекст</i>	http, server, location

Задаёт таймаут, в течение которого неактивное keepalive-соединение к серверу назначения остается открытым при помощи *Fetch API*.

js_header_filter

<i>Синтаксис</i>	<code>js_header_filter функция модуль.функция;</code>
По умолчанию	—
<i>Контекст</i>	location, if in location, limit_except

Задаёт функцию `pjs` в качестве фильтра заголовка ответа. Директива позволяет менять произвольные поля заголовка ответа.

Примечание

Так как обработчик `js_header_filter` должен сразу возвращать результат, то поддерживаются только синхронные операции. Таким образом, асинхронные операции, например `r.subrequest()` или `setTimeout()`, не поддерживаются.

js_import

<i>Синтаксис</i>	<code>js_import модуль.js имя_экспорта from модуль.js;</code>
По умолчанию	—
<i>Контекст</i>	http, server, location

Импортирует модуль, позволяющий задавать обработчики location и переменных на `pjs`. *Имя_экспорта* является пространством имен при доступе к функциям модуля. Если *имя_экспорта* не задано, то пространством имен будет являться имя модуля.

```
js_import http.js;
```

В примере при доступе к экспорту в качестве пространства имен используется имя модуля `http`. Если импортируемый модуль экспортирует `foo()`, то для доступа используется `http.foo`.

Директив `js_import` может быть несколько.

js_path

<i>Синтаксис</i>	<code>js_path путь;</code>
По умолчанию	—
<i>Контекст</i>	http, server, location

Задаёт дополнительный путь для модулей `pjs`.

js_periodic

<i>Синтаксис</i>	<code>js_periodic модуль.функция [interval=\ время] [jitter=\ число] [worker_affinity=\ маска];</code>
По умолчанию	—
<i>Контекст</i>	location

Задаёт обработчик содержимого для запуска через регулярные интервалы. Обработчик получает объект сеанса в качестве первого аргумента, он также имеет доступ к глобальным объектам, таким как `ngx`.

Необязательный параметр `interval` задаёт интервал между двумя последовательными запусками, по умолчанию — 5 секунд.

Необязательный параметр `jitter` задаёт время, в течение которого обработчик содержимого location будет случайно задержан, по умолчанию задержки нет.

По умолчанию `js_handler` выполняется в рабочем процессе 0. Необязательный параметр `worker_affinity` позволяет указать определённые рабочие процессы, в которых должен выполняться обработчик содержимого location. Каждый набор рабочих процессов представлен битовой маской разрешённых рабочих процессов. Маска `all` позволяет выполнять обработчик во всех рабочих процессах.

Пример:

```
example.conf:

location @periodics {
    # запуск с интервалом в 1 минуту в рабочем процессе 0
    js_periodic main.handler interval=60s;

    # запуск с интервалом в 1 минуту во всех рабочих процессах
    js_periodic main.handler interval=60s worker_affinity=all;

    # запуск с интервалом в 1 минуту в рабочих процессах 1 и 3
    js_periodic main.handler interval=60s worker_affinity=0101;

    resolver 10.0.0.1;
    js_fetch_trusted_certificate /path/to/ISRG_Root_X1.pem;
}
```

```
example.js:

async function handler(s) {
    let reply = await ngx.fetch('https://example.com/');
    let body = await reply.text();

    ngx.log(ngx.INFO, body);
}
```

js_preload_object

<i>Синтаксис</i>	<code>js_preload_object имя.json имя from файл.json;</code>
По умолчанию	—
<i>Контекст</i>	http, server, location

Предварительно загружает неизменяемый объект во время конфигурации. *Имя* используется в качестве имени глобальной переменной, через которую объект доступен в коде `цjs`. Если *имя* не указано, то будет использоваться имя файла.

```
js_preload_object map.json;
```

В примере `map` используется в качестве имени во время доступа к предварительно загруженному объекту.

Директив `js_preload_object` может быть несколько.

js_set

<i>Синтаксис</i>	<code>js_set \$переменная функция модуль.функция [nocache];</code>
По умолчанию	—
<i>Контекст</i>	<code>http, server, location</code>

Задаёт функцию `цjs` для указанной переменной. Можно ссылаться на функцию модуля.

Функция вызывается в момент первого обращения к переменной для данного запроса. Точный момент вызова функции зависит от *фазы*, в которой происходит обращение к переменной. Это можно использовать для реализации дополнительной логики, не относящейся к вычислению переменной. Например, если переменная указана в директиве `log_format`, то её обработчик не будет выполняться до фазы записи в лог. Этот обработчик также может использоваться для выполнения процедур непосредственно перед освобождением запроса.

Начиная с `цjs 0.8.6`, если указан необязательный аргумент `nocache`, обработчик вызывается каждый раз при обращении к нему. Из-за текущих ограничений модуля `rewrite`, когда переменная `nocache` используется директивой `set`, её обработчик должен всегда возвращать значение фиксированной длины.

Примечание

Так как обработчик `js_set` должен сразу возвращать результат, то поддерживаются только синхронные операции. Таким образом, асинхронные операции, например `r.subrequest()` или `setTimeout()`, не поддерживаются.

js_shared_dict_zone

<i>Синтаксис</i>	<code>js_shared_dict_zone zone=имя:размер [timeout=время] [type=string number] [evict] [state=файл];</code>
По умолчанию	—
<i>Контекст</i>	<code>http</code>

Задаёт *имя* и *размер* зоны разделяемой памяти, в которой хранится словарь ключей и значений, разделяемый между рабочими процессами.

type	необязательный параметр, позволяет изменить тип значения на числовой (number); по умолчанию в качестве ключа и значения используется строка (string).
timeout	необязательный параметр, задает время, по завершении которого все записи в словаре удаляются из зоны
evict	необязательный параметр, удаляет самую старую пару "ключ-значение" при переполнении зоны
state	необязательный параметр, указывает файл, который хранит состояние разделяемого словаря в формате JSON и делает его постоянным при перезапусках nginx

Пример:

```
example.conf:
# Создается словарь размером 1Мб со строковыми значениями,
# пары "ключ-значение" удаляются при отсутствии активности в течение 60 секунд:
js_shared_dict_zone zone=foo:1M timeout=60s;

# Создается словарь размером 512Кб со строковыми значениями,
# удаляется самая старая пара "ключ-значение" при переполнении зоны:
js_shared_dict_zone zone=bar:512K timeout=30s evict;

# Создается постоянный словарь размером 32Кб с числовыми значениями:
js_shared_dict_zone zone=num:32k type=number;

# Создается словарь размером 1Мб со строковыми значениями и постоянным
↪ состоянием:
js_shared_dict_zone zone=persistent:1M state=/tmp/dict.json;
```

```
example.js:
function get(r) {
    r.return(200, ngx.shared.foo.get(r.args.key));
}

function set(r) {
    r.return(200, ngx.shared.foo.set(r.args.key, r.args.value));
}

function delete(r) {
    r.return(200, ngx.shared.bar.delete(r.args.key));
}

function increment(r) {
    r.return(200, ngx.shared.num.incr(r.args.key, 2));
}
```

js_var

Синтаксис	js_var \$переменная [значение];
По умолчанию	—
Контекст	http, server, location

Объявляет *перезаписываемую* переменную. В качестве значения можно использовать текст, переменные и их комбинации. Переменная не перезаписывается после перенаправления, в отличие от

переменных, созданных при помощи директивы `set`.

Аргумент запроса

Каждый HTTP-обработчик `njs` получает один аргумент, объект *запроса*.

Stream JS

Позволяет задавать обработчики на `njs` — подмножестве языка JavaScript.

Пример конфигурации

```
stream {
    js_import stream.js;

    js_set $bar stream.bar;
    js_set $req_line stream.req_line;

    server {
        listen 12345;

        js_preread stream.preread;
        return $req_line;
    }

    server {
        listen 12346;

        js_access stream.access;
        proxy_pass 127.0.0.1:8000;
        js_filter stream.header_inject;
    }
}

http {
    server {
        listen 8000;
        location / {
            return 200 $http_foo\n;
        }
    }
}
```

Файл `stream.js`:

```
var line = '';

function bar(s) {
    var v = s.variables;
    s.log("hello from bar() handler!");
    return "bar-var" + v.remote_port + "; pid=" + v.pid;
}

function preread(s) {
    s.on('upload', function (data, flags) {
        var n = data.indexOf('\n');
        if (n != -1) {
            line = data.substr(0, n);
        }
    });
}
```

```

        s.done();
    }
});
}

function req_line(s) {
    return line;
}

// Чтение строки HTTP-запроса.
// Получение байт в 'req' до того как
// будет прочитана строка запроса.
// Добавление HTTP-заголовка в запрос клиента

var my_header = 'Foo: foo';
function header_inject(s) {
    var req = '';
    s.on('upload', function(data, flags) {
        req += data;
        var n = req.search('\n');
        if (n != -1) {
            var rest = req.substr(n + 1);
            req = req.substr(0, n + 1);
            s.send(req + my_header + '\r\n' + rest, flags);
            s.off('upload');
        }
    });
}

function access(s) {
    if (s.remoteAddress.match('^192.*')) {
        s.deny();
        return;
    }

    s.allow();
}

export default {bar, preread, req_line, header_inject, access};

```

Директивы

js_access

<i>Синтаксис</i>	js_access функция модуль.функция;
По умолчанию	—
<i>Контекст</i>	stream, server

Задаёт функцию `pjs`, которая будет вызываться в *access-фазе*. Можно ссылаться на функцию модуля.

Функция вызывается однократно при первом достижении сессией *access-фазе*. Функция вызывается со следующими аргументами:

s	объект <i>stream-сессии</i>
----------	-----------------------------

В этой фазе может происходить инициализация, также при помощи метода *s.on()* может регистрироваться вызов для каждого входящего блока данных пока не будет вызван один из методов: *s.done()*, *s.decline()*, *s.allow()*. При вызове любого из этих методов обработка сессии переходит на следующую фазу и все текущие вызовы *s.on()* сбрасываются.

js_context_reuse

Синтаксис	js_context_reuse число;
По умолчанию	js_context_reuse 128;
Контекст	stream, server

Устанавливает максимальное количество контекстов JS для повторного использования движком QuickJS. Каждый контекст используется для одной потоковой сессии. Завершенный контекст помещается в пул повторно используемых контекстов. Если пул заполнен, контекст уничтожается.

js_engine

Синтаксис	js_engine njs qjs;
По умолчанию	js_engine njs;
Контекст	stream, server

Устанавливает движок JavaScript для использования в скриптах njs. Параметр njs задает движок njs, также используемый по умолчанию. Параметр qjs задает движок QuickJS.

js_fetch_buffer_size

Синтаксис	js_fetch_buffer_size размер;
По умолчанию	js_fetch_buffer_size 16k;
Контекст	stream, server

Задаёт размер буфера, который будет использоваться для чтения и записи для *Fetch API*.

js_fetch_ciphers

Синтаксис	js_fetch_ciphers шифры;
По умолчанию	js_fetch_ciphers HIGH:!aNULL:!MD5;
Контекст	stream, server

Описывает разрешенные шифры для HTTPS-соединений при помощи *Fetch API*. Шифры задаются в формате, поддерживаемом библиотекой OpenSSL.

Список шифров зависит от установленной версии OpenSSL. Полный список можно посмотреть с помощью команды `openssl ciphers`.

js_fetch_max_response_buffer_size

<i>Синтаксис</i>	js_fetch_max_response_buffer_size <i>размер</i> ;
По умолчанию	js_fetch_max_response_buffer_size 1m;
<i>Контекст</i>	stream, server

Задаёт максимальный размер ответа, полученного при помощи *Fetch API*.

js_fetch_protocols

<i>Синтаксис</i>	js_fetch_protocols [TLSv1] [TLSv1.1] [TLSv1.2] [TLSv1.3];
По умолчанию	js_fetch_protocols TLSv1 TLSv1.1 TLSv1.2;
<i>Контекст</i>	stream, server

Разрешает указанные протоколы для HTTPS-соединений при помощи *Fetch API*.

js_fetch_timeout

<i>Синтаксис</i>	js_fetch_timeout <i>время</i> ;
По умолчанию	js_fetch_timeout 60s;
<i>Контекст</i>	stream, server

Задаёт таймаут при чтении и записи при помощи *Fetch API*. Таймаут устанавливается не на всю передачу ответа, а только между двумя операциями чтения. Если по истечении этого времени данные не передавались, соединение закрывается.

js_fetch_trusted_certificate

<i>Синтаксис</i>	js_fetch_trusted_certificate <i>файл</i> ;
По умолчанию	—
<i>Контекст</i>	stream, server

Задаёт файл с доверенными сертификатами СА в формате PEM, используемыми при проверке HTTPS-сертификата при помощи *Fetch API*.

js_fetch_verify

<i>Синтаксис</i>	js_fetch_verify on off;
По умолчанию	js_fetch_verify on;
<i>Контекст</i>	stream, server

Разрешает или запрещает проверку сертификата HTTPS-сервера при помощи *Fetch API*.

js_fetch_verify_depth

<i>Синтаксис</i>	<code>js_fetch_verify_depth</code> <i>число</i> ;
По умолчанию	<code>js_fetch_verify_depth 100</code> ;
<i>Контекст</i>	stream, server

Устанавливает глубину проверки в цепочке HTTPS-сертификатов при помощи *Fetch API*.

js_fetch_keepalive

<i>Синтаксис</i>	<code>js_fetch_keepalive</code> <i>соединения</i> ;
По умолчанию	<code>js_fetch_keepalive 0</code> ;
<i>Контекст</i>	stream, server

Активирует кэш соединений к серверам назначения. Когда значение больше 0, включает keepalive-соединения для *Fetch API*.

Параметр *соединения* задает максимальное количество неактивных keepalive-соединений к серверам назначения, которые сохраняются в кэше каждого рабочего процесса. Когда это число превышено, закрываются наименее недавно использованные соединения.

Пример:

```
server {
    listen 12345;
    js_fetch_keepalive 32;
    js_fetch_trusted_certificate /path/to/ISRG_Root_X1.pem;
    js_preread main.fetch_handler;
}
```

js_fetch_keepalive_requests

<i>Синтаксис</i>	<code>js_fetch_keepalive_requests</code> <i>число</i> ;
По умолчанию	<code>js_fetch_keepalive_requests 1000</code> ;
<i>Контекст</i>	stream, server

Задаёт максимальное количество запросов, которые могут обрабатываться через одно keepalive-соединение с *Fetch API*. После выполнения максимального количества запросов соединение закрывается.

Периодическое закрытие соединений необходимо для освобождения выделенной памяти для каждого соединения. Поэтому использование слишком большого максимального количества запросов может привести к чрезмерному использованию памяти и не рекомендуется.

js_fetch_keepalive_time

<i>Синтаксис</i>	<code>js_fetch_keepalive_time</code> <i>время</i> ;
По умолчанию	<code>js_fetch_keepalive_time 1h</code> ;
<i>Контекст</i>	stream, server

Ограничивает максимальное время, в течение которого запросы могут обрабатываться через одно keepalive-соединение с *Fetch API*. По достижении этого времени соединение закрывается после обработки следующего запроса.

js_fetch_keepalive_timeout

<i>Синтаксис</i>	<code>js_fetch_keepalive_timeout время;</code>
По умолчанию	<code>js_fetch_keepalive_timeout 60s;</code>
<i>Контекст</i>	stream, server

Задаёт таймаут, в течение которого неактивное keepalive-соединение к серверу назначения остается открытым при помощи *Fetch API*.

js_filter

<i>Синтаксис</i>	<code>js_filter функция модуль.функция;</code>
По умолчанию	—
<i>Контекст</i>	stream, server

Задаёт фильтр данных. Можно ссылаться на функцию модуля.

Функция фильтра вызывается однократно при первом достижении сессией *content-фазы*. Функция фильтра вызывается со следующими аргументами:

s	объект <i>stream-сессии</i>
----------	-----------------------------

В этой фазе может происходить инициализация, также при помощи метода *s.on()* может регистрироваться вызов для каждого входящего блока данных. Для отмены регистрации вызова и отмены фильтра можно использовать метод *s.off()*.

Примечание

Так как обработчик *js_filter* должен сразу возвращать результат, то поддерживаются только синхронные операции. Таким образом, асинхронные операции, например *ngx.fetch()* или *setTimeout()*, не поддерживаются.

js_import

<i>Синтаксис</i>	<code>js_import модуль.js имя_экспорта from модуль.js;</code>
По умолчанию	—
<i>Контекст</i>	stream, server

Импортирует модуль, позволяющий задавать обработчики location и переменных на *njs*. *Имя_экспорта* является пространством имен при доступе к функциям модуля. Если *имя_экспорта* не задано, то пространством имен будет являться имя модуля.

```
js_import stream.js;
```

В примере при доступе к экспорту в качестве пространства имен используется имя модуля *stream*. Если импортируемый модуль экспортирует *foo()*, то для доступа используется *stream.foo*.

Директив *js_import* может быть несколько.

js_path

<i>Синтаксис</i>	<code>js_path путь;</code>
По умолчанию	—
<i>Контекст</i>	stream, server

Задаёт дополнительный путь для модулей *njs*.

js_periodic

<i>Синтаксис</i>	<code>js_periodic модуль.функция [interval=\ время] [jitter=\ число] [worker_affinity=\ маска];</code>
По умолчанию	—
<i>Контекст</i>	server

Задаёт обработчик содержимого для запуска через регулярные интервалы. Обработчик получает объект сеанса в качестве первого аргумента, он также имеет доступ к глобальным объектам, таким как *ngx*.

Необязательный параметр *interval* задаёт интервал между двумя последовательными запусками, по умолчанию — 5 секунд.

Необязательный параметр *jitter* задаёт время, в течение которого обработчик содержимого location будет случайно задержан, по умолчанию задержки нет.

По умолчанию *js_handler* выполняется в рабочем процессе 0. Необязательный параметр *worker_affinity* позволяет указать определённые рабочие процессы, в которых должен выполняться обработчик содержимого location. Каждый набор рабочих процессов представлен битовой маской разрешённых рабочих процессов. Маска *all* позволяет выполнять обработчик во всех рабочих процессах.

Пример:

```
example.conf:

location @periodics {
    # запуск с интервалом в 1 минуту в рабочем процессе 0
    js_periodic main.handler interval=60s;

    # запуск с интервалом в 1 минуту во всех рабочих процессах
    js_periodic main.handler interval=60s worker_affinity=all;

    # запуск с интервалом в 1 минуту в рабочих процессах 1 и 3
    js_periodic main.handler interval=60s worker_affinity=0101;

    resolver 10.0.0.1;
    js_fetch_trusted_certificate /path/to/ISRG_Root_X1.pem;
}
```



```
example.js:

async function handler(s) {
  let reply = await ngx.fetch('https://example.com/');
  let body = await reply.text();

  ngx.log(ngx.INFO, body);
}
```

js_preload_object

Синтаксис	js_preload_object <i>имя.json</i> <i>имя from файл.json</i> ;
По умолчанию	—
Контекст	stream, server

Предварительно загружает неизменяемый объект во время конфигурации. *Имя* используется в качестве имени глобальной переменной, через которую объект доступен в коде `njs`. Если *имя* не указано, то будет использоваться имя файла.

```
js_preload_object map.json;
```

В примере *map* используется в качестве имени во время доступа к предварительно загруженному объекту.

Директив `js_preload_object` может быть несколько.

js_preread

Синтаксис	js_preread <i>функция</i> <i>модуль.функция</i> ;
По умолчанию	—
Контекст	stream, server

Задаёт функцию `njs`, которая будет вызываться в *preread-фазе*. Можно ссылаться на функцию модуля.

Функция вызывается однократно при первом достижении сессией *preread-фазы*. Функция вызывается со следующими аргументами:

s	объект <i>stream-сессии</i>
----------	-----------------------------

В этой фазе может происходить инициализация, также при помощи метода `s.on()` может регистрироваться вызов для каждого входящего блока данных пока не будет вызван один из методов: `s.done()`, `s.decline()`, `s.allow()`. При вызове любого из этих методов обработка сессии переходит на следующую фазу и все текущие вызовы `s.on()` сбрасываются.

Примечание

Так как обработчик `js_preread` должен сразу возвращать результат, то поддерживаются только синхронные операции. Таким образом, асинхронные операции, например `ngx.fetch()` или `setTimeout()`, не поддерживаются. Тем не менее асинхронные операции поддерживаются в вызовах `s.on()` в *preread-фазе*.

js_set

<i>Синтаксис</i>	<code>js_set \$переменная функция модуль.функция [nocache];</code>
По умолчанию	—
<i>Контекст</i>	stream, server

Задаёт функцию `njs` для указанной переменной. Можно ссылаться на функцию модуля.

Функция вызывается в момент первого обращения к переменной для данного запроса. Точный момент вызова функции зависит от *фазы*, в которой происходит обращение к переменной. Это можно использовать для реализации дополнительной логики, не относящейся к вычислению переменной. Например, если переменная указана в директиве `log_format`, то её обработчик не будет выполняться до фазы записи в лог. Этот обработчик также может использоваться для выполнения процедур непосредственно перед освобождением запроса.

Начиная с `njs 0.8.6`, если указан необязательный аргумент `nocache`, обработчик вызывается каждый раз при обращении к нему. Из-за текущих ограничений модуля `rewrite`, когда переменная `nocache` используется директивой `set`, её обработчик должен всегда возвращать значение фиксированной длины.

Примечание

Так как обработчик `js_set` должен сразу возвращать результат, то поддерживаются только синхронные операции. Таким образом, асинхронные операции, например `ngx.fetch()` или `setTimeout()`, не поддерживаются.

js_shared_dict_zone

<i>Синтаксис</i>	<code>js_shared_dict_zone zone=имя:размер [timeout=время] [type=string number] [evict] [state=файл];</code>
По умолчанию	—
<i>Контекст</i>	stream

Задаёт *имя* и *размер* зоны разделяемой памяти, в которой хранится словарь ключей и значений, разделяемый между рабочими процессами.

type	Необязательный параметр, позволяет изменить тип значения на числовой (number); по умолчанию в качестве ключа и значения используется строка (string).
timeout	Необязательный параметр, задаёт время, по завершении которого все записи в словаре удаляются из зоны.
evict	Необязательный параметр, удаляет самую старую пару "ключ-значение" при переполнении зоны.
state	необязательный параметр, указывает файл, который хранит состояние разделяемого словаря в формате JSON и делает его постоянным при перезапусках <code>nginx</code>

Пример:

```
example.conf:
# Создается словарь размером 1Мб со строковыми значениями,
# пары "ключ-значение" удаляются при отсутствии активности в течение 60 секунд:
```

```
js_shared_dict_zone zone=foo:1M timeout=60s;

# Создается словарь размером 512Kб со строковыми значениями,
# удаляется самая старая пара "ключ-значение" при переполнении зоны:
js_shared_dict_zone zone=bar:512K timeout=30s evict;

# Создается постоянный словарь размером 32Kб с числовыми значениями:
js_shared_dict_zone zone=num:32k type=number;

# Создается словарь размером 1Mб со строковыми значениями и постоянным состоянием:
js_shared_dict_zone zone=persistent:1M state=/tmp/dict.json;
```

```
example.js:
function get(r) {
    r.return(200, ngx.shared.foo.get(r.args.key));
}

function set(r) {
    r.return(200, ngx.shared.foo.set(r.args.key, r.args.value));
}

function delete(r) {
    r.return(200, ngx.shared.bar.delete(r.args.key));
}

function increment(r) {
    r.return(200, ngx.shared.num.incr(r.args.key, 2));
}
```

js_var

<i>Синтаксис</i>	js_var \$переменная [значение];
По умолчанию	—
Контекст	stream, server

Объявляет *перезаписываемую* переменную. В качестве значения можно использовать текст, переменные и их комбинации.

Свойства объекта сессии

Каждый stream-обработчик njs получает один аргумент, объект *stream-сессии*.

Справочник API NJS

Модуль NJS предоставляет объекты, методы и свойства для расширения функциональности Angie.

Данный справочник содержит только специфичные для NJS свойства, методы и модули, не соответствующие ECMAScript. Определения свойств и методов NJS, соответствующих ECMAScript, можно найти в спецификации ECMAScript.

Объекты Angie

HTTP-запрос

- `r.args{}`
- `r.done()`
- `r.error()`
- `r.finish()`
- `r.headersIn{}`
- `r.headersOut{}`
- `r.httpVersion`
- `r.internal`
- `r.internalRedirect()`
- `r.log()`
- `r.method`
- `r.parent`
- `r.remoteAddress`
- `r.requestBody`
- `r.requestBuffer`
- `r.requestText`
- `r.rawHeadersIn[]`
- `r.rawHeadersOut[]`
- `r.responseBody`
- `r.responseBuffer`
- `r.responseText`
- `r.return()`
- `r.send()`
- `r.sendBuffer()`
- `r.sendHeader()`
- `r.setReturnValue()`
- `r.status`
- `r.subrequest()`
- `r.uri`
- `r.rawVariables{}`
- `r.variables{}`
- `r.warn()`

Объект HTTP-запроса доступен только в модуле HTTP JS. До версии 0.8.5 все строковые свойства объекта были байтовыми строками.

`r.args{}`

Объект аргументов запроса, только для чтения.

Строка запроса возвращается в виде объекта. Начиная с версии 0.7.6 дублирующиеся ключи возвращаются в виде массива, ключи чувствительны к регистру, как ключи, так и значения декодируются из процентной кодировки.

Например, строка запроса

```
a=1&b=%32&A=3&b=4&B=two%20words
```

преобразуется в `r.args` следующим образом:

```
{a: "1", b: ["2", "4"], A: "3", B: "two words"}
```

Более сложные сценарии разбора можно реализовать с помощью модуля Query String и переменной `$args`, например:

```
import qs from 'querystring';

function args(r) {
  return qs.parse(r.variables.args);
}
```

Объект аргументов вычисляется при первом обращении к `r.args`. Если требуется только один аргумент, например `foo`, можно использовать переменные Angie:

```
r.variables.arg_foo
```

В этом случае объект переменных Angie возвращает первое значение для заданного ключа, без учета регистра и без декодирования процентной кодировки.

Для преобразования `r.args` обратно в строку можно использовать метод `stringify` модуля Query String.

`r.done()`

После вызова этой функции следующие фрагменты данных будут передаваться клиенту без вызова `js_body_filter` (0.5.2). Может вызываться только из функции `js_body_filter`.

`r.error(string)`

Записывает `string` в журнал ошибок на уровне логирования `error`.

Примечание

Поскольку в Angie жестко задано ограничение максимальной длины строки, в журнал может быть записано только первые 2048 байт строки.

`r.finish()`

Завершает отправку ответа клиенту.

`r.headersIn{}`

Объект входящих заголовков, только для чтения.

Заголовок запроса `Foo` может быть доступен с помощью синтаксиса: `headersIn.foo` или `headersIn['Foo']`.

Заголовки запроса `Authorization`, `Content-Length`, `Content-Range`, `Content-Type`, `ETag`, `Expect`, `From`, `Host`, `If-Match`, `If-Modified-Since`, `If-None-Match`, `If-Range`, `If-Unmodified-Since`, `Max-Forwards`, `Proxy-Authorization`, `Referer`, `Transfer-Encoding` и `User-Agent` могут иметь только одно значение поля (0.4.1). Дублирующиеся значения полей в заголовках `Cookie` разделяются точкой с запятой (;). Дублирующиеся значения полей во всех остальных заголовках запроса разделяются запятыми.

`r.headersOut{}`

Объект исходящих заголовков для основного запроса, для записи.

Если `r.headersOut{}` является объектом ответа подзапроса, он представляет заголовки ответа. В этом случае значения полей в заголовках ответа `Accept-Ranges`, `Connection`, `Content-Disposition`, `Content-Encoding`, `Content-Length`, `Content-Range`, `Date`, `Keep-Alive`, `Server`, `Transfer-Encoding`, `X-Accel-*` могут быть опущены.

Заголовок ответа `Foo` может быть доступен с помощью синтаксиса: `headersOut.foo` или `headersOut['Foo']`.

Исходящие заголовки должны быть установлены до отправки заголовка ответа клиенту; в противном случае обновление заголовка будет проигнорировано. Это означает, что `r.headersOut{}` фактически доступен для записи в:

- обработчике `js_content` до вызова `r.sendHeader()` или `r.return()`
- обработчике `js_header_filter`

Значения полей многозначных заголовков ответа (0.4.0) могут быть установлены с помощью синтаксиса:

```
r.headersOut['Foo'] = ['a', 'b']
```

где результат будет:

```
Foo: a
Foo: b
```

Все предыдущие значения полей заголовка ответа `Foo` будут удалены.

Для стандартных заголовков ответа, которые принимают только одно значение поля, таких как `Content-Type`, будет учитываться только последний элемент массива. Значения полей заголовка ответа `Set-Cookie` всегда возвращаются в виде массива. Дублирующиеся значения полей в заголовках ответа `Age`, `Content-Encoding`, `Content-Length`, `Content-Type`, `ETag`, `Expires`, `Last-Modified`, `Location`, `Retry-After` игнорируются. Дублирующиеся значения полей во всех остальных заголовках ответа разделяются запятыми.

`r.httpVersion`

Версия HTTP, только для чтения.

`r.internal`

Логическое значение, `true` для внутренних location.

`r.internalRedirect(uri)`

Выполняет внутреннее перенаправление на указанный `uri`. Если URI начинается с префикса `@`, он считается именованным location. В новом location вся обработка запроса повторяется, начиная с фазы `NGX_HTTP_SERVER_REWRITE_PHASE` для обычных location и с `NGX_HTTP_REWRITE_PHASE` для именованных location. В результате перенаправление на именованный location не проверяет ограничение `client_max_body_size`. Перенаправленные запросы становятся внутренними и могут обращаться к внутренним location. Фактическое перенаправление происходит после завершения выполнения обработчика.

Примечание

После перенаправления в целевом location запускается новая виртуальная машина NJS, а виртуальная машина в исходном location останавливается. Значения переменных Angie сохраняются и могут использоваться для передачи информации в целевой location. Начиная с версии 0.5.3 может использоваться переменная, объявленная с помощью директивы `js_var` для HTTP или Stream.

Примечание

Начиная с версии 0.7.4 метод принимает экранированные URI.

`r.log(string)`

Записывает `string` в журнал ошибок на уровне логирования `info`.

Примечание

Поскольку в Angie жестко задано ограничение максимальной длины строки, в журнал может быть записано только первые 2048 байт строки.

`r.method`

HTTP-метод, только для чтения.

`r.parent`

Ссылка на объект родительского запроса.

`r.remoteAddress`

Адрес клиента, только для чтения.

`r.requestBody`

Свойство устарело в версии 0.5.0 и было удалено в версии 0.8.0. Вместо него следует использовать свойство `r.requestBuffer` или `r.requestText`.

`r.requestBuffer`

Тело запроса клиента, если оно не было записано во временный файл (начиная с версии 0.5.0). Чтобы тело запроса клиента находилось в памяти, его размер должен быть ограничен директивой `client_max_body_size`, а размер буфера должен быть установлен с помощью `client_body_buffer_size`. Свойство доступно только в директиве `js_content`.

`r.requestText`

То же, что и `r.requestBuffer`, но возвращает `string`. Обратите внимание, что байты, недопустимые в кодировке UTF-8, могут быть преобразованы в символ замены.

`r.rawHeadersIn[]`

Возвращает массив пар ключ-значение точно так, как они были получены от клиента (0.4.1).

Например, при следующих заголовках запроса:

```
Host: localhost
Foo: bar
foo: bar2
```

вывод `r.rawHeadersIn` будет:

```
[
  ['Host', 'localhost'],
  ['Foo', 'bar'],
  ['foo', 'bar2']
]
```

Все заголовки `foo` можно собрать с помощью синтаксиса:

```
r.rawHeadersIn.filter(v=>v[0].toLowerCase() == 'foo').map(v=>v[1])
```

результат будет:

```
['bar', 'bar2']
```

Имена полей заголовков не преобразуются в нижний регистр, дублирующиеся значения полей не объединяются.

r.rawHeadersOut[]

Возвращает массив пар ключ-значение заголовков ответа (0.4.1). Имена полей заголовков не преобразуются в нижний регистр, дублирующиеся значения полей не объединяются.

r.responseBody

Свойство устарело в версии 0.5.0 и было удалено в версии 0.8.0. Вместо него следует использовать свойство **r.responseBuffer** или **r.responseText**.

r.responseBuffer

Содержит тело ответа подзапроса, только для чтения (начиная с версии 0.5.0). Размер **r.responseBuffer** ограничен директивой **subrequest_output_buffer_size**.

r.responseText

То же, что и **r.responseBuffer**, но возвращает строку (начиная с версии 0.5.0). Обратите внимание, что байты, недопустимые в кодировке UTF-8, могут быть преобразованы в символ замены.

r.return(status[, string | Buffer])

Отправляет полный ответ с указанным **status** клиенту. Ответ может быть строкой или буфером **Buffer** (0.5.0).

В качестве второго аргумента можно указать либо URL перенаправления (для кодов 301, 302, 303, 307 и 308), либо текст тела ответа (для других кодов).

r.send(string | Buffer)

Отправляет часть тела ответа клиенту. Отправляемые данные могут быть строкой или буфером **Buffer** (0.5.0).

r.sendBuffer(data[, options])

Добавляет данные в цепочку фрагментов данных, которые будут переданы следующему фильтру тела (0.5.2). Фактическая передача происходит позже, когда все фрагменты данных текущей цепочки обработаны.

Данные могут быть строкой или буфером **Buffer**. **options** — это объект, используемый для переопределения флагов буфера **Angie**, полученных из буфера входящего фрагмента данных. Флаги могут быть переопределены следующими флагами:

last

Логическое значение, **true**, если буфер является последним буфером.

flush

Логическое значение, **true**, если буфер должен иметь флаг **flush**.

Метод может вызываться только из функции **js_body_filter**.

r.sendHeader()

Отправляет HTTP-заголовки клиенту.

r.setReturnValue(value)

Устанавливает возвращаемое значение обработчика **js_set** (0.7.0). В отличие от обычного оператора **return**, этот метод следует использовать, когда обработчик является асинхронной функцией JS. Например:

```
async function js_set(r) {
  const digest = await crypto.subtle.digest('SHA-256', r.headersIn.host);
  r.setReturnValue(digest);
}
```

r.status

Статус, для записи.

`r.subrequest(uri[, options[, callback]])`

Создает подзапрос с заданными `uri` и `options` и устанавливает необязательный обратный вызов завершения `callback`.

Подзапрос разделяет свои входящие заголовки с клиентским запросом. Для отправки заголовков, отличных от исходных, прокси-серверу можно использовать директиву `proxy_set_header`. Для отправки совершенно нового набора заголовков прокси-серверу можно использовать директиву `proxy_pass_request_headers`.

Если `options` является строкой, она содержит строку аргументов подзапроса. В противном случае ожидается, что `options` будет объектом со следующими ключами:

`args`

Строка аргументов, по умолчанию используется пустая строка.

`body`

Тело запроса, по умолчанию используется тело запроса родительского объекта запроса.

`method`

HTTP-метод, по умолчанию используется метод `GET`.

`detached`

Логический флаг (0.3.9); если `true`, созданный подзапрос является отдельным подзапросом. Ответы на отдельные подзапросы игнорируются. В отличие от обычных подзапросов, отдельный подзапрос может быть создан внутри обработчика переменной. Флаг `detached` и аргумент `callback` взаимоисключающи.

Обратный вызов завершения `callback` получает объект ответа подзапроса с методами и свойствами, идентичными родительскому объекту запроса.

Начиная с версии 0.3.8, если `callback` не предоставлен, возвращается объект `Promise`, который разрешается в объект ответа подзапроса.

Например, для просмотра всех заголовков ответа в подзапросе:

```
async function handler(r) {
  const reply = await r.subrequest('/path');

  for (const h in reply.headersOut) {
    r.log(`${h}: ${reply.headersOut[h]}`);
  }

  r.return(200);
}
```

`r.uri`

Текущий URI в запросе, нормализованный, только для чтения.

`r.rawVariables{}`

Переменные Angie в виде буферов, для записи (начиная с версии 0.5.0).

`r.variables{}`

Объект переменных Angie, для записи (начиная с версии 0.2.8).

Например, для получения переменной `$foo` можно использовать один из следующих синтаксисов:

```
r.variables['foo']
r.variables.foo
```

Начиная с версии 0.8.6 к захватам регулярных выражений можно обращаться с помощью следующего синтаксиса:

```
r.variables['1']
r.variables[1]
```

Angie обрабатывает переменные, на которые есть ссылки в `angie.conf`, и переменные, на которые нет ссылок, по-разному. Когда на переменную есть ссылка, она может кэшироваться, но когда на нее нет ссылки, она всегда не кэшируется. Например, когда к переменной `$request_id` обращаются только из NJS, она имеет новое значение каждый раз при вычислении. Но когда на `$request_id` есть ссылка, например:

```
proxy_set_header X-Request-Id $request_id;
```

`r.variables.request_id` возвращает одно и то же значение каждый раз.

Переменная доступна для записи, если:

- она была создана с помощью директивы `js_var` для HTTP или Stream (начиная с версии 0.5.3)
- на нее есть ссылка в файле конфигурации Angie

Тем не менее, некоторым встроенным переменным все еще нельзя присвоить значение (например, `$http_`).

`r.warn(string)`

Записывает `string` в журнал ошибок на уровне логирования `warning`.

Примечание

Поскольку в Angie жестко задано ограничение максимальной длины строки, в журнал может быть записано только первые 2048 байт строки.

Stream-сессия

- `s.allow()`
- `s.decline()`
- `s.deny()`
- `s.done()`
- `s.error()`
- `s.log()`
- `s.off()`
- `s.on()`
- `s.remoteAddress`
- `s.rawVariables{}`
- `s.send()`
- `s.sendDownstream()`
- `s.sendUpstream()`
- `s.status`
- `s.setReturnValue()`
- `s.variables{}`
- `s.warn()`

Объект stream-сессии доступен только в модуле Stream JS. До версии 0.8.5 все строковые свойства объекта были байтовыми строками.

`s.allow()`

Псевдоним для `s.done(0)` (0.2.4).

`s.decline()`

Псевдоним для `s.done(-5)` (0.2.4).

`s.deny()`

Псевдоним для `s.done(403)` (0.2.4).

`s.done([code])`

Устанавливает код выхода `code` для обработчика текущей фазы в значение кода, по умолчанию 0. Фактическое завершение происходит, когда обработчик `js` завершен и все ожидающие события, например, из `ngx.fetch()` или `setTimeout()`, обработаны (0.2.4).

Возможные значения кода:

- 0 — успешное завершение, передача управления следующей фазе
- -5 — не определено, передача управления следующему обработчику текущей фазы (если есть)
- 403 — доступ запрещен

Может вызываться только из функции-обработчика фазы: `js_access` или `js_preread`.

`s.error(string)`

Записывает отправленную `string` в журнал ошибок на уровне логирования `error`.

Примечание

Поскольку в Angie жестко задано ограничение максимальной длины строки, в журнал может быть записано только первые 2048 байт строки.

`s.log(string)`

Записывает отправленную `string` в журнал ошибок на уровне логирования `info`.

Примечание

Поскольку в Angie жестко задано ограничение максимальной длины строки, в журнал может быть записано только первые 2048 байт строки.

`s.off(eventName)`

Отменяет регистрацию обратного вызова, установленного методом `s.on()` (0.2.4).

`s.on(event, callback)`

Регистрирует `callback` для указанного `event` (0.2.4).

`event` может быть одной из следующих строк:

`upload`

Новые данные (строка) от клиента.

`download`

Новые данные (строка) клиенту.

`upstream`

Новые данные (буфер) от клиента (начиная с версии 0.5.0).

`downstream`

Новые данные (буфер) клиенту (начиная с версии 0.5.0).

Обратный вызов завершения имеет следующий прототип: `callback(data, flags)`, где `data` — строка или буфер `Buffer` (в зависимости от типа события); `flags` — объект со следующими свойствами:

`last`

Логическое значение, `true`, если `data` является последним буфером.

`s.remoteAddress`

Адрес клиента, только для чтения.

`s.rawVariables`

Переменные Angie в виде буферов, для записи (начиная с версии 0.5.0).

`s.send(data[, options])`

Добавляет данные в цепочку фрагментов данных, которые будут переданы в прямом направлении: в обратном вызове `download` клиенту; в `upload` восходящему серверу (0.2.4). Фактическая передача происходит позже, когда все фрагменты данных текущей цепочки обработаны.

Данные могут быть строкой или буфером `Buffer` (0.5.0). `options` — это объект, используемый для переопределения флагов буфера Angie, полученных из буфера входящего фрагмента данных. Флаги могут быть переопределены следующими флагами:

`last`

Логическое значение, `true`, если буфер является последним буфером.

`flush`

Логическое значение, `true`, если буфер должен иметь флаг `flush`.

Метод может вызываться несколько раз за вызов обратного вызова.

`s.sendDownstream()`

Идентичен `s.send()`, за исключением того, что всегда отправляет данные клиенту (начиная с версии 0.7.8).

`s.sendUpstream()`

Идентичен `s.send()`, за исключением того, что всегда отправляет данные от клиента (начиная с версии 0.7.8).

`s.status`

Код статуса сессии, псевдоним переменной `$status`, только для чтения (начиная с версии 0.5.2).

`s.setReturnValue(value)`

Устанавливает возвращаемое значение обработчика `js_set` (0.7.0). В отличие от обычного оператора `return`, этот метод следует использовать, когда обработчик является асинхронной функцией JS. Например:

```
async function js_set(r) {
  const digest = await crypto.subtle.digest('SHA-256', r.headersIn.host);
  r.setReturnValue(digest);
}
```

`s.variables{}`

Объект переменных Angie, для записи (начиная с версии 0.2.8). Переменная может быть доступна для записи только в том случае, если на нее есть ссылка в файле конфигурации Angie. Тем не менее, некоторым встроенным переменным все еще нельзя присвоить значение.

`s.warn(string)`

Записывает отправленную `string` в журнал ошибок на уровне логирования `warning`.

Примечание

Поскольку в Angie жестко задано ограничение максимальной длины строки, в журнал может быть записано только первые 2048 байт строки.

Периодическая сессия

- `PeriodicSession.rawVariables{}`
- `PeriodicSession.variables{}`

Объект `Periodic Session` предоставляется в качестве первого аргумента обработчика `js_periodic` для HTTP и Stream (начиная с версии 0.8.1).

`PeriodicSession.rawVariables{}`

Переменные Angie в виде буферов, для записи.

`PeriodicSession.variables{}`

Объект переменных Angie, для записи.

Заголовки

- `Headers()`
- `Headers.append()`
- `Headers.delete()`
- `Headers.get()`
- `Headers.getAll()`
- `Headers.forEach()`
- `Headers.has()`
- `Headers.set()`

Интерфейс `Headers` из API Fetch доступен начиная с версии 0.5.1.

Новый объект `Headers` можно создать с помощью конструктора `Headers()` (начиная с версии 0.7.10):

`Headers([init])`

`init`

Объект, содержащий HTTP-заголовки для предварительного заполнения объекта `Headers`, может быть строкой, массивом пар имя-значение или существующим объектом `Headers`.

Новый объект `Headers` можно создать со следующими свойствами и методами:

`append()`

Добавляет новое значение в существующий заголовок в объекте `Headers` или добавляет заголовок, если он еще не существует (начиная с версии 0.7.10).

`delete()`

Удаляет заголовок из объекта `Headers` (начиная с версии 0.7.10).

`get()`

Возвращает строку, содержащую значения всех заголовков с указанным именем, разделенные запятой и пробелом.

`getAll(name)`

Возвращает массив, содержащий значения всех заголовков с указанным именем.

`forEach()`

Выполняет предоставленную функцию один раз для каждой пары ключ-значение в объекте `Headers` (начиная с версии 0.7.10).

`has()`

Возвращает логическое значение, указывающее, существует ли заголовок с указанным именем.

`set()`

Устанавливает новое значение для существующего заголовка в объекте **Headers** или добавляет заголовок, если он еще не существует (начиная с версии 0.7.10).

Запрос

- `Request()`
- `Request.arrayBuffer()`
- `Request.bodyUsed`
- `Request.cache`
- `Request.credentials`
- `Request.headers`
- `Request.json()`
- `Request.method`
- `Request.mode`
- `Request.text()`
- `Request.url`

Интерфейс **Request** из API Fetch доступен начиная с версии 0.7.10.

Новый объект **Request** можно создать с помощью конструктора `Request()`:

`Request[resource[, options]]()`

Создает объект **Request** для получения данных, который может быть позже передан в `ngx.fetch()`. Аргумент **resource** может быть URL-адресом или существующим объектом **Request**. Аргумент **options** является опциональным и ожидается быть объектом со следующими ключами:

body

Тело запроса, по умолчанию пусто.

headers

Объект заголовков ответа — объект, содержащий HTTP-заголовки для предварительного заполнения объекта **Headers**, может быть строкой, массивом пар имя-значение или существующим объектом **Headers**.

method

HTTP-метод, по умолчанию используется метод GET.

Новый объект **Request** можно создать со следующими свойствами и методами:

arrayBuffer()

Возвращает **Promise**, который разрешается в **ArrayBuffer**.

bodyUsed

Логическое значение, **true**, если тело было использовано в запросе.

cache

Содержит режим кэширования запроса.

credentials

Содержит учетные данные запроса, по умолчанию **same-origin**.

headers

Объект **Headers**, доступный только для чтения, связанный с **Request**.

json()

Возвращает **Promise**, который разрешается в результат анализа тела запроса как JSON.

method

Содержит метод запроса.

mode

Содержит режим запроса.

text()

Возвращает **Promise**, который разрешается в строковое представление тела запроса.

url

Содержит URL запроса.

Ответ

- **Response()**
- **Response.arrayBuffer()**
- **Response.bodyUsed**
- **Response.headers**
- **Response.json()**
- **Response.ok**
- **Response.redirected**
- **Response.status**
- **Response.statusText**
- **Response.text()**
- **Response.type**
- **Response.url**

Интерфейс **Response** доступен начиная с версии 0.5.1.

Новый объект **Response** можно создать с помощью конструктора **Response()** (начиная с версии 0.7.10):

Response[body[, options]])

Создает объект **Response**. Аргумент **body** является опциональным, может быть строкой или буфером, по умолчанию **null**. Аргумент **options** является опциональным и ожидается быть объектом со следующими ключами:

headers

Объект заголовков ответа — объект, содержащий HTTP-заголовки для предварительного заполнения объекта **Headers**, может быть строкой, массивом пар имя-значение или существующим объектом **Headers**.

status

Код состояния ответа.

statusText

Сообщение о состоянии, соответствующее коду состояния.

Новый объект **Response()** можно создать со следующими свойствами и методами:

arrayBuffer()

Берет поток **Response** и читает его до конца. Возвращает **Promise**, который разрешается в **ArrayBuffer**.

bodyUsed

Логическое значение, **true**, если тело было прочитано.

headers

Объект **Headers**, доступный только для чтения, связанный с **Response**.

`json()`

Берет поток **Response** и читает его до конца. Возвращает **Promise**, который разрешается в результат анализа текста тела как JSON.

`ok`

Логическое значение, **true**, если ответ был успешным (коды состояния между 200–299).

`redirected`

Логическое значение, **true**, если ответ является результатом перенаправления.

`status`

Код состояния ответа.

`statusText`

Сообщение о состоянии, соответствующее коду состояния.

`text()`

Берет поток **Response** и читает его до конца. Возвращает **Promise**, который разрешается в строку.

`type`

Тип ответа.

`url`

URL ответа.

ngx

- `ngx.build`
- `ngx.conf_file_path`
- `ngx.conf_prefix`
- `ngx.error_log_path`
- `ngx.fetch()`
- `ngx.log()`
- `ngx.prefix`
- `ngx.version`
- `ngx.version_number`
- `ngx.worker_id`

Глобальный объект **ngx** доступен начиная с версии 0.5.0.

`ngx.build`

Строка, содержащая опциональное имя сборки Angie, соответствует аргументу `--build=name` скрипта `configure`, по умолчанию `"` (0.8.0).

`ngx.conf_file_path`

Строка, содержащая путь к файлу текущей конфигурации Angie (0.8.0).

`ngx.conf_prefix`

Строка, содержащая путь к префиксу конфигурации Angie — каталог, в котором Angie ищет конфигурацию (0.7.8).

`ngx.error_log_path`

Строка, содержащая путь к файлу текущего журнала ошибок (0.8.0).

`ngx.fetch(resource, [options])`

Выполняет запрос для получения **resource** (0.5.1), который может быть URL-адресом или объектом **Request** (0.7.10). Возвращает **Promise**, который разрешается в объект **Response**. Начиная с версии 0.7.0 поддерживается схема `https://`; перенаправления не обрабатываются.

Если URL в `resource` указан как доменное имя, он определяется с помощью распознавателя. Если указана схема `https://`, директива `js_fetch_trusted_certificate` должна быть настроена для аутентификации HTTPS-сервера `resource`.

Параметр `options` ожидается быть объектом со следующими ключами:

`body`

Тело запроса, по умолчанию пусто.

`buffer_size`

Размер буфера для чтения ответа, по умолчанию 4096.

`headers`

Объект заголовков запроса.

`max_response_body_size`

Максимальный размер тела ответа в байтах, по умолчанию 32768.

`method`

HTTP-метод, по умолчанию используется метод `GET`.

`verify`

Включает или отключает проверку сертификата HTTPS-сервера, по умолчанию `true` (0.7.0).

Пример:

```
let reply = await ngx.fetch('http://example.com/');
let body = await reply.text();

r.return(200, body);
```

`ngx.log(level, message)`

Записывает сообщение в журнал ошибок с указанным уровнем логирования. Параметр `level` задает один из уровней логирования; параметр `message` может быть строкой или буфером. Можно задать следующие уровни логирования: `ngx.INFO`, `ngx.WARN` и `ngx.ERR`.

Примечание

Поскольку в Angie жестко задано ограничение максимальной длины строки, в журнал может быть записано только первые 2048 байт строки.

`ngx.prefix`

Строка, содержащая путь к префиксу Angie — каталогу, который содержит файлы сервера (0.8.0).

`ngx.version`

Строка, содержащая версию Angie, например: 1.25.0 (0.8.0).

`ngx.version_number`

Число, содержащее номер версии Angie, например: 1025000 (0.8.0).

`ngx.worker_id`

Число, соответствующее внутреннему идентификатору рабочего процесса Angie, значение между 0 и значением, указанным в директиве `worker_processes` (0.8.0).

`ngx.shared`

Глобальный объект `ngx.shared` доступен начиная с версии 0.8.0.

SharedDict

- `ngx.shared.SharedDict.add()`
- `ngx.shared.SharedDict.capacity`
- `ngx.shared.SharedDict.clear()`
- `ngx.shared.SharedDict.delete()`
- `ngx.shared.SharedDict.freeSpace()`
- `ngx.shared.SharedDict.get()`
- `ngx.shared.SharedDict.has()`
- `ngx.shared.SharedDict.incr()`
- `ngx.shared.SharedDict.items()`
- `ngx.shared.SharedDict.keys()`
- `ngx.shared.SharedDict.name`
- `ngx.shared.SharedDict.pop()`
- `ngx.shared.SharedDict.replace()`
- `ngx.shared.SharedDict.set()`
- `ngx.shared.SharedDict.size()`
- `ngx.shared.SharedDict.type`

Объект общей памяти доступен начиная с версии 0.8.0. Имя общей памяти, тип и размер устанавливаются с помощью директивы `js_shared_dict_zone` в HTTP или Stream.

Объект `SharedDict()` имеет следующие свойства и методы:

`ngx.shared.SharedDict.add(key, value [,timeout])`

Устанавливает `value` для указанного `key` в словаре только если ключ еще не существует. Аргумент `key` — это строка, представляющая ключ элемента для добавления; аргумент `value` — это значение элемента для добавления.

Опциональный аргумент `timeout` задается в миллисекундах и переопределяет параметр `timeout` директивы `js_shared_dict_zone` в HTTP или Stream (начиная с версии 0.8.5). Это может быть полезно, когда некоторые ключи ожидают иметь уникальные таймауты.

Возвращает `true`, если значение успешно добавлено в словарь `SharedDict`; `false`, если ключ уже существует в словаре. Выбрасывает `SharedMemoryError`, если в словаре `SharedDict` недостаточно свободного места. Выбрасывает `TypeError`, если значение `value` имеет другой тип, чем ожидается этот словарь.

`ngx.shared.SharedDict.capacity`

Возвращает емкость словаря `SharedDict`, соответствует параметру `size` директивы `js_shared_dict_zone` в HTTP или Stream.

`ngx.shared.SharedDict.clear()`

Удаляет все элементы из словаря `SharedDict`.

`ngx.shared.SharedDict.delete(key)`

Удаляет элемент, связанный с указанным ключом, из словаря `SharedDict`; `true`, если элемент в словаре существовал и был удален, `false` иначе.

`ngx.shared.SharedDict.freeSpace()`

Возвращает размер свободной страницы в байтах. Если размер равен нулю, словарь `SharedDict` все еще может принимать новые значения, если есть место на занятых страницах.

`ngx.shared.SharedDict.get(key)`

Получает элемент по его `key`; возвращает значение, связанное с `key`, или `undefined`, если его нет.

`ngx.shared.SharedDict.has(key)`

Ищет элемент по его `key`; возвращает `true`, если такой элемент существует, или `false` иначе.

`ngx.shared.SharedDict.incr(key, delta[, init], timeout)`

Увеличивает целое число, связанное с `key`, на `delta`. Аргумент `key` — это строка; аргумент `delta` — это число для увеличения или уменьшения значения. Если ключ не существует, элемент будет инициализирован опциональным аргументом `init`, по умолчанию 0.

Опциональный аргумент `timeout` задается в миллисекундах и переопределяет параметр `timeout` директивы `js_shared_dict_zone` в HTTP или Stream (начиная с версии 0.8.5). Это может быть полезно, когда некоторые ключи ожидают иметь уникальные таймауты.

Возвращает новое значение. Выбрасывает `SharedMemoryError`, если в словаре `SharedDict` недостаточно свободного места. Выбрасывает `TypeError`, если этот словарь не ожидает чисел.

Примечание

Этот метод можно использовать только если тип словаря был объявлен с параметром `type=number` директивы `js_shared_dict_zone` в HTTP или Stream.

`ngx.shared.SharedDict.items([maxCount])`

Возвращает массив элементов ключ-значение словаря `SharedDict` (начиная с версии 0.8.1). Параметр `maxCount` устанавливает максимальное количество элементов для получения, по умолчанию 1024.

`ngx.shared.SharedDict.keys([maxCount])`

Возвращает массив ключей словаря `SharedDict`. Параметр `maxCount` устанавливает максимальное количество ключей для получения, по умолчанию 1024.

`ngx.shared.SharedDict.name`

Возвращает имя словаря `SharedDict`, соответствует параметру `zone=` директивы `js_shared_dict_zone` в HTTP или Stream.

`ngx.shared.SharedDict.pop(key)`

Удаляет элемент, связанный с указанным `key`, из словаря `SharedDict`; возвращает значение, связанное с `key`, или `undefined`, если его нет.

`ngx.shared.SharedDict.replace(key, value)`

Заменяет `value` для указанного `key` только если ключ уже существует; возвращает `true`, если значение было успешно заменено, `false`, если ключ не существует в словаре `SharedDict`. Выбрасывает `SharedMemoryError`, если в словаре `SharedDict` недостаточно свободного места. Выбрасывает `TypeError`, если значение `value` имеет другой тип, чем ожидается этот словарь.

`ngx.shared.SharedDict.set(key, value [,timeout])`

Устанавливает `value` для указанного `key`; возвращает этот словарь `SharedDict` (для связывания методов).

Опциональный аргумент `timeout` задается в миллисекундах и переопределяет параметр `timeout` директивы `js_shared_dict_zone` в HTTP или Stream (начиная с версии 0.8.5). Это может быть полезно, когда некоторые ключи ожидают иметь уникальные таймауты.

`ngx.shared.SharedDict.size()`

Возвращает количество элементов для словаря `SharedDict`.

`ngx.shared.SharedDict.type`

Возвращает `string` или `number`, что соответствует типу словаря `SharedDict`, установленному параметром `type=` директивы `js_shared_dict_zone` в HTTP или Stream.

Встроенные объекты

console

- `console.error()`
- `console.info()`
- `console.log()`
- `console.time()`
- `console.timeEnd()`
- `console.warn()`

Объект `console` доступен в Angie начиная с версии 0.8.2, в CLI начиная с версии 0.2.6.

`console.error(msg[, msg2 ...])`

Выводит одно или несколько сообщений об ошибках. Сообщение может быть строкой или объектом.

`console.info(msg[, msg2 ...])`

Выводит одно или несколько информационных сообщений. Сообщение может быть строкой или объектом.

`console.log(msg[, msg2 ...])`

Выводит одно или несколько сообщений журнала. Сообщение может быть строкой или объектом.

`console.time(label)`

Запускает таймер, который может отслеживать, сколько времени занимает операция. Параметр `label` позволяет назвать разные таймеры. Если вызывается `console.timeEnd()` с тем же именем, будет выведено время, прошедшее с начала работы таймера, в миллисекундах.

`console.timeEnd(label)`

Останавливает таймер, ранее запущенный `console.time()`. Параметр `label` позволяет назвать разные таймеры.

`console.warn(msg[, msg2 ...])`

Выводит одно или несколько предупреждающих сообщений. Сообщение может быть строкой или объектом.

crypto

- `crypto.getRandomValues()`
- `crypto.subtle.encrypt()`
- `crypto.subtle.decrypt()`
- `crypto.subtle.deriveBits()`
- `crypto.subtle.deriveKey()`
- `crypto.subtle.digest()`
- `crypto.subtle.exportKey()`
- `crypto.subtle.generateKey()`
- `crypto.subtle.importKey()`
- `crypto.subtle.sign()`
- `crypto.subtle.verify()`

Объект `crypto` — это глобальный объект, который позволяет использовать криптографические функции (начиная с версии 0.7.0).

`crypto.getRandomValues(typedArray)`

Получает криптографически надежные случайные значения. Возвращает тот же массив, переданный как `typedArray`, но с его содержимым, замененным на новые сгенерированные случайные числа. Возможные значения:

`typedArray`

Может быть `Int8Array`, `Int16Array`, `Uint16Array`, `Int32Array` или `Uint32Array`.

`crypto.subtle.encrypt(algorithm, key, data)`

Шифрует `data` с использованием предоставленного `algorithm` и `key`. Возвращает `Promise`, который выполняется `ArrayBuffer`, содержащим зашифрованный текст. Возможные значения:

`algorithm`

Объект, который определяет используемый алгоритм и любые дополнительные параметры, если требуется:

- Для **RSA-OAEP** передайте объект со следующими ключами:

`name`

Строка, должна быть установлена на **RSA-OAEP**:

```
crypto.subtle.encrypt({name: "RSA-OAEP"}, key, data)
```

- Для **AES-CTR** передайте объект со следующими ключами:

`name`

Строка, должна быть установлена на **AES-CTR**.

`counter`

`ArrayBuffer`, `TypedArray` или `DataView` — начальное значение блока счетчика, должно быть длиной 16 байт (размер блока AES). Крайние биты длины этого блока используются для счетчика, остальные используются для поппсе. Например, если `length` установлена на 64, то первая половина `counter` — это поппсе, а вторая половина используется для счетчика.

`length`

Количество битов в блоке счетчика, используемых для фактического счетчика. Счетчик должен быть достаточно большим, чтобы не переполняться.

- Для **AES-CBC** передайте объект со следующими ключами:

`name`

Строка, должна быть установлена на **AES-CBC**.

`iv`

Или вектор инициализации, это `ArrayBuffer`, `TypedArray` или `DataView`, должно быть 16 байт, непредсказуемо и предпочтительно криптографически случайно. Однако это не должно быть секретом, например, оно может быть передано в открытом виде вместе с зашифрованным текстом.

- Для **AES-GCM** передайте объект со следующими ключами:

`name`

Строка, должна быть установлена на **AES-GCM**.

`iv`

Или вектор инициализации, это `ArrayBuffer`, `TypedArray` или `DataView`, должно быть 16 байт и должно быть уникальным для каждой операции шифрования, выполняемой с заданным ключом.

`additionalData`

(опционально) это `ArrayBuffer`, `TypedArray` или `DataView`, которое содержит дополнительные данные, которые не будут зашифрованы, но будут аутентифицированы вместе с зашифрованными данными. Если `additionalData` указан, то же данные должны быть указаны в соответствующем вызове `decrypt()`: если

данные, переданные в вызов `decrypt()`, не совпадают с исходными данными, расшифровка выбросит исключение. Длина бита `additionalData` должна быть меньше $2^{64} - 1$.

`tagLength`

(опционально, по умолчанию 128) — `number`, который определяет размер в битах тега аутентификации, сгенерированного в операции шифрования и используемого для аутентификации в соответствующем расшифровании. Возможные значения: 32, 64, 96, 104, 112, 120 или 128. Спецификация AES-GCM рекомендует, чтобы он был 96, 104, 112, 120 или 128, хотя 32 или 64 бита могут быть приемлемыми в некоторых приложениях.

`key`

`CryptoKey`, которая содержит ключ, который должен быть использован для шифрования.

`data`

`ArrayBuffer`, `TypedArray` или `DataView`, которая содержит данные для шифрования (также известные как открытый текст).

`crypto.subtle.decrypt(algorithm, key, data)`

Расшифровывает зашифрованные данные. Возвращает `Promise` с расшифрованными данными. Возможные значения:

`algorithm`

Объект, который определяет используемый алгоритм и любые дополнительные параметры, если требуется. Значения, указанные для дополнительных параметров, должны совпадать со значениями, переданными в соответствующий вызов `encrypt()`.

- Для **RSA-OAEP** передайте объект со следующими ключами:

`name`

Строка, должна быть установлена на **RSA-OAEP**:

```
crypto.subtle.encrypt({name: "RSA-OAEP"}, key, data)
```

- Для **AES-CTR** передайте объект со следующими ключами:

`name`

Строка, должна быть установлена на **AES-CTR**.

`counter`

`ArrayBuffer`, `TypedArray` или `DataView` — начальное значение блока счетчика, должно быть длиной 16 байт (размер блока AES). Крайние биты длины этого блока используются для счетчика, остальные используются для поппсе. Например, если `length` установлена на 64, то первая половина `counter` — это поппсе, а вторая половина используется для счетчика.

`length`

Количество битов в блоке счетчика, используемых для фактического счетчика. Счетчик должен быть достаточно большим, чтобы не переполняться.

- Для **AES-CBC** передайте объект со следующими ключами:

`name`

Строка, должна быть установлена на **AES-CBC**.

`iv`

Или вектор инициализации, это `ArrayBuffer`, `TypedArray` или `DataView`, должно быть 16 байт, непредсказуемо и предпочтительно криптографически случайно. Однако это не должно быть секретом (например, оно может быть передано в открытом виде вместе с зашифрованным текстом).

- Для **AES-GCM** передайте объект со следующими ключами:

`name`

Строка, должна быть установлена на **AES-GCM**.

`iv`

Или вектор инициализации, это `ArrayBuffer`, `TypedArray` или `DataView`, должно быть 16 байт и должно быть уникальным для каждой операции шифрования, выполняемой с заданным ключом.

`additionalData`

(опционально) это `ArrayBuffer`, `TypedArray` или `DataView`, которое содержит дополнительные данные, которые не будут зашифрованы, но будут аутентифицированы вместе с зашифрованными данными. Если `additionalData` указан, то же данные должны быть указаны в соответствующем вызове `decrypt()`: если данные, переданные в вызов `decrypt()`, не совпадают с исходными данными, расшифровка выбросит исключение. Длина бита `additionalData` должна быть меньше $2^{64} - 1$.

`tagLength`

(опционально, по умолчанию 128) — `number`, который определяет размер в битах тега аутентификации, сгенерированного в операции шифрования и используемого для аутентификации в соответствующем расшифровании. Возможные значения: 32, 64, 96, 104, 112, 120 или 128. Спецификация AES-GCM рекомендует, чтобы он был 96, 104, 112, 120 или 128, хотя 32 или 64 бита могут быть приемлемыми в некоторых приложениях.

`key`

`CryptoKey`, которая содержит ключ, который должен быть использован для расшифровки. Если используется `RSA-OAEP`, это свойство `privateKey` объекта `CryptoKeyPair`.

`data`

`ArrayBuffer`, `TypedArray` или `DataView`, которая содержит данные для расшифровки (также известные как зашифрованный текст).

`crypto.subtle.deriveBits(algorithm, baseKey, length)`

Производит массив битов из базового ключа. Возвращает `Promise`, который будет выполнен `ArrayBuffer`, содержащим производные биты. Возможные значения:

`algorithm`

Объект, который определяет используемый алгоритм производства:

- Для HKDF передайте объект со следующими ключами:

`name`

Строка, должна быть установлена на HKDF.

`hash`

Строка с алгоритмом дайджеста для использования: `SHA-1`, `SHA-256`, `SHA-384` или `SHA-512`.

`salt`

`ArrayBuffer`, `TypedArray` или `DataView`, который представляет случайное или псевдослучайное значение с той же длиной, что и результат функции `digest`. В отличие от входного материала ключа, передаваемого в `deriveKey()`, соль не должна держаться в секрете.

`info`

`ArrayBuffer`, `TypedArray` или `DataView`, который представляет информацию о контексте, зависящую от приложения, используемую для привязки производного ключа к приложению или контексту и позволяющую производить различные ключи для разных контекстов при использовании одного и того же входного материала ключа. Это свойство требуется, но может быть пустым буфером.

- Для PBKDF2 передайте объект со следующими ключами:

`name`

Строка, должна быть установлена на PBKDF2.

hash

Строка с алгоритмом дайджеста для использования: SHA-1, SHA-256, SHA-384 или SHA-512.

salt

`ArrayBuffer`, `TypedArray` или `DataView`, который представляет случайное или псевдослучайное значение не менее 16 байт. В отличие от входного материала ключа, передаваемого в `deriveKey()`, соль не должна держаться в секрете.

iterations

number, который представляет количество раз, которое функция хэша будет выполнена в `deriveKey()`.

- Для ECDH передайте объект со следующими ключами (начиная с версии 0.9.1):

name

Строка, должна быть установлена на ECDH.

public

`CryptoKey`, который представляет открытый ключ другой стороны. Ключ должен быть сгенерирован с использованием той же кривой, что и базовый ключ.

baseKey

`CryptoKey`, который представляет входные данные для алгоритма производства — исходный материал ключа для функции производства: например, для PBKDF2 это может быть пароль, импортированный как `CryptoKey` с использованием `crypto.subtle.importKey()`.

length

Число, представляющее количество битов, которые нужно производить. Для совместимости с браузером число должно быть кратно 8.

`crypto.subtle.deriveKey(algorithm, baseKey, derivedKeyAlgorithm, extractable, keyUsages)`

Производит секретный ключ из главного ключа. Возможные значения:

algorithm

Объект, который определяет используемый алгоритм производства:

- Для HKDF передайте объект со следующими ключами:

name

Строка, должна быть установлена на HKDF.

hash

Строка с алгоритмом дайджеста для использования: SHA-1, SHA-256, SHA-384 или SHA-512.

salt

`ArrayBuffer`, `TypedArray` или `DataView`, который представляет случайное или псевдослучайное значение с той же длиной, что и результат функции `digest`. В отличие от входного материала ключа, передаваемого в `deriveKey()`, соль не должна держаться в секрете.

info

`ArrayBuffer`, `TypedArray` или `DataView`, который представляет информацию о контексте, зависящую от приложения, используемую для привязки производного ключа к приложению или контексту и позволяющую производить различные ключи для разных контекстов при использовании одного и того же входного материала ключа. Это свойство требуется, но может быть пустым буфером.

- Для PBKDF2 передайте объект со следующими ключами:

name

Строка, должна быть установлена на PBKDF2.

hash

Строка с алгоритмом дайджеста для использования: **SHA-1**, **SHA-256**, **SHA-384** или **SHA-512**.

salt

ArrayBuffer, **TypedArray** или **DataView**, который представляет случайное или псевдослучайное значение не менее 16 байт. В отличие от входного материала ключа, передаваемого в **deriveKey()**, соль не должна держаться в секрете.

iterations

number, который представляет количество раз, которое функция хэша будет выполнена в **deriveKey()**.

- Для ECDH передайте объект со следующими ключами (начиная с версии 0.9.1):

name

Строка, должна быть установлена на ECDH.

publicKey

CryptoKey, который представляет открытый ключ другой стороны. Ключ должен быть сгенерирован с использованием той же кривой, что и базовый ключ.

baseKey

CryptoKey, который представляет входные данные для алгоритма производства — исходный материал ключа для функции производства: например, для PBKDF2 это может быть пароль, импортированный как **CryptoKey** с использованием **crypto.subtle.importKey()**.

derivedKeyAlgorithm

Объект, который определяет алгоритм, для которого будет использован производный ключ:

- Для HMAC передайте объект со следующими ключами:

name

Строка, должна быть установлена на HMAC.

hash

Строка с именем функции дайджеста для использования: **SHA-1**, **SHA-256**, **SHA-384** или **SHA-512**.

length

(опционально) это **number**, который представляет длину в битах ключа. Если не указано, длина ключа равна размеру блока выбранной функции хэша.

- Для AES-CTR, AES-CBC или AES-GCM передайте объект со следующими ключами:

name

Строка, должна быть установлена на AES-CTR, AES-CBC или AES-GCM, в зависимости от используемого алгоритма.

length

number, который представляет длину в битах ключа для генерации: 128, 192 или 256.

extractable

Логическое значение, которое указывает, будет ли возможен экспорт ключа.

keyUsages

Array, который указывает, что можно делать с производным ключом. Использование ключей должно быть разрешено алгоритмом, установленным в **derivedKeyAlgorithm**. Возможные значения:

encrypt

Ключ для шифрования сообщений.

decrypt

Ключ для расшифровки сообщений.

sign
 Ключ для подписания сообщений.

verify
 Ключ для проверки подписей.

deriveKey
 Ключ для производства нового ключа.

deriveBits
 Ключ для производства битов.

wrapKey
 Ключ для обертывания ключа.

unwrapKey
 Ключ для разворачивания ключа.

crypto.subtle.digest(algorithm, data)

Генерирует дайджест указанных данных. Принимает как аргументы идентификатор алгоритма дайджеста для использования и данные для дайджеста. Возвращает **Promise**, который будет выполнен дайджестом. Возможные значения:

algorithm
 Строка, которая определяет функцию хэша для использования: **SHA-1** (не для криптографических приложений), **SHA-256**, **SHA-384** или **SHA-512**.

data
ArrayBuffer, **TypedArray** или **DataView**, которая содержит данные для дайджеста.

crypto.subtle.exportKey(format, key)

Экспортирует ключ: принимает ключ как объект **CryptoKey** и возвращает ключ во внешнем, переносимом формате (начиная с версии 0.7.10). Если **format** был **jwk**, то **Promise** выполняется объектом **JSON**, содержащим ключ. В противном случае обещание выполняется **ArrayBuffer**, содержащим ключ. Возможные значения:

format
 Строка, которая описывает формат данных, в котором должен быть экспортирован ключ, может быть следующей:

raw
 Формат данных **raw**.

pkcs8
 Формат **PKCS #8**.

spki
 Формат **SubjectPublicKeyInfo**.

jwk
 Формат **JSON Web Key (JWK)** (начиная с версии 0.7.10).

key
CryptoKey, которая содержит ключ, который должен быть экспортирован.

crypto.subtle.generateKey(algorithm, extractable, usage)

Генерирует новый ключ для симметричных алгоритмов или пару ключей для алгоритмов с открытым ключом (начиная с версии 0.7.10). Возвращает **Promise**, который выполняется сгенерированным ключом как объект **CryptoKey** или **CryptoKeyPair**. Возможные значения:

algorithm
 Объект словаря, который определяет тип ключа для генерации и предоставляет дополнительные параметры, специфичные для алгоритма:

- Для **RSASSA-PKCS1-v1_5**, **RSA-PSS** или **RSA-OAEP** передайте объект со следующими ключами:

name

Строка, должна быть установлена на RSASSA-PKCS1-v1_5, RSA-PSS или RSA-OAEP, в зависимости от используемого алгоритма.

hash

Строка, которая представляет имя функции **digest** для использования, может быть SHA-256, SHA-384 или SHA-512.

- Для ECDSA передайте объект со следующими ключами:

name

Строка, должна быть установлена на ECDSA.

namedCurve

Строка, которая представляет имя эллиптической кривой для использования, может быть P-256, P-384 или P-521.

- Для HMAC передайте объект со следующими ключами:

name

Строка, должна быть установлена на HMAC.

hash

Строка, которая представляет имя функции **digest** для использования, может быть SHA-256, SHA-384 или SHA-512.

length

(опционально) это число, которое представляет длину в битах ключа. Если опущено, длина ключа равна длине дайджеста, сгенерированного выбранной функцией дайджеста.

- Для AES-CTR, AES-CBC или AES-GCM передайте строку, определяющую алгоритм, или объект вида `"name": "ALGORITHM"`, где ALGORITHM — это имя алгоритма.
- Для ECDH передайте объект со следующими ключами (начиная с версии 0.9.1):

name

Строка, должна быть установлена на ECDH.

namedCurve

Строка, которая представляет имя эллиптической кривой для использования, может быть P-256, P-384 или P-521.

extractable

Логическое значение, которое указывает, возможен ли экспорт ключа.

usage

array, который указывает возможные действия с ключом:

encrypt

Ключ для шифрования сообщений.

decrypt

Ключ для расшифровки сообщений.

sign

Ключ для подписания сообщений.

verify

Ключ для проверки подписей.

deriveKey

Ключ для производства нового ключа.

deriveBits

Ключ для производства битов.

wrapKey

Ключ для обертывания ключа.

`unwrapKey`

Ключ для разворачивания ключа.

`crypto.subtle.importKey(format, keyData, algorithm, extractable, keyUsages)`

Импортирует ключ: принимает ключ во внешнем, переносимом формате и дает объект `CryptoKey`. Возвращает `Promise`, который выполняется импортированным ключом как объект `CryptoKey`. Возможные значения:

`format`

Строка, которая описывает формат данных ключа для импорта, может быть следующей:

`raw`

Формат данных `raw`.

`pkcs8`

Формат `PKCS #8`.

`spki`

Формат `SubjectPublicKeyInfo`.

`jwk`

Формат `JSON Web Key (JWK)` (начиная с версии 0.7.10).

`keyData`

Объект `ArrayBuffer`, `TypedArray` или `DataView`, который содержит ключ в указанном формате.

`algorithm`

Объект словаря, который определяет тип ключа для импорта и предоставляет дополнительные параметры, специфичные для алгоритма:

- Для `RSASSA-PKCS1-v1_5`, `RSA-PSS` или `RSA-OAEP` передайте объект со следующими ключами:

`name`

Строка, должна быть установлена на `RSASSA-PKCS1-v1_5`, `RSA-PSS` или `RSA-OAEP`, в зависимости от используемого алгоритма.

`hash`

Строка, которая представляет имя функции `digest` для использования, может быть `SHA-1`, `SHA-256`, `SHA-384` или `SHA-512`.

- Для `ECDSA` передайте объект со следующими ключами:

`name`

Строка, должна быть установлена на `ECDSA`.

`namedCurve`

Строка, которая представляет имя эллиптической кривой для использования, может быть `P-256`, `P-384` или `P-521`.

- Для `HMAC` передайте объект со следующими ключами:

`name`

Строка, должна быть установлена на `HMAC`.

`hash`

Строка, которая представляет имя функции `digest` для использования, может быть `SHA-256`, `SHA-384` или `SHA-512`.

`length`

(опционально) это число, которое представляет длину в битах ключа. Если опущено, длина ключа равна длине дайджеста, сгенерированного выбранной функцией дайджеста.

- Для `AES-CTR`, `AES-CBC` или `AES-GCM` передайте строку, определяющую алгоритм, или объект вида `"name": "ALGORITHM"`, где `ALGORITHM` — это имя алгоритма.

- Для PBKDF2 передайте строку PBKDF2.
- Для HKDF передайте строку HKDF.
- Для ECDH передайте объект со следующими ключами (начиная с версии 0.9.1):

name

Строка, должна быть установлена на ECDH.

namedCurve

Строка, которая представляет имя эллиптической кривой для использования, может быть P-256, P-384 или P-521.

extractable

Логическое значение, которое указывает, возможен ли экспорт ключа.

keyUsages

array, который указывает возможные действия с ключом:

encrypt

Ключ для шифрования сообщений.

decrypt

Ключ для расшифровки сообщений.

sign

Ключ для подписания сообщений.

verify

Ключ для проверки подписей.

deriveKey

Ключ для производства нового ключа.

deriveBits

Ключ для производства битов.

wrapKey

Ключ для обертывания ключа.

unwrapKey

Ключ для разворачивания ключа.

crypto.subtle.sign(algorithm, key, data)

Возвращает **signature** как **Promise**, который выполняется **ArrayBuffer**, содержащим подпись. Возможные значения:

algorithm

Строка или объект, который определяет используемый алгоритм подписи и его параметры:

- Для RSASSA-PKCS1-v1_5 передайте строку, определяющую алгоритм, или объект вида `"name": "ALGORITHM"`.
- Для RSA-PSS передайте объект со следующими ключами:

name

Строка, должна быть установлена на RSA-PSS.

saltLength

Длинное **integer**, которое представляет длину случайной соли для использования, в байтах.

- Для ECDSA передайте объект со следующими ключами:

name

Строка, должна быть установлена на ECDSA.

hash

Идентификатор алгоритма дайджеста для использования, может быть SHA-256, SHA-384 или SHA-512.

- Для HMAC передайте строку, определяющую алгоритм, или объект вида `"name": "ALGORITHM"`.

key

Объект `CryptoKey`, который содержит ключ для использования при подписании. Если алгоритм определяет криптосистему с открытым ключом, это закрытый ключ.

data

Объект `ArrayBuffer`, `TypedArray` или `DataView`, который содержит данные для подписания.

`crypto.subtle.verify(algorithm, key, signature, data)`

Проверяет цифровую подпись; возвращает `Promise`, который выполняется логическим значением: `true`, если подпись действительна, в противном случае `false`. Возможные значения:

algorithm

Строка или объект, который определяет используемый алгоритм и его параметры:

- Для RSASSA-PKCS1-v1_5 передайте строку, определяющую алгоритм, или объект вида `"name": "ALGORITHM"`.
- Для RSA-PSS передайте объект со следующими ключами:

name

Строка, должна быть установлена на RSA-PSS.

saltLength

Длинное `integer`, которое представляет длину случайной соли для использования, в байтах.

- Для ECDSA передайте объект со следующими ключами:

name

Строка, должна быть установлена на ECDSA.

hash

Идентификатор алгоритма дайджеста для использования, может быть SHA-256, SHA-384 или SHA-512.

- Для HMAC передайте строку, определяющую алгоритм, или объект вида `"name": "ALGORITHM"`.

key

Объект `CryptoKey`, который содержит ключ для использования при проверке. Это секретный ключ для симметричного алгоритма и открытый ключ для системы с открытым ключом.

signature

`ArrayBuffer`, `TypedArray` или `DataView`, которая содержит подпись для проверки.

data

Объект `ArrayBuffer`, `TypedArray` или `DataView`, который содержит данные, подпись которых должна быть проверена.

CryptoKey

- `CryptoKey.algorithm`
- `CryptoKey.extractable`
- `CryptoKey.type`
- `CryptoKey.usages`

Объект `CryptoKey` представляет криптографический `key`, полученный из одного из методов `SubtleCrypto`: `crypto.subtle.generateKey()`, `crypto.subtle.deriveKey()`, `crypto.subtle.importKey()`.

`CryptoKey.algorithm`

Возвращает объект, описывающий алгоритм, для которого этот ключ может быть использован, и любые связанные дополнительные параметры (начиная с версии 0.8.0), только для чтения.

`CryptoKey.extractable`

Логическое значение, `true`, если ключ может быть экспортирован (начиная с версии 0.8.0), только для чтения.

`CryptoKey.type`

Строковое значение, которое указывает, какой вид ключа представлен объектом, только для чтения. Возможные значения:

`secret`

Этот ключ — это секретный ключ для использования с симметричным алгоритмом.

`private`

Этот ключ — это закрытая половина `CryptoKeyPair` асимметричного алгоритма.

`public`

Этот ключ — это открытая половина `CryptoKeyPair` асимметричного алгоритма.

`CryptoKey.usages`

Массив строк, указывающих, что этот ключ может быть использован для (начиная с версии 0.8.0), только для чтения. Возможные значения массива:

`encrypt`

Ключ для шифрования сообщений.

`decrypt`

Ключ для расшифровки сообщений.

`sign`

Ключ для подписания сообщений.

`verify`

Ключ для проверки подписей.

`deriveKey`

Ключ для производства нового ключа.

`deriveBits`

Ключ для производства битов.

`CryptoKeyPair`

- `CryptoKeyPair.privateKey`
- `CryptoKeyPair.publicKey`

`CryptoKeyPair` — это объект словаря `WebCrypto API`, который представляет асимметричную пару ключей.

`CryptoKeyPair.privateKey`

Объект `CryptoKey`, который представляет закрытый ключ.

`CryptoKeyPair.publicKey`

Объект `CryptoKey`, который представляет открытый ключ.

njs

- `njs.version`
- `njs.version_number`
- `njs.dump()`
- `njs.memoryStats`
- `njs.on()`

Объект `njs` — это глобальный объект, представляющий текущий экземпляр VM (с версии 0.2.0).

`njs.version`

Возвращает строку с текущей версией NJS (например, "0.7.4").

`njs.version_number`

Возвращает число с текущей версией NJS. Например, "0.7.4" возвращается как 0x000704 (с версии 0.7.4).

`njs.dump(value)`

Возвращает красиво отформатированное строковое представление значения.

`njs.memoryStats`

Объект, содержащий статистику использования памяти для текущего экземпляра VM (с версии 0.7.8).

`size`

Объем памяти в байтах, занятый пулом памяти NJS от операционной системы.

`njs.on(event, callback)`

Регистрирует обработчик для указанного события VM (с версии 0.5.2). Событие может быть одной из следующих строк:

`exit`

Вызывается перед уничтожением VM. Обработчик вызывается без аргументов.

process

- `process.argv`
- `process.env`
- `process.kill()`
- `process.pid`
- `process.ppid`

Объект `process` — это глобальный объект, предоставляющий информацию о текущем процессе (0.3.3).

`process.argv`

Возвращает массив, содержащий аргументы командной строки, переданные при запуске текущего процесса.

`process.env`

Возвращает объект, содержащий переменные окружения пользователя.

Примечание

По умолчанию Angie удаляет все переменные окружения, унаследованные от родительского процесса, за исключением переменной TZ. Используйте директиву `env` для сохранения некоторых унаследованных переменных.

`process.kill(pid, number | string)`

Отправляет сигнал процессу, идентифицируемому `pid`. Имена сигналов — это числа или строки, такие как `SIGINT` или `SIGHUP`. Дополнительную информацию см. в `kill(2)`.

`process.pid`

Возвращает PID текущего процесса.

`process.ppid`

Возвращает PID родительского процесса текущего процесса.

String

По умолчанию все строки в NJS — это Unicode-строки. Они соответствуют строкам ECMAScript, содержащим символы Unicode. До версии 0.8.0 также поддерживались байтовые строки.

Byte Strings (Removed)

Примечание

Начиная с версии 0.8.0, поддержка байтовых строк и методов байтовых строк была удалена. При работе с последовательностями байтов следует использовать объект `Buffer` и свойства `Buffer`, такие как `r.requestBuffer`, `r.rawVariables`.

Байтовые строки содержали последовательность байтов и использовались для сериализации Unicode строк во внешние данные и десериализации из внешних источников. Например, метод `toUTF8()` сериализовал Unicode строку в байтовую строку с использованием кодировки UTF-8. Метод `toBytes()` сериализовал Unicode строку с кодовыми точками до 255 в байтовую строку; в противном случае возвращалось `null`.

Следующие методы были объявлены устаревшими и удалены в версии 0.8.0:

- `String.bytesFrom()` (удалено в 0.8.0, используйте `Buffer.from()`)
- `String.prototype.fromBytes()` (удалено в 0.8.0)
- `String.prototype.fromUTF8()` (удалено в 0.8.0, используйте `TextDecoder`)
- `String.prototype.toBytes()` (удалено в 0.8.0)
- `String.prototype.toString()` с кодировкой (удалено в 0.8.0)
- `String.prototype.toUTF8()` (удалено в 0.8.0, используйте `TextEncoder`)

Web API

TextDecoder

- `TextDecoder()`
- `TextDecoder.prototype.encoding`
- `TextDecoder.prototype.fatal`
- `TextDecoder.prototype.ignoreBOM`
- `TextDecoder.prototype.decode()`

`TextDecoder` создает поток кодовых точек из потока байтов (0.4.3).

`TextDecoder([encoding], options)`

Создает новый объект `TextDecoder` для указанной `encoding`; в настоящее время поддерживается только UTF-8. `options` — это словарь `TextDecoderOptions` со свойством:

fatal

Логический флаг, указывающий, должен ли `TextDecoder.decode()` вызывать исключение `TypeError` при обнаружении ошибки кодирования, по умолчанию `false`.

TextDecoder.prototype.encoding

Возвращает строку с именем кодировки, используемой `TextDecoder()`, только для чтения.

TextDecoder.prototype.fatal

Логический флаг, `true` если режим ошибок является критическим, только для чтения.

TextDecoder.prototype.ignoreBOM

Логический флаг, `true` если маркер порядка байтов игнорируется, только для чтения.

TextDecoder.prototype.decode(buffer, [options])

Возвращает строку с текстом, декодированным из `buffer` посредством `TextDecoder()`. Буфер может быть `ArrayBuffer`. `options` — это словарь `TextDecodeOptions` со свойством:

stream

Логический флаг, указывающий, будут ли следующие данные в последующих вызовах `decode()`: `true` при обработке данных по частям, и `false` для последнего фрагмента или если данные не разбиты на части. По умолчанию `false`.

Пример:

```
>> (new TextDecoder()).decode(new Uint8Array([206,177,206,178]))
\alpha\betaa
```

TextEncoder

- `TextEncoder()`
- `TextEncoder.prototype.encode()`
- `TextEncoder.prototype.encodeInto()`

Объект `TextEncoder` создает поток байтов с кодировкой UTF-8 из потока кодовых точек (0.4.3).

TextEncoder()

Возвращает вновь созданный `TextEncoder`, который будет генерировать поток байтов с кодировкой UTF-8.

TextEncoder.prototype.encode(string)

Кодирует `string` в `Uint8Array` с текстом в кодировке UTF-8.

TextEncoder.prototype.encodeInto(string, uint8Array)

Кодирует `string` в UTF-8, помещает результат в целевой `Uint8Array` и возвращает объект словаря, показывающий ход кодирования. Объект словаря содержит двух членов:

read

Количество единиц UTF-16 кодовых точек из исходной `string`, преобразованных в UTF-8.

written

Количество байтов, измененных в целевом `Uint8Array`.

Таймеры

- `clearTimeout()`
- `setTimeout()`

clearTimeout(timeout)

Отменяет объект `timeout`, созданный `setTimeout()`.

`setTimeout(function, milliseconds[, argument1, argumentN])`

Вызывает `function` после указанного количества `milliseconds`. Можно передать один или несколько опциональных `arguments` в указанную функцию. Возвращает объект `timeout`.

Пример:

```
function handler(v)
{
    // ...
}

t = setTimeout(handler, 12);

// ...

clearTimeout(t);
```

Глобальные функции

- `atob()`
- `btoa()`

`atob(encodedData)`

Декодирует строку данных, которая была закодирована с использованием кодировки Base64. Параметр `encodedData` — это двоичная строка, содержащая данные в кодировке Base64. Возвращает строку, содержащую декодированные данные из `encodedData`.

Подобный метод `btoa()` может использоваться для кодирования и передачи данных, которые иначе могут вызвать проблемы связи, а затем их передачи и использования метода `atob()` для повторного декодирования данных. Например, можно кодировать, передавать и декодировать управляющие символы, такие как значения ASCII от 0 до 31.

Пример:

```
const encodedData = btoa("text to encode"); // encode a string
const decodedData = atob(encodedData); // decode the string
```

`btoa(stringToEncode)`

Создает строку ASCII в кодировке Base64 из двоичной строки. Параметр `stringToEncode` — это двоичная строка для кодирования. Возвращает строку ASCII, содержащую представление `stringToEncode` в Base64.

Метод может использоваться для кодирования данных, которые иначе могут вызвать проблемы связи, их передачи и затем использования метода `atob()` для повторного декодирования данных. Например, можно кодировать управляющие символы, такие как значения ASCII от 0 до 31.

Пример:

```
const encodedData = btoa("text to encode"); // encode a string
const decodedData = atob(encodedData); // decode the string
```

Встроенные модули

Buffer

Объект `Buffer` — это совместимый с Node.js способ работы с двоичными данными. Из-за большого размера файла этот раздел ограничен полным списком методов `Buffer`.

- `Buffer.alloc()`

- `Buffer.allocUnsafe()`
- `Buffer.byteLength()`
- `Buffer.compare()`
- `Buffer.concat()`
- `Buffer.from(array)`
- `Buffer.from(arrayBuffer)`
- `Buffer.from(buffer)`
- `Buffer.from(object)`
- `Buffer.from(string)`
- `Buffer.isBuffer()`
- `Buffer.isEncoding()`
- `buffer[]`
- `buf.buffer`
- `buf.byteOffset`
- `buf.compare()`
- `buf.copy()`
- `buf.equals()`
- `buf.fill()`
- `buf.includes()`
- `buf.indexOf()`
- `buf.lastIndexOf()`
- `buf.length`
- `buf.readIntBE()`
- `buf.readIntLE()`
- `buf.readUIntBE()`
- `buf.readUIntLE()`
- `buf.readDoubleBE()`
- `buf.readDoubleLE()`
- `buf.readFloatBE()`
- `buf.readFloatLE()`
- `buf.subarray()`
- `buf.slice()`
- `buf.swap16()`
- `buf.swap32()`
- `buf.swap64()`
- `buf.toJSON()`
- `buf.toString()`
- `buf.write()`
- `buf.writeIntBE()`

- `buf.writeIntLE()`
- `buf.writeUIntBE()`
- `buf.writeUIntLE()`
- `buf.writeDoubleBE()`
- `buf.writeDoubleLE()`
- `buf.writeFloatBE()`
- `buf.writeFloatLE()`

Подробную документацию по методам Buffer см. в [документации Node.js по Buffer](#).

Crypto

Модуль Crypto предоставляет поддержку криптографической функциональности. Объект модуля Crypto импортируется с использованием `import crypto from 'crypto'`.

Примечание

Начиная с версии 0.7.0, расширенный API криптографии доступен как глобальный объект `crypto`.

- `crypto.createHash()`
- `crypto.createHmac()`

`crypto.createHash(algorithm)`

Создает и возвращает объект Hash, который может использоваться для генерации дайджестов хэшей с использованием заданного `algorithm`. Алгоритм может быть `md5`, `sha1` и `sha256`.

`crypto.createHmac(algorithm, secret key)`

Создает и возвращает объект HMAC, который использует заданный `algorithm` и `secret key`. Алгоритм может быть `md5`, `sha1` и `sha256`.

Hash

- `hash.update()`
- `hash.digest()`

`hash.update(data)`

Обновляет содержимое хэша с заданными `data`.

`hash.digest([encoding])`

Вычисляет дайджест всех данных, переданных с использованием `hash.update()`. Кодировка может быть `hex`, `base64` и `base64url`. Если кодировка не предоставлена, возвращается объект Buffer (0.4.4).

Примечание

До версии 0.4.4 вместо объекта Buffer возвращалась байтовая строка.

`hash.copy()`

Создает копию текущего состояния хэша (с версии 0.7.12).

Пример:

```
import crypto from 'crypto';
```

```
crypto.createHash('sha1').update('A').update('B').digest('base64url');
/* BtlFlCqiamG-GMPiK_GbvKjdK10 */
```

HMAC

- `hmac.update()`
- `hmac.digest()`

`hmac.update(data)`

Обновляет содержимое HMAC с заданными `data`.

`hmac.digest([encoding])`

Вычисляет дайджест HMAC всех данных, переданных с использованием `hmac.update()`. Кодировка может быть `hex`, `base64` и `base64url`. Если кодировка не предоставлена, возвращается объект Buffer (0.4.4).

Примечание

До версии 0.4.4 вместо объекта Buffer возвращалась байтовая строка.

fs

Модуль `fs` предоставляет операции с файловой системой. Объект модуля импортируется с использованием `import fs from 'fs'`.

- `fs.accessSync()`
- `fs.appendFileSync()`
- `fs.mkdirSync()`
- `fs.readdirSync()`
- `fs.readFileSync()`
- `fs.realpathSync()`
- `fs.renameSync()`
- `fs.rmdirSync()`
- `fs.symlinkSync()`
- `fs.unlinkSync()`
- `fs.writeFileSync()`
- `fs.promises.readFile()`
- `fs.promises.appendFile()`
- `fs.promises.writeFile()`
- `fs.promises.readdir()`
- `fs.promises.mkdir()`
- `fs.promises.rmdir()`
- `fs.promises.rename()`
- `fs.promises.unlink()`
- `fs.promises.symlink()`
- `fs.promises.access()`

- `fs.promises.realpath()`

За более подробной документацией по методам `fs` обратитесь к [документации Node.js](#) по `fs`.

Query String

Модуль `Query String` предоставляет методы для парсинга и форматирования строк запроса URL. Объект модуля импортируется с использованием `import qs from 'querystring'`.

- `querystring.decode()`
- `querystring.encode()`
- `querystring.escape()`
- `querystring.parse()`
- `querystring.stringify()`
- `querystring.unescape()`

`querystring.decode()`

Псевдоним для `querystring.parse()`.

`querystring.encode()`

Псевдоним для `querystring.stringify()`.

`querystring.escape(string)`

Выполняет процентное кодирование URL `string` оптимизированным для требований строк запроса URL образом. Метод используется `querystring.stringify()` и не должен использоваться напрямую.

`querystring.parse(string[, separator[, equal[, options]])`

Парсит `string` как строку запроса URL и возвращает объект. Опциональный параметр `separator` (по умолчанию: `&`) указывает подстроку для разделения пар ключ-значение. Опциональный параметр `equal` (по умолчанию: `=`) указывает подстроку для разделения ключей и значений. Опциональный параметр `options` — это объект, который может содержать следующее свойство:

decodeURIComponent

Функция, используемая при декодировании процентно-кодированных символов в строке запроса, по умолчанию: `querystring.unescape()`.

maxKeys

Максимальное количество ключей для парсинга, по умолчанию: 1000. Значение 0 удаляет ограничения на подсчет ключей.

Пример:

```
>> qs.parse('foo=bar&abc=xyz&abc=123')
{
  foo: 'bar',
  abc: ['xyz', '123']
}
```

`querystring.stringify(object[, separator[, equal[, options]])`

Создает строку запроса URL из `object` путем итерации по его собственным свойствам. Опциональный параметр `separator` (по умолчанию: `&`) указывает подстроку для разделения пар ключ-значение. Опциональный параметр `equal` (по умолчанию: `=`) указывает подстроку для разделения ключей и значений. Опциональный параметр `options` — это объект, который может содержать следующее свойство:

encodeURIComponent

Функция, используемая при преобразовании небезопасных для URL символов в процентное кодирование в строке запроса, по умолчанию: `querystring.escape()`.

Пример:

```
>> qs.stringify({ foo: 'bar', baz: ['qux', 'quux'], corge: '' })
'foo=bar&baz=qux&baz=quux&corge='
```

`querystring.unescape(string)`

Выполняет декодирование процентно-кодированных символов URL в `string`. Метод используется `querystring.parse()` и не должен использоваться напрямую.

XML

- `xml.parse()`
- `xml.c14n()`
- `xml.exclusiveC14n()`
- `xml.serialize()`
- `xml.serializeToString()`
- `XMLDoc`
- `XMLNode`
- `XMLAttr`

Модуль XML позволяет работать с XML документами (с версии 0.7.10). Объект модуля XML импортируется с использованием `import xml from 'xml'`.

Пример:

```
import xml from 'xml';

let data = `<to b="bar" a="foo" >Tove</to><from>Jani</from></note>`;
let doc = xml.parse(data);

console.log(doc.note.to.$text) /* 'Tove' */
console.log(doc.note.to.$attr$b) /* 'bar' */
console.log(doc.note.$tags[1].$text) /* 'Jani' */

let dec = new TextDecoder();
let c14n = dec.decode(xml.exclusiveC14n(doc.note));
console.log(c14n) /* '<note><to a="foo" b="bar">Tove</to><from>Jani</from></note>' */

c14n = dec.decode(xml.exclusiveC14n(doc.note.to));
console.log(c14n) /* '<to a="foo" b="bar">Tove</to>' */

c14n = dec.decode(xml.exclusiveC14n(doc.note, doc.note.to /* excluding 'to' */));
console.log(c14n) /* '<note><from>Jani</from></note>' */
```

`parse(string | Buffer)`

Парсит строку или Buffer для XML документа; возвращает объект-обертку `XMLDoc`, представляющий проанализированный XML документ.

`c14n(root_node[, excluding_node])`

Канонизирует `root_node` и его потомков согласно [Canonical XML Version 1.1](#). `root_node` может быть объектом-оберткой `XMLNode` или `XMLDoc` вокруг XML структуры. Возвращает объект Buffer, содержащий канонизированный вывод.

`excluding_node`

Позволяет исключить из вывода часть документа.

`exclusiveC14n(root_node[, excluding_node[, withComments[, prefix_list]]])`

Канонизирует `root_node` и его потомков согласно [Exclusive XML Canonicalization Version 1.0](#).

root_node

Является объектом-оберткой **XMLNode** или **XMLDoc** вокруг XML структуры.

excluding_node

Позволяет исключить из вывода часть документа, соответствующую узлу и его потомкам.

withComments

Логическое значение, по умолчанию **false**. Если **true**, канонизация соответствует **Exclusive XML Canonicalization Version 1.0**. Возвращает объект **Buffer**, содержащий канонизированный вывод.

prefix_list

Опциональная строка с пробелами, разделяющими префиксы пространств имен для пространств имен, которые также должны быть включены в выход.

serialize()

То же самое, что **xml.c14n()** (с версии 0.7.11).

serializeToString()

То же самое, что **xml.c14n()**, за исключением того, что возвращает результат как **string** (с версии 0.7.11).

XMLDoc

Объект-обертка **XMLDoc** вокруг XML структуры, корневой узел документа.

doc.\$root

Корень документа по его имени или **undefined**.

doc.abc

Первый корневой тег, названный **abc**, как объект-обертка **XMLNode**.

XMLNode

Объект-обертка **XMLNode** вокруг узла XML тега.

node.abc

То же самое, что **node.\$tag\$abc**.

node.\$attr\$abc

Значение атрибута узла **abc**, доступно для записи с версии 0.7.11.

node.\$attr\$abc=xyz

То же самое, что **node.setAttribute('abc', xyz)** (с версии 0.7.11).

node.\$attrs

Объект-обертка **XMLAttr** для всех атрибутов узла.

node.\$name

Имя узла.

node.\$ns

Пространство имен узла.

node.\$parent

Родительский узел текущего узла.

node.\$tag\$abc

Первый дочерний тег узла, названный **abc**, доступен для записи с версии 0.7.11.

node.\$tags

Массив всех дочерних тегов.

node.\$tags = [node1, node2, ...]

То же самое, что **node.removeChildren(); node.addChild(node1); node.addChild(node2)** (с версии 0.7.11).

node.\$tags\$abc

Все дочерние теги, названные **abc**, узла, доступны для записи с версии 0.7.11.

`node.$text`

Содержимое узла, доступно для записи с версии 0.7.11.

`node.$text = 'abc'`

То же самое, что `node.setText('abc')` (с версии 0.7.11).

`node.addChild(nd)`

Добавляет XMLNode как дочерний узел к узлу (с версии 0.7.11). `nd` рекурсивно копируется перед добавлением к узлу.

`node.removeAllAttributes()`

Удаляет все атрибуты узла (с версии 0.7.11).

`node.removeAttribute(attr_name)`

Удаляет атрибут, названный `attr_name` (с версии 0.7.11).

`node.removeChildren(tag_name)`

Удаляет все дочерние теги, названные `tag_name` (с версии 0.7.11). Если `tag_name` отсутствует, все дочерние теги удаляются.

`node.removeText()`

Удаляет текстовое значение узла (с версии 0.7.11).

`node.setAttribute(attr_name, value)`

Устанавливает значение для `attr_name` (с версии 0.7.11). Когда значение `null`, атрибут, названный `attr_name`, удаляется.

`node.setText(value)`

Устанавливает текстовое значение для узла (с версии 0.7.11). Когда значение `null`, текст узла удаляется.

XMLAttr

Объект-обертка XMLAttrs вокруг атрибутов узла XML.

`attr.abc`

Значение атрибута `abc`.

zlib

Модуль **zlib** (0.5.2) предоставляет функциональность сжатия и распаковки с использованием **zlib**. Объект модуля импортируется с использованием `import zlib from 'zlib'`.

- `zlib.constants`
- `zlib.deflateRawSync()`
- `zlib.deflateSync()`
- `zlib.inflateRawSync()`
- `zlib.inflateSync()`

`zlib.constants`

Возвращает словарь констант **zlib**.

`zlib.deflateRawSync(data[, options])`

Сжимает `data` с использованием алгоритма Deflate без заголовка **zlib**.

`zlib.deflateSync(data[, options])`

Сжимает `data` с использованием алгоритма Deflate.

`zlib.inflateRawSync(data[, options])`

Распаковывает `data` с использованием алгоритма Deflate без заголовка **zlib**.

`zlib.inflateSync(data[, options])`

Распаковывает `data` с использованием алгоритма Deflate.

Параметр `options` — это объект, который может содержать следующие свойства:

level

Уровень сжатия (по умолчанию: `zlib.constants.Z_DEFAULT_COMPRESSION`).

memLevel

Указывает, сколько памяти должно быть выделено для состояния сжатия (по умолчанию: `zlib.constants.Z_DEFAULT_MEMLEVEL`).

strategy

Настраивает алгоритм сжатия (по умолчанию: `zlib.constants.Z_DEFAULT_STRATEGY`).

windowBits

Устанавливает размер окна (по умолчанию: `zlib.constants.Z_DEFAULT_WINDOWBITS`).

dictionary

Buffer, содержащий предопределенный словарь сжатия.

info

Логическое значение, если `true`, возвращает объект с буфером и движком.

chunkSize

Размер блока для сжатия (по умолчанию: `zlib.constants.Z_DEFAULT_CHUNK`).

Пример:

```
import zlib from 'zlib';

const deflated = zlib.deflateSync('Hello World!');
const inflated = zlib.inflateSync(deflated);

console.log(inflated.toString()); // 'Hello World!'
```

4.10.8.8 Настройка ACME

Модуль ACME в Angie обеспечивает автоматическое получение сертификатов с использованием протокола ACME. Протокол ACME предусматривает несколько способов проверки доменов (также используется термин *верификация*); модуль реализует HTTP-проверку, DNS-проверку, ALPN-проверку, а также проверку с помощью хуков через самостоятельно реализуемый внешний сервис.

Шаги настройки

Общие шаги для включения запроса сертификатов в конфигурации:

- **Настройте ACME-клиент** в блоке `http` с помощью директивы `acme_client`, задающей уникальное имя клиента и другие параметры; можно настроить несколько клиентов ACME.
- **Укажите домены, для которых запрашиваются сертификаты:** для доменных имен, перечисленных во всех директивах `server_name` всех блоков `server` с директивами `acme`, указывающими на один и тот же ACME-клиент, будет выдан единый сертификат.

IP-адреса (IPv4 или IPv6), указанные в `server_name`, также являются допустимыми идентификаторами сертификата, за исключением DNS-проверки.

Обратите внимание: публичные центры сертификации могут выдавать сертификаты для IP-адресов только в рамках определенных профилей сертификатов; например, Let's Encrypt в настоящее время требует профиль, допускающий IP-идентификаторы (см. [документацию по профилям](#)). Чтобы запросить такой профиль, задайте параметр `profile=` директивы `acme_client`.

- **Настройте обработку запросов и вызовов ACME:** это нужно для проверки владения доменом. Способ настройки зависит от способа проверки доменных имен:

Способ	Требования к пользователю	Мультидомены	Домены со звездочкой
<i>HTTP-проверка</i>	Открыть порт 80 (или указанный в <i>acme_http_port</i>) для входящих соединений на сервере Angie.	✓	
<i>DNS-проверка</i>	Открыть порт 53 (или указанный в <i>acme_dns_port</i>) для входящих соединений на сервере Angie. Настроить NS-запись для поддомена <i>_acme-challenge.</i> , направив ее на свой сервер Angie.	✓	✓
<i>ALPN-проверка</i>	Открыть порт 443 (или TLS-порт, используемый сервером Angie) для входящих соединений.	✓	
Проверка хуками	Реализовать внешний сервис (скрипт или приложение), который по команде Angie сможет внести изменения в DNS-зону или разместить специальный ответ на веб-сервере.	✓	✓

- **Настройте SSL с использованием полученного сертификата и ключа:** Модуль делает сертификаты и ключи доступными в виде встроенных переменных, которые можно использовать в конфигурации для заполнения *ssl_certificate* и *ssl_certificate_key*.

Инструкции по настройке SSL см. в разделе Настройка SSL.

💡 Совет

Процесс получения и обновления сертификатов зависит от работы многих служб и может занимать какое-то время. Запаситесь терпением, а в случае возникновения проблем или сомнений обратитесь к отладочному логгу.

Подробности реализации

Здесь ключи и сертификаты клиентов хранятся в **кодировке PEM** в соответствующих подкаталогах каталога, заданного с помощью параметра сборки `--http-acme-client-path`:

```
$ ls /var/lib/angie/acme/example/
account.key  certificate.pem  private.key
```

ℹ Примечание

Эти файлы сохраняются на диске между перезапусками. При запуске клиент повторно использует сохраненный сертификат, если тот еще действителен, вместо запроса нового, что устраняет задержку на выпуск и лишние обращения к серверу СА, на которые могут действовать **ограничения частоты запросов**. В контейнере размещайте каталог хранения (по умолчанию `/var/lib/angie/acme/`) на постоянном томе, чтобы выданные сертификаты сохранялись при пересоздании контейнера; см. запуск Angie в контейнере.

ℹ Примечание

Перезагрузка конфигурации (`angie -s reload`) заново сверяет каждый клиент ACME с сертификатами, сохраненными на диске. Клиент, сертификат которого в данный момент недействителен (в том числе если его предыдущий запрос завершился ошибкой), запрашивается заново немедленно, независимо от `retry_after_error`. Задержка перед повторной попыткой (как и значение `retry_after_error=off`) действует, только пока Angie работает, и не сохраняется после перезагрузки. Поэтому повторные перезагрузки при постоянно неудачных запросах могут приводить к частым обращениям к серверу СА, на которые действуют упомянутые выше ограничения частоты запросов.

Клиенту ACME требуется учетная запись на сервере СА. Для ее создания и управления ею клиент использует закрытый ключ (`account.key`); если ключа у него еще нет, ключ создается при запуске. Затем клиент использует его для регистрации учетной записи на сервере.

Примечание

Если у вас уже есть ключ учетной записи, поместите его в подкаталог клиента перед запуском для повторного использования учетной записи. Файл ключа также можно указать с помощью параметра `account_key` в `acme_client`.

Некоторые СА требуют привязать учетную запись ACME к уже существующей внешней учетной записи с помощью [External Account Binding](#) (EAB), прежде чем выпускать сертификаты. Для этого задайте параметр `eab` директивы `acme_client`, указав идентификатор ключа и MAC-ключ, предоставленные СА; точный синтаксис приведен в описании директивы.

Клиент ACME также использует отдельный ключ (`private.key`) для запросов на подпись сертификата (CSR); если нужно, этот ключ сертификата также создается автоматически при запуске.

При запуске клиент запрашивает сертификат, если его еще нет, подписывая и отправляя CSR для всех доменов, которыми он управляет, серверу СА. Сервер проверяет владение доменом путем HTTP- или DNS-проверки и выдает сертификат, который клиент сохраняет локально (`certificate.pem`).

Как сказано выше, сертификат будет единым для всех доменных имен, для которых используется один и тот же ACME-клиент, то есть потенциально может быть мультидоменным. Список всех имен, для которых выдан сертификат, см. в разделе *Subject Alternative Name* (SAN) полученного сертификата. Проверить его можно в командной строке, например:

```
$ openssl x509 -in certificate.pem -noout -text | grep -A5 "Subject Alternative Name"
```

Когда приближается завершение срока действия сертификата или изменяется список доменов, клиент подписывает и отправляет еще один CSR на сервер СА. Сервер снова проверяет владение и выдает новый сертификат, который клиент устанавливает локально, заменяя предыдущий.

В конфигурации полученный сертификат и соответствующий ключ доступны через префиксные переменные `$acme_cert_<имя>` и `$acme_cert_key_<имя>`. Их значения — содержимое соответствующих файлов, которое следует использовать с директивами `ssl_certificate` и `ssl_certificate_key`, например:

```
server {
    listen 443 ssl;

    server_name example.com www.example.com;
    acme example;

    ssl_certificate $acme_cert_example;
    ssl_certificate_key $acme_cert_key_example;
}
```

Примечание

Клиент ACME определяет адрес сервера CA по его имени через *resolver* того же контекста, который по умолчанию обращается к DNS-серверам из `/etc/resolv.conf`; задавайте директиву явно, только если нужны другие серверы или параметры. На узле без поддержки IPv6 resolver может все равно отправлять AAAA-запросы (IPv6) для сервера CA и не суметь подключиться; отключите их параметром `ipv6=off`, например `resolver conf ipv6=off`;

Сбор доменов и использование сертификата

Директива *acme* служит только для сбора доменных имен для запросов сертификатов. Она не определяет, где можно использовать сертификат: любой блок **server** может ссылаться на полученный сертификат через переменную `$acme_cert_<имя>`, независимо от того, содержит ли блок директиву *acme*.

Например, если у вас есть блок **server** с маской, который уже охватывает все поддомены, дополнительные блоки **server** для конкретных поддоменов не нуждаются в директиве *acme*:

```
http {

    acme_client example https://acme-v02.api.letsencrypt.org/directory
        challenge=dns;

    # В этом блоке перечислены домены для запроса сертификата
    server {

        listen 443 ssl;

        server_name example.com *.example.com;
        acme example;

        ssl_certificate $acme_cert_example;
        ssl_certificate_key $acme_cert_key_example;
    }

    # Этот блок использует тот же сертификат, но не добавляет
    # свой server_name в запрос на сертификат
    server {

        listen 443 ssl;

        server_name app.example.com;

        ssl_certificate $acme_cert_example;
        ssl_certificate_key $acme_cert_key_example;
    }
}
```

Явный список доменов

Чтобы точно контролировать набор доменных имен в сертификате, не полагаясь на автоматический сбор из всех блоков **server**, создайте отдельный блок **server**, содержащий только директивы *server_name* и *acme*. Чтобы этот блок не обрабатывал реальный трафик, привяжите его к Unix-сокету:

```
# Отдельный блок, определяющий список доменов сертификата
server {
```

```
listen unix:/tmp/acme_example.sock;

server_name example.com www.example.com;
acme example;
}
```

Другие блоки **server** могут затем использовать сертификат через переменную `$acme_cert_<имя>`, не влияя на то, какие домены запрашиваются.

Отдельные сертификаты для разных доменов

Каждый *acme_client* управляет одним сертификатом. Чтобы получить несколько независимых сертификатов (например, для несвязанных доменов, которым не следует использовать общий сертификат), настройте отдельный *acme_client* для каждого из них в блоке **http** и укажите в директиве *acme* каждого блока **server** соответствующий клиент по имени:

```
http {

    # Два независимых клиента, каждый управляет своим сертификатом
    acme_client shop https://acme-v02.api.letsencrypt.org/directory
        challenge=http;

    acme_client blog https://acme-v02.api.letsencrypt.org/directory
        challenge=http;

    server {

        listen 443 ssl;

        server_name shop.example.com www.shop.example.com;
        acme shop;

        ssl_certificate $acme_cert_shop;
        ssl_certificate_key $acme_cert_key_shop;
    }

    server {

        listen 443 ssl;

        server_name blog.example.com;
        acme blog;

        ssl_certificate $acme_cert_blog;
        ssl_certificate_key $acme_cert_key_blog;
    }
}
```

HTTP-проверка

Проверка выполняется автоматически. Когда Angie заказывает сертификат, АСМЕ-сервер запрашивает по HTTP файл-токен по адресу `/.well-known/acme-challenge/<TOKEN>`. Angie отвечает на такие запросы сам, поэтому отдавать этот путь вручную не требуется. Получив файл, АСМЕ-сервер сверяет токен с выданным значением и, если они совпадают, засчитывает проверку домена.

Пример конфигурации

Здесь АСМЕ-клиент с именем `example` управляет сертификатами для `example.com` и `www.example.com` (учтите, что wildcard-сертификаты не поддерживаются при HTTP-проверке):

```
http {

    acme_client example https://acme-v02.api.letsencrypt.org/directory;

    server {

        listen 80; # Не обязательно: если порт HTTP-проверки
                  # никто не слушает, модуль откроет его сам
                  # (см. директиву 'acme_http_port')

        listen 443 ssl;

        server_name example.com www.example.com;
        acme example;

        ssl_certificate $acme_cert_example;
        ssl_certificate_key $acme_cert_key_example;
    }
}
```

Как уже отмечалось, порт 80 должен быть открыт для приема вызовов АСМЕ по HTTP. Если ни один сервер не слушает порт HTTP-проверки, модуль создает отдельный слушающий сокет на порту 80 (или указанном в `acme_http_port`). Отдельный блок `server` не требуется.

DNS-проверка

Проверка выполняется автоматически, но требует настройки DNS с вашей стороны. Когда Angie заказывает сертификат, АСМЕ-сервер отправляет **DNS-запрос** TXT-записи в поддомене `_acme-challenge.` проверяемого домена. Angie отвечает на такие запросы сам: он выступает авторитетным сервером имен для этого поддомена и возвращает TXT-запись с тем значением, которого ждет АСМЕ-сервер. АСМЕ-сервер сверяет полученную запись с выданным значением и, если они совпадают, засчитывает проверку домена.

Чтобы запрос дошел до Angie, DNS вашего домена должен делегировать поддомен `_acme-challenge.` серверу Angie; нужные записи приведены ниже. По умолчанию Angie отвечает с очень небольшим TTL, а некоторые DNS-провайдеры отфильтровывают ответы с таким низким TTL — если проверка не завершается, увеличьте его директивой `acme_dns_ttl`.

DNS-проверка подтверждает контроль над доменным именем, поэтому не может проверять идентификаторы в виде IP-адресов: для клиента с `challenge=dns` Angie игнорирует IP-адрес в `server_name` и выводит предупреждение при запуске. Блок `server`, в котором указан IP-адрес, должен по-прежнему содержать хотя бы одно доменное имя, которое клиент может проверить.

Примечание

Сервер Angie должен быть доступен из интернета на UDP-порту 53 (или указанном в `acme_dns_port`). Если сервер находится за межсетевым экраном, убедитесь, что этот порт открыт для входящих соединений.

Например, чтобы подтвердить домен `example.com`, используя сервер Angie с IP-адресом 203.0.113.10, DNS-конфигурация вашего домена должна включать следующие записи:

<code>_acme-challenge.example.com.</code>	60	IN	NS	<code>ns.example.com.</code>
<code>ns.example.com.</code>	60	IN	A	203.0.113.10

Эта конфигурация делегирует разрешение DNS для `_acme-challenge.example.com` на `ns.example.com`, обеспечивая доступность `ns.example.com` путем сопоставления с IP-адресом (203.0.113.10).

Предупреждение

Распространение изменений NS-записей может занимать от нескольких минут до 48 часов в зависимости от TTL и DNS-провайдера. Рекомендуется проверить корректность настройки перед запросом сертификата.

Для проверки корректности настройки DNS можно использовать следующие команды:

```
$ dig NS _acme-challenge.example.com +short # Проверка NS-записи для поддомена _acme-
→challenge

ns.example.com.

$ dig A ns.example.com +short # Проверка A-записи для сервера имен

203.0.113.10

$ nc -zv 203.0.113.10 53 # Проверка доступности DNS-сервера на порту 53
```

Этот способ позволяет запрашивать wildcard-сертификаты, например сертификат, включающий запись `*.example.com` в разделе *Subject Alternative Name* (SAN). Чтобы в явной форме запросить сертификат для поддомена, например `www.example.com`, следует отдельно подтвердить этот поддомен описанным выше способом.

Предупреждение

Применимость данного сценария во многом зависит от возможностей, предоставляемых вашим DNS-провайдером; некоторые провайдеры не позволяют выполнять такие настройки.

Пример конфигурации

В целом, конфигурация схожа с примером из предыдущего раздела. Нет необходимости в настройках, специфичных для HTTP; вместо этого достаточно установить `challenge=dns` для директивы `acme_client`.

Здесь АСМЕ-клиент с именем `example` управляет сертификатами для `example.com` и `*.example.com`:

```
http {

    acme_client example https://acme-v02.api.letsencrypt.org/directory
        challenge=dns;

    server {

        server_name example.com *.example.com;
        acme example;

        ssl_certificate $acme_cert_example;
        ssl_certificate_key $acme_cert_key_example;
    }
}
```

ALPN-проверка

Проверка работает автоматически. ACME-сервер устанавливает TLS-соединение и запрашивает через ALPN протокол `acme-tls/1`. Модуль отдает временный сертификат для валидационного запроса.

Чтобы использовать этот способ, задайте `challenge=alpn` в директиве `acme_client` и убедитесь, что ваш TLS-слушатель доступен на порту 443 (или на используемом TLS-порту).

Пример конфигурации

Конфигурация аналогична предыдущим разделам; достаточно задать `challenge=alpn` для директивы `acme_client` и убедиться, что TLS-сервер доступен на порту 443.

Здесь ACME-клиент с именем `example` управляет сертификатом для `example.com` и `www.example.com`:

```
http {

    acme_client example https://acme-v02.api.letsencrypt.org/directory
        challenge=alpn;

    server {

        listen 443 ssl;

        server_name example.com www.example.com;
        acme example;

        ssl_certificate $acme_cert_example;
        ssl_certificate_key $acme_cert_key_example;
    }
}
```

Проверка с помощью хуков

В отличие от предыдущих способов, эта проверка требует дополнительных усилий. Здесь ACME-сервер производит обычную HTTP-проверку или DNS-проверку, но обращается не к самому серверу Angie, а к внешнему сервису, которым сервер Angie управляет с помощью вызовов-хуков (`acme_hook`). В свою очередь, этот сервис настраивает отдельный DNS- или HTTP-сервер, куда и направляются запросы ACME-сервера.

Получив ожидаемый ответ от настроенного таким образом DNS- или HTTP-сервера, ACME-сервер подтверждает, что домен принадлежит клиенту.

Когда для выпуска или обновления сертификата требуется проверка домена, Angie формирует внутренний запрос к именованному `location`, содержащему директиву `acme_hook`. Способ обработки этого запроса полностью зависит от других директив, заданных в том же `location`.

Общая схема такова:

1. Создайте именованный `location` с директивой `acme_hook`.
2. Настройте обработчик запроса в том же `location` с помощью модуля, подходящего для вашей конфигурации: `fastcgi_pass` для FastCGI, `proxy_pass` для HTTP, `cgi_pass` для CGI-скриптов и т. д.
3. Передайте обработчику переменные ACME с помощью поддерживаемого им механизма, например, `fastcgi_param` для FastCGI или `cgi_set_var` для CGI.

Обработчик должен возвращать код состояния `2xx`, который можно передать через заголовок `Status`. Любой другой код от хука `add` прерывает попытку обновления сертификата; от хука `remove`

тот же код только записывается в журнал как предупреждение, и обновление продолжается. Вывод обработчика игнорируется.

Минимальная конфигурация

Независимо от используемого обработчика, `location` для хука имеет следующую структуру:

```
location @acme_hook_location {

    acme_hook example;

    # Директива обработчика (fastcgi_pass, proxy_pass, cgi_pass, ...)
    # Передайте переменные ACME с помощью механизма обработчика:
    # ACME_HOOK - $acme_hook_name ("add" или "remove")
    # ACME_CHALLENGE - $acme_hook_challenge ("dns" или "http")
    # ACME_DOMAIN - $acme_hook_domain
    # ACME_TOKEN - $acme_hook_token
    # ACME_KEYAUTH - $acme_hook_keyauth
}
```

При DNS-проверке обработчик должен использовать `ACME_HOOK` для определения действия: если значение `add`, создать TXT-запись для `_acme-challenge.ACME_DOMAIN` со значением из `ACME_KEYAUTH`; если значение `remove`, удалить эту запись.

Пример с FastCGI

Здесь настраивается ACME-клиент `example` для подтверждения домена при помощи DNS-вызова, на что указывает параметр `challenge=dns` директивы `acme_client`.

Блок `server` применяется ко всем поддоменам `example.com` (например, `*.example.com`) и использует ACME-клиент `example` для управления сертификатами, что указано в директиве `acme`.

Именованный блок `location` обрабатывает вызовы хуков. Директива `acme_hook` связывает его с ACME-клиентом `example`. Запросы хуков отправляются на локальный FastCGI-сервер на порту 9000 с помощью `fastcgi_pass`. Директивы `fastcgi_param` передают переменные ACME во внешний сервис.

```
acme_client example https://acme-v02.api.letsencrypt.org/directory
    challenge=dns;

server {

    listen 80;

    server_name *.example.com;

    acme example;

    ssl_certificate $acme_cert_example;
    ssl_certificate_key $acme_cert_key_example;

    location @acme_hook_location {

        acme_hook example;

        fastcgi_pass localhost:9000;

        fastcgi_param ACME_CLIENT $acme_hook_client;
        fastcgi_param ACME_HOOK $acme_hook_name;
    }
}
```

```

    fastcgi_param ACME_CHALLENGE $acme_hook_challenge;
    fastcgi_param ACME_DOMAIN $acme_hook_domain;
    fastcgi_param ACME_TOKEN $acme_hook_token;
    fastcgi_param ACME_KEYAUTH $acme_hook_keyauth;

    include fastcgi.conf;
}
}

```

Пример соответствующего внешнего FastCGI-сервиса на Perl:

```

#!/usr/bin/perl

use strict; use warnings;

use FCGI;

my $socket = FCGI::OpenSocket(":9000", 5);
my $request = FCGI::Request(\*STDIN, \*STDOUT, \*STDERR, \%ENV, $socket);

while ($request->Accept() >= 0) {
    print "\r\n";

    my $client = $ENV{ACME_CLIENT};
    my $hook = $ENV{ACME_HOOK};
    my $challenge = $ENV{ACME_CHALLENGE};
    my $domain = $ENV{ACME_DOMAIN};
    my $token = $ENV{ACME_TOKEN};
    my $keyauth = $ENV{ACME_KEYAUTH};

    if ($hook eq 'add') {

        DNS_set_TXT_record("_acme-challenge.$domain.", $keyauth);

    } elsif ($hook eq 'remove') {

        DNS_clear_TXT_record("_acme-challenge.$domain.");

    }
};

FCGI::CloseSocket($socket);

```

Здесь `DNS_set_TXT_record()` и `DNS_clear_TXT_record()` — функции, предположительно добавляющие и удаляющие TXT-записи в конфигурации некоего внешнего DNS-сервера, к которому и обратится ACME-сервер. В этих записях должны содержаться переданные сервером Angie данные, что позволит внешнему DNS-серверу успешно пройти проверку, аналогичную описанной в разделе *DNS-проверка*. Подробности реализации таких функций выходят за рамки этого руководства; так, например, передавать параметры можно и через URI запроса:

```

# ...

location @acme_hook_location {

    acme_hook example uri=/acme_hook/$acme_hook_name?domain=$acme_hook_domain&key=
    ↪$acme_hook_keyauth;

    fastcgi_pass localhost:9000;
}

```

```
fastcgi_param REQUEST_URI $request_uri;
fastcgi_param ACME_CLIENT $acme_hook_client;
fastcgi_param ACME_CHALLENGE $acme_hook_challenge;
fastcgi_param ACME_TOKEN $acme_hook_token;

include fastcgi.conf;
}
```

Пример с PHP-FPM

Еще один пример, с использованием PHP-FPM:

```
location @acme_hook_location {

    acme_hook example;
    root /var/www/dns;
    fastcgi_pass unix:/run/php-fpm/php-dns.sock;
    fastcgi_index hook.php;
    fastcgi_param SCRIPT_FILENAME /var/www/dns/hook.php;
    include fastcgi_params;

    fastcgi_param ACME_CLIENT $acme_hook_client;
    fastcgi_param ACME_HOOK $acme_hook_name;
    fastcgi_param ACME_CHALLENGE $acme_hook_challenge;
    fastcgi_param ACME_DOMAIN $acme_hook_domain;
    fastcgi_param ACME_TOKEN $acme_hook_token;
    fastcgi_param ACME_KEYAUTH $acme_hook_keyauth;
}
```

```
[dns]
listen = /run/php-fpm/php-dns.sock
listen.mode = 0666
user = angie
group = angie
chdir = /var/www/dns
# ...
```

Переданные параметры доступны в PHP через `$_SERVER['...']`.

Пример с CGI и octoDNS

Здесь хук — это скрипт, запускаемый модулем CGI; скрипт добавляет и удаляет TXT-запись `_acme-challenge` через `octoDNS`, инструмент для описания DNS как кода: зона задается в YAML и синхронизируется с DNS-провайдером. В примере используется Cloudflare, но точно так же подойдет любой другой провайдер `octoDNS`, если объявить его в `providers` и указать в `targets`. У API-токена должны быть права `Zone:Read`, `DNS:Read` и `DNS>Edit` для этой зоны.

Модуль подключается в контексте `main`. ACME-клиент `example` заказывает один сертификат для `example.com` и `*.example.com`, подтверждая домен при помощи DNS-вызовов, на что указывает параметр `challenge=dns` директивы `acme_client`. Вызовы хуков обрабатывает именованный блок `location`: директива `acme_hook` связывает его с клиентом, директивы `cgi_set_var` передают переменные ACME под стандартными именами, а `cgi_pass` запускает скрипт при каждом вызове. Из этих переменных скрипт использует `ACME_CHALLENGE`, `ACME_HOOK`, `ACME_DOMAIN` и `ACME_KEYAUTH`.

```
# в контексте main
load_module modules/nginx_http_cgi_module.so;

acme_client example https://acme-v02.api.letsencrypt.org/directory
```

```
challenge=dns;

server {

    listen 443 ssl;

    server_name example.com *.example.com;

    acme example;

    ssl_certificate $acme_cert_example;
    ssl_certificate_key $acme_cert_key_example;

    location @acme_hook_location {

        acme_hook example;

        cgi_set_var ACME_CLIENT $acme_hook_client;
        cgi_set_var ACME_HOOK $acme_hook_name;
        cgi_set_var ACME_CHALLENGE $acme_hook_challenge;
        cgi_set_var ACME_DOMAIN $acme_hook_domain;
        cgi_set_var ACME_TOKEN $acme_hook_token;
        cgi_set_var ACME_KEYAUTH $acme_hook_keyauth;

        cgi_pass /usr/share/angie/cgi-bin/acme-octodns;

    }
}
```

octoDNS собирает зону из источников по порядку и объединяет их, поэтому хук хранит проверочную запись в отдельном каталоге-источнике `/etc/octodns/acme`, и обновление сертификата никогда не переписывает файл зоны в `/etc/octodns/zones`. Параметр `ignore_missing_zones` (octoDNS 1.16 и новее) разрешает каталогу `/etc/octodns/acme` оставаться пустым между обновлениями. Сам файл зоны не должен содержать записей `_acme-challenge`.

Список 1: `/etc/octodns/octodns.yaml`

```
providers:
  config:
    class: octodns.provider.yaml.YamlProvider
    directory: /etc/octodns/zones
  acme:
    class: octodns.provider.yaml.YamlProvider
    directory: /etc/octodns/acme
    ignore_missing_zones: true
  cloudflare:
    class: octodns.cloudflare.CloudflareProvider
    token: YOUR_API_TOKEN

zones:
  example.com.:
    sources:
      - config
      - acme
    targets:
      - cloudflare
```

В файле зоны находятся записи, которые вы ведете сами:

Список 2: /etc/octodns/zones/example.com.yaml

```
':
  type: A
  value: 203.0.113.10
www:
  type: A
  value: 203.0.113.10
```

Чтобы проверить конфигурацию octoDNS вручную, запустите `octodns-sync` без `--doit`.

Скрипт читает переменные ACME из своего окружения и записывает проверочную запись в источник acme:

Скрипт acme-octodns

Список 3: /usr/share/angie/cgi-bin/acme-octodns

```
#!/opt/octodns/bin/python3
# Хук DNS-01 для ACME-клиента Angie: хранит TXT-запись _acme-challenge
# в отдельном каталоге-источнике octodns и публикует ее через octodns-sync.

import fcntl
import os
import subprocess
import sys
import time

import dns.message
import dns.query
import dns.resolver
import yaml

CONFIG = "/etc/octodns/octodns.yaml"
ACME_DIR = "/etc/octodns/acme"
SYNC = "/opt/octodns/bin/octodns-sync"
TTL = 120
WAIT = int(os.environ.get("ACME_WAIT", "300"))

def respond(code, message=""):
    if message:
        print(message, file=sys.stderr)
    sys.stdout.write(f"Status: {code}\r\n\r\n")
    sys.exit(0)

def find_zone(domain):
    with open(CONFIG) as f:
        zones = [z.rstrip(".") for z in yaml.safe_load(f)["zones"]]
    match = [z for z in zones if domain == z or domain.endswith("." + z)]
    if not match:
        respond(500, f"no zone for {domain} in {CONFIG}")
    return max(match, key=len)

def update(zone, name, value, add):
    # В одной записи может быть несколько значений: сертификат для example.com
    # и *.example.com проверяется двумя TXT-значениями с одним именем.
    path = f"{ACME_DIR}/{zone}.yaml"
    with open(f"{ACME_DIR}/.lock", "w") as lock:
```

```

fcntl.flock(lock, fcntl.LOCK_EX)
data = {}
if os.path.exists(path):
    with open(path) as f:
        data = yaml.safe_load(f) or {}
values = set(data.get(name, {}).get("values", []))
(values.add if add else values.discard)(value)
if values:
    data[name] = {"type": "TXT", "ttl": TTL, "values": sorted(values)}
else:
    data.pop(name, None)
if data:
    with open(path, "w") as f:
        yaml.safe_dump(data, f)
elif os.path.exists(path):
    os.remove(path)
cmd = [SYNC, f"--config-file={CONFIG}", "--doit", zone + "."]
run = subprocess.run(cmd, capture_output=True, text=True)
if run.returncode != 0:
    respond(500, "octodns-sync failed:\n" + run.stdout + run.stderr)

def has_txt(server, fqdn, value):
    try:
        reply = dns.query.udp(dns.message.make_query(fqdn, "TXT"), server, timeout=3)
    except Exception:
        return False
    return any(value == b"".join(getattr(r, "strings", ())).decode()
               for rrset in reply.answer for r in rrset)

def wait_visible(zone, fqdn, value):
    # ACME-сервер обращается к авторитетным серверам зоны, поэтому опрашиваем
    # их напрямую: резолвер мог закэшировать отсутствие записи.
    deadline = time.monotonic() + WAIT
    while time.monotonic() < deadline:
        try:
            servers = [str(a) for ns in dns.resolver.resolve(zone, "NS")
                       for a in dns.resolver.resolve(str(ns.target), "A")]
            if servers and all(has_txt(s, fqdn, value) for s in servers):
                return True
        except Exception:
            pass
        time.sleep(5)
    return False

def main():
    hook = os.environ.get("ACME_HOOK")
    domain = os.environ.get("ACME_DOMAIN", "")
    value = os.environ.get("ACME_KEYAUTH", "")
    if os.environ.get("ACME_CHALLENGE") != "dns" or hook not in ("add", "remove"):
        respond(500, "the hook supports only DNS validation")
    zone = find_zone(domain)
    rel = domain[: -len(zone) - 1]
    name = "_acme-challenge" + (f".{rel}" if rel else "")
    update(zone, name, value, hook == "add")
    if hook == "add" and WAIT and not wait_visible(zone, f"{name}.{zone}.", value):
        update(zone, name, value, False)
        respond(500, f"{name}.{zone} is not visible on the name servers after {WAIT}")

```



```
→ s")
    respond(200)

try:
    main()
except Exception as e:
    respond(500, f"{type(e).__name__}: {e}")
```

Как только хук `add` вернет код `2xx`, Angie попросит ACME-сервер выполнить проверку и будет опрашивать результат в течение 60 секунд. Распространения записи в DNS Angie не дожидается, поэтому ждать приходится самому скрипту. Он опрашивает авторитетные серверы имен зоны напрямую, поскольку резолвер мог закэшировать отсутствие записи, и завершается, как только значение отдадут все серверы; по умолчанию он ждет не дольше 300 секунд.

Если значение не появится вовремя, скрипт отвечает **Status: 500**, Angie прерывает попытку и повторяет ее через интервал, заданный параметром `retry_after_error`. После неудачного `add` Angie не вызывает хук `remove`, поэтому скрипт сам удаляет свое значение, прежде чем ответить.

Предел ожидания задается переменной окружения `ACME_WAIT`. В показанном выше `location` для хука она не задается; чтобы изменить предел, добавьте ее туда, передав значение через переменную, как и остальные:

```
set $acme_wait 600;
cgi_set_var ACME_WAIT $acme_wait;
```

Значение 0 отключает ожидание — это подходит для собственных авторитетных серверов, на которых изменение через API видно сразу.

Переменная `$acme_hook_domain` приходит без префикса `*`, поэтому сертификат для `example.com` и `*.example.com` порождает два вызова `add` для одного и того же имени `_acme-challenge.example.com`. Скрипт хранит список значений в одной записи, а при `remove` удаляет только значение из этого вызова; когда значений не остается, он удаляет и саму запись, и хранивший ее файл в `/etc/octodns/acme`.

Скрипт выполняется от имени пользователя рабочих процессов Angie — `angie`. Ему нужен доступ на чтение к конфигурации octoDNS, где хранится токен, и на запись в `/etc/octodns/acme`; больше ничего. CGI-процесс не наследует окружение Angie, поэтому токен хранится в файле конфигурации, а скрипт использует абсолютные пути.

Стандартный поток ошибок скрипта попадает в журнал ошибок Angie на уровне `warn` — именно там разбирают сбои хуков; эти сообщения видны только при уровне `warn` или менее серьезном.

Чтобы все это настроить в системе на основе Debian:

1. Установите модуль CGI:

```
$ sudo apt-get install angie-module-cgi
```

2. Установите octoDNS и провайдер Cloudflare в отдельное виртуальное окружение:

```
$ sudo apt-get install python3-venv
$ sudo python3 -m venv /opt/octodns
$ sudo /opt/octodns/bin/pip install 'octodns>=1.16' octodns-cloudflare
```

3. Создайте каталоги; пакет модуля не создает `/usr/share/angie/cgi-bin`:

```
$ sudo install -d /etc/octodns/zones /usr/share/angie/cgi-bin
$ sudo install -d -o angie -g angie -m 750 /etc/octodns/acme
```

4. Создайте `/etc/octodns/octodns.yaml` и `/etc/octodns/zones/example.com.yaml`, как показано выше, и ограничьте доступ к конфигурации группой `angie`, поскольку в ней хранится API-токен:

```
$ sudo chgrp angie /etc/octodns/octodns.yaml
$ sudo chmod 640 /etc/octodns/octodns.yaml
```

5. Сохраните показанный выше скрипт под именем `acme-octodns` и установите его:

```
$ sudo install -m 755 acme-octodns /usr/share/angie/cgi-bin/
```

6. Добавьте показанную выше конфигурацию, проверьте ее и перезагрузите Angie:

```
$ sudo angie -t
$ sudo kill -HUP $(cat /run/angie.pid)
```

Сразу после перезагрузки клиент запрашивает сертификат; его состояние и статус сертификата видны в разделе API `/status/http/acme_clients/`, а полученный сертификат сохраняется в каталоге хранения клиента.

АСМЕ в потоковом модуле

Потоковый модуль АСМЕ позволяет автоматизировать выпуск и использование сертификатов для TCP-трафика. Для его корректной работы необходимо сначала настроить HTTP-аналог: АСМЕ-клиент должен быть объявлен в контексте `http`, а сам блок `stream` должен располагаться *после* блока `http` в конфигурации.

Пример конфигурации

По умолчанию для получения сертификатов используется режим HTTP-проверки. Для него, как упоминалось в разделе *HTTP-проверка*, нужен HTTP-сервер, слушающий на порту 80:

```
# HTTP-часть
http {

    # АСМЕ-клиент для потоковой части
    acme_client example https://acme-v02.api.letsencrypt.org/directory;

    # Сервер для HTTP-проверки
    server {

        listen 80;
        return 444;
    }
}

# Потоковая часть
stream {

    server {

        listen 12345 ssl;
        proxy_pass backend_upstream;

        ssl_certificate $acme_cert_example;
        ssl_certificate_key $acme_cert_key_example;

        server_name example.com www.example.com;
        acme example; # ссылка на АСМЕ-клиент, определенный в HTTP-части
    }

    upstream backend_upstream {
```

```
server 127.0.0.1:54321;
}
}
```

Также можно использовать DNS-проверку, настроив `challenge=dns` в директиве `acme_client`; тогда сервер будет не нужен.

Миграция с certbot

Если до перехода с nginx на Angie вы использовали `certbot` для получения и продления SSL-сертификатов от центра сертификации Let's Encrypt, выполните следующие шаги, чтобы перейти к использованию нашего модуля ACME.

Предположим, вы настроили сертификаты следующим образом:

```
$ sudo certbot --nginx -d example.com -d www.example.com
```

Автоматически созданная при этом конфигурация обычно находится в файле `/etc/nginx/sites-available/example.conf` и выглядит приблизительно так:

```
server {
    listen 80;
    server_name example.com www.example.com;
    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl;
    server_name example.com www.example.com;

    root /var/www/example;
    index index.html;

    ssl_certificate /etc/letsencrypt/live/example.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/example.com/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
}
```

В примере выше выделены строки, которые потребуется изменить. В зависимости от ваших обстоятельств и предпочтений настройте HTTP-проверку или DNS-проверку с помощью модуля ACME.

Итоговая конфигурация Angie может выглядеть приблизительно так:

```
http {
    acme_client example https://acme-v02.api.letsencrypt.org/directory;

    server {
        listen 80;
        server_name example.com www.example.com;
        return 301 https://$host$request_uri;
    }

    server {
```

```
listen 443 ssl;
server_name example.com www.example.com;

root /var/www/example;
index index.html;

acme example;

ssl_certificate $acme_cert_example;
ssl_certificate_key $acme_cert_key_example;
}
```

Не забудьте перезагрузить конфигурацию после изменения:

```
$ sudo kill -HUP $(cat /run/angie.pid)
```

Убедившись, что эта конфигурация работает, вы можете удалить сертификаты `certbot`, а также отключить или целиком удалить его с сервера, если он больше нигде не используется, например:

```
$ sudo rm -rf /etc/letsencrypt

$ sudo systemctl stop certbot.timer
$ sudo systemctl disable certbot.timer
$ # -- или --
$ sudo rm /etc/cron.d/certbot

$ sudo apt remove certbot
$ # -- или --
$ sudo dnf remove certbot
```

4.10.8.9 Настройка пользовательских метрик

Angie может собирать пользовательские числовые метрики в разделяемой памяти и выводить их через API статистики по адресу `/status/http/metric_zones/`. Это обеспечивает модуль `Metric`.

Примечание

Это руководство описывает метрики для HTTP-трафика. Для TCP- и UDP-трафика см. потоковый модуль `Metric`, который выводит свои зоны по адресу `/status/stream/metric_zones/`.

Шаги настройки

1. Создайте зону метрик в блоке `http`:
 - `metric_zone` создает зону с одной метрикой.
 - `metric_complex_zone` создает зону с несколькими именованными метриками.
2. Обновляйте метрики при обработке запросов директивой `metric`. Используйте пару `key=value` (оба — комплексные значения), и выберите этап обновления параметром `on=` (`request`, `response` или `end`).
3. Откройте API через `location`:

```
location /status/ {
    api /status/http/metric_zones/;
}
```

Пример

Подсчет запросов по хостам и вывод метрик через API:

```
http {
    metric_zone requests:128k count;

    server {
        listen 80;

        location / {
            metric requests $host=1;
        }

        location /status/ {
            api /status/http/metric_zones/;
        }
    }
}
```

Примечания

- При `expire=on` и переполнении памяти истекают самые давно неиспользуемые записи. При `expire=off` новые обновления отбрасываются, а счетчик `discarded` увеличивается.
- Если задан `discard_key`, метрики истекших записей агрегируются под этим ключом в API.
- Длина ключей и значений ограничена 255 байт; длинные ключи усекутся в API.
- Пустое значение трактуется как 0, а непустая строка без числа в начале — как 1.

4.10.8.10 Режим отладки

Режим отладки следует включать перед самостоятельной диагностикой или по рекомендации технической поддержки. Чтобы включить режим отладки, выполните команду `angie debug` в *CLI*. Для отключения режима отладки выполните команду `no angie debug`.

Примечание

Использование режима отладки может снизить производительность, а также увеличить расход места на диске.

Отладочный лог

Чтобы включить отладочный лог, задайте в конфигурации уровень `debug` с помощью директивы `error_log`:

```
error_log /path/to/log debug;
```

Если выключить режим отладки, но оставить уровень `debug` в директиве `error_log`, Angie ADC будет добавлять записи в лог на уровне `info`.

Если переопределить `error_log` в конфигурации, но не указать в ней уровень `debug`, отладочный лог будет отключен. Здесь переопределение лога на уровне `server` отключает отладочный лог для отдельного сервера:

```
error_log /path/to/log debug;
```

```
http {
```

```
server {
    error_log /path/to/log;
    # ...
}
```

Во избежание этого уберите строку, переопределяющую `error_log`, либо задайте в ней уровень `debug`:

```
error_log /path/to/log debug;

http {
    server {
        error_log /path/to/log debug;
        # ...
    }
}
```

Расположение директивы

Расположение директивы `error_log` влияет на полноту собираемой отладочной информации.

Директива, указанная на более низком уровне конфигурации (например, внутри блока `server` или `location`), заменяет настройки логирования, заданные на более высоком уровне (например, на основном уровне конфигурации или внутри блока `http`).

Отключение отладочного лога для конкретного сервера

Если глобально отладочный лог включен, но для отдельного сервера `error_log` указан без уровня `debug`, то для этого сервера отладочная информация собираться не будет.

```
error_log /var/log/angie-lb/error.log debug; # Глобальный отладочный лог

http {

    server {

        listen 80;
        server_name example.com;

        error_log /var/log/angie-lb/example.com.error.log;
        # Отладочный лог для example.com отключен, в файле - уровень info

        # ...
    }

    server {

        listen 80;
        server_name another.com;

        # Для этого сервера будет использоваться глобальный отладочный лог
        # ...
    }
}
```

Сохранение отладочного лога для сервера

Чтобы сохранить сбор отладочной информации для конкретного сервера, но направить ее в другой файл, необходимо также указать уровень `debug`:

```
error_log /var/log/angie-lb/error.log debug; # Глобальный отладочный лог

http {

    server {

        listen 80;
        server_name example.com;

        error_log /var/log/angie-lb/example.com.error.log debug;
        # Отладочный лог для example.com включен, но пишется в отдельный файл

        # ...
    }
}
```

Таким образом, чтобы включить отладочный лог глобально, но переопределить файл лога для отдельных блоков, также укажите уровень **debug** в этих переопределениях. Иначе, если в директиве *error_log* не указан уровень логирования, по умолчанию будет использоваться уровень **error** и отладочная информация для этих блоков будет потеряна.

Лог для отдельных адресов

Можно включить отладочный лог только для *указанных клиентских адресов*:

```
error_log /path/to/log;

events {
    debug_connection 192.168.1.1;
    debug_connection 192.168.10.0/24;
}
```

ГЛАВА 5

Глобальная балансировка

Глобальная балансировка, также GSLB (Global Server Load Balancing) – это сервис, который позволяет распределять запросы клиентов между несколькими дата-центрами на основе их доступности и производительности.

Для работы глобальной балансировки собирается кластер, настраивается конфигурация GSLB и делегирование зон на GSLB-серверы Angie ADC. Локальный DNS-сервер будет отправлять клиентские запросы для этих зон на один из GSLB-серверов. Сервер GSLB решает, какой дата-центр лучше обслужит запрос, выбирает сервер и передает его IP-адрес DNS-серверу. DNS-сервер отправляет этот IP-адрес клиенту, после чего клиент направляет свой запрос на новый сервер по полученному адресу.

Поддерживаются динамические записи типа A и AAAA и статические записи типа A, AAAA, CNAME, MX, TXT.

Для динамических записей поддерживаются следующие типы балансировки:

- **round_robin** — запросы распределяются по серверам последовательно.
- **weighted round_robin** — запросы распределяются по серверам последовательно с учетом весов серверов (на серверы с большим весом запросы будут попадать чаще).
- **failover** — запрос направляется на доступный сервер с наивысшим приоритетом.

В фоновом режиме проводятся проверки работоспособности бэкенд-серверов, заданные в конфигурации.

Подробнее см: [Настройка конфигурации](#).

Возможности GSLB в Angie ADC:

- Гибкая настройка балансировки: поддержка статических и динамических записей, поддержка нескольких записей для одного домена.
- Масштабируемость: простота добавления новых доменов и записей через веб-интерфейс.
- Отказоустойчивость: непрерывная доступность приложений даже при сбоях в одном из дата-центров, возможность настройки TTL.
- Кластеризация: упрощение администрирования нескольких GSLB-модулей в разных дата-центрах.

В этом разделе:

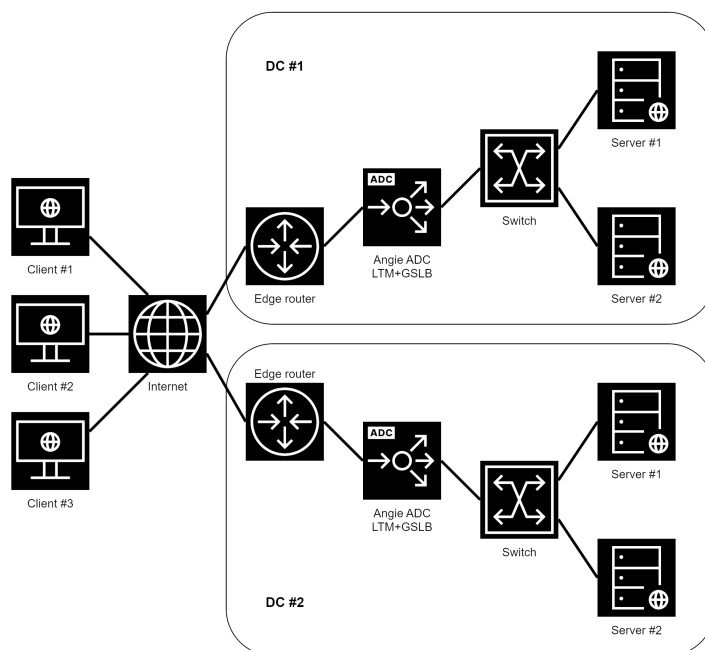


Рис. 1: Пример развертывания кластера из двух Angie ADC с GSLB в двух дата-центрах

- *Настройка сервиса GSLB*
- *Настройка конфигурации*

5.1 Настройка сервиса GSLB

Примечание

Перед настройкой GSLB необходимо создать кластер Angie ADC. Подробнее см. в статье *Создание кластера*.

Настройка GSLB состоит из следующих шагов:

1. Настройка конфигурации GSLB в веб-интерфейсе (на любом из узлов кластера):
 - Включение функции глобальной балансировки.
 - Добавление доменов, для которых требуется выполнять глобальную балансировку.
 - Настройка проверок работоспособности для контроля доступности бэкендов.
 - Добавление DNS-записей для каждого домена.
 - Применение настроек конфигурации.

Синхронизация конфигурации на другие узлы кластера происходит автоматически.

Подробнее см. в статье *Настройка конфигурации*.

2. Настройка делегирования заданных доменов на GSLB-узлы Angie ADC на первичном DNS-сервере.

Пример для `www.example.com`:

```
adc-dc1 IN A 172.22.0.10 # IP-адрес первого GSLB-сервера Angie ADC
adc-dc2 IN A 172.22.1.10 # IP-адрес второго GSLB-сервера Angie ADC
www IN NS adc-dc1.example.com # делегирование зоны www.example.com на первый
```

→ GSLB-сервер
www IN NS adc-dc2.example.com # делегирование зоны www.example.com на второй
→ GSLB-сервер

5.2 Настройка конфигурации

Конфигурация GSLB настраивается в веб-интерфейсе Angie ADC (Управление трафиком → Глобальная балансировка).

adc-rs1-2 > Глобальная балансировка

Настройки глобальной балансировки

☒ Глобальная балансировка

Домены: 1 Записи: 0 Проверки работоспособности: 0 Порт: 53

Домен	Динамические записи	Статические записи
example.com	0	0

Добавить домен

Выбранный домен: example.com Удалить домен

Имя	Тип	Балансировка	TTL	Бэкенды	Проверки
В этом домене пока нет динамических записей					

Добавить запись

Имя	Тип	TTL	Бэкенды
В этом домене пока нет статических записей			

Добавить запись

Имя	Режим	Интервал	Таймаут	Используется в
Проверок пока нет				

Добавить проверку

Сбросить Применить

5.2.1 Добавление домена и создание записей

Чтобы добавить домен и настроить для него глобальную балансировку, выполните следующие действия:

- Откройте веб-интерфейс Angie ADC и перейдите в раздел **Управление трафиком** → **Глобальная балансировка**.
Откроется окно настройки глобальной балансировки.
- В блоке **Таблица доменов** нажмите **Добавить домен** и укажите имя домена, например `example.com`. Имя домена должно быть корректным DNS-именем без завершающей точки.
- В блоке **Проверки работоспособности** настройте проверки доступности бэкендов. Эти проверки необходимо назначить соответствующим бэкенд-серверам при создании динамических записей. Нажмите **Добавить проверку**, выберите ее тип и задайте остальные параметры. Поддерживаются следующие типы проверок:
 - HTTP** — активные проверки работоспособности по протоколу HTTP.
 - ICMP** — проверка доступности с помощью ICMP Echo Request (ping).
 - TCP** — активные проверки работоспособности по протоколу TCP.
 - Angie** — проверка на основе данных о состоянии бэкендов, полученных через *Status API Angie*. Подходит для случаев, когда бэкенды не имеют собственного URL для проверки работоспособности.

Параметры для проверки HTTP

Имя	Имя проверки работоспособности (до 255 символов).
Режим	Режим проверки HTTP — активные проверки работоспособности по протоколу HTTP.
Интервал	Интервал между проверками в секундах.
Таймаут	Таймаут ожидания в секундах. Если за это время ответ не получен, проверка считается неуспешной.
Хост	Значение заголовка <code>Host</code> : в HTTP-запросе (DNS-имя проверяемого сервиса), например <code>app.example.com</code> .
Порт	Порт для запроса.
Путь запроса	Путь, по которому выполняется запрос, например <code>/health</code> или <code>/status</code> .
Метод	HTTP-метод запроса проверки, например <code>GET</code> .
Код ответа	Код ответа сервера, используемый в качестве критерия успешного прохождения проверки, например <code>200</code> .
Заголовки	Заголовок ответа сервера, используемый в качестве критерия успешного прохождения проверки.
Тело ответа	Тело ответа сервера, используемое в качестве критерия успешного прохождения проверки.

Параметры для проверки ICMP

Имя	Имя проверки работоспособности (до 255 символов).
Режим	Режим проверки ICMP — проверка доступности с помощью ICMP Echo Request (ping).
Интервал	Интервал между проверками в секундах.
Таймаут	Таймаут ожидания в секундах. Если за это время ответ не получен, проверка считается неуспешной.
Хост	DNS-имя проверяемого сервиса, например <code>app.example.com</code> .
Число пакетов	Количество ICMP-пакетов, отправляемых при выполнении проверки.

Параметры для проверки TCP

Имя	Имя проверки работоспособности (до 255 символов).
Режим	Режим проверки TCP — активные проверки работоспособности по протоколу TCP.
Интервал	Интервал между проверками в секундах.
Таймаут	Таймаут ожидания в секундах. Если за это время ответ не получен, проверка считается неуспешной.
Хост	IP-адрес проверяемого сервиса.
Порт	Порт, используемый для выполнения проверки.

Параметры для проверки Angie

Имя	Имя проверки работоспособности (до 255 символов)
Режим	Режим проверки Angie — проверка на основе данных о состоянии бэкендов, полученных через <i>Status API Angie</i> . Подходит для случаев, когда бэкенды не имеют собственного URL для проверки работоспособности.
Интервал	Интервал между проверками в секундах.
Таймаут	Таймаут ожидания в секундах. Если за это время ответ не получен, проверка считается неуспешной.
Хост	Значение заголовка <code>Host</code> : в HTTP-запросе (DNS-имя проверяемого сервиса), например <code>app.example.com</code> .
Порт	Порт, используемый для выполнения проверки.
URI	Путь, по которому выполняется запрос.
Заголовки	Заголовок ответа сервера, используемый в качестве критерия успешного прохождения проверки.

- Настройте записи для добавленного домена. Выберите тип записи (статическая или динамическая) и укажите необходимые параметры.

Совет

- Чтобы настроить балансировку для домена, а не только для записей внутри него, создайте динамическую запись типа **A** с именем **@** и укажите необходимые параметры балансировки.
- Чтобы задать общее правило балансировки для разных поддоменов, создайте динамическую запись типа **A** с именем ***** и укажите необходимые параметры балансировки.

Параметры динамической записи типа A и AAAA

Имя	Имя записи. Должно быть уникальным в рамках одного домена.
Тип	Тип записи. Возможные значения: • A — сопоставляет доменное имя с IPv4-адресом. • AAAA — сопоставляет доменное имя с IPv6-адресом.
Балансировка	Тип балансировки. Возможные значения: • round_robin — запросы распределяются по серверам последовательно. • weighted round_robin — запросы распределяются по серверам последовательно с учетом весов серверов (на серверы с большим весом запросы будут попадать чаще). • failover — запрос направляется на доступный сервер с наивысшим приоритетом.
TTL	Значение TTL (Time to Live) в секундах. Определяет, как долго DNS-запись может храниться в кэше. По умолчанию установлено значение 30.
Бэкенды	Таблица бэкенд-серверов. Необходимо задать как минимум один бэкенд. Для каждого бэкенд-сервера задаются параметры: • IP: IP-адрес сервера. Поддерживаются IPv4 и IPv6-адреса. • Проверка: проверка работоспособности, выполняемая для этого сервера. Выбирается по имени из списка заданных проверок. • Приоритет: приоритет сервера (необходим для режима failover). Чем меньше значение, тем выше приоритет сервера, которому это значение назначено. • Вес: вес сервера (необходим для режима weighted round_robin). На серверы с большим весом запросы будут попадать чаще.

Параметры статической записи типа A и AAAA

Имя	Имя записи.
Тип	Тип записи. Возможные значения: • A — сопоставляет доменное имя с IPv4-адресом. • AAAA — сопоставляет доменное имя с IPv6-адресом.
TTL	Значение TTL (Time to Live) в секундах. Определяет, как долго DNS-запись может храниться в кэше. По умолчанию установлено значение 300.
Бэкенды	Таблица бэкенд-серверов. Для каждого бэкенд-сервера задаются его параметры: • IP: IP-адрес сервера.

Параметры статической записи типа CNAME

Имя	Имя записи.
Тип	Тип записи CNAME — задает псевдоним для другого доменного имени. Необходимо указать ровно один хост.
TTL	Значение TTL (Time to Live) в секундах. Определяет, как долго DNS-запись может храниться в кэше. По умолчанию установлено значение 300.
Бэкенды	Таблица бэкенд-серверов. Необходимо указать ровно один бэкенд: • Хост: доменное имя, на которое будет указывать CNAME-запись.

Параметры статической записи типа MX

Имя	Имя записи.
Тип	Тип записи MX — указывает почтовый сервер для домена. Необходимо указать хотя бы один почтовый сервер.
TTL	Значение TTL (Time to Live) в секундах. Определяет, как долго DNS-запись может храниться в кэше. По умолчанию установлено значение 300.
Бэкенды	Таблица бэкенд-серверов. Для каждого бэкенд-сервера задаются его параметры: • Хост: доменное имя почтового сервера. • Приоритет: приоритет сервера. Чем меньше значение, тем выше приоритет. Необязательный параметр.

Параметры статической записи типа TXT

Имя	Имя записи.
Тип	Тип записи TXT — содержит произвольный текст.
TTL	Значение TTL (Time to Live) в секундах. Определяет, как долго DNS-запись может храниться в кэше. По умолчанию установлено значение 300.
Бэкенды	Таблица записей бэкенд-серверов. Для каждой записи бэкенд-сервера задаются его параметры: • Текст: TXT-запись должна содержать непустой текст длиной не более 4096 символов.

5. Нажмите Применить, чтобы применить конфигурацию.

Статус применения конфигурации будет отображаться в верхней части экрана.

ГЛАВА 6

IP-маршрутизация

Angie ADC поддерживает ручную настройку режимов высокой доступности на устройстве Angie ADC по протоколам BGP и OSPF (динамическая маршрутизация) и с VRRP (статическая маршрутизация) в вариантах Active-Standby и Active-Active. Также поддерживается использование протокола BFD для уменьшения времени реакции на события и механизма RHI для оценки состояния апстримов.

В этом разделе:

- *BGP-маршрутизация*
- *OSPF-маршрутизация*
- *VRRP-маршрутизация*
- *Использование протокола BFD для уменьшения времени реакции*
- *Настройка RHI*

Примечание

Рекомендуется резервировать все компоненты инфраструктуры. Отказ сопутствующей инфраструктуры (например выход из строя вышестоящего маршрутизатора или сбой серверов) выходит за рамки решения высокой доступности Angie ADC.

Пример отказоустойчивой инфраструктуры:

6.1 BGP-маршрутизация

В этом разделе описана настройка BGP-маршрутизации для обеспечения высокой доступности в режиме резервирования (Active-Standby) и в режиме распределения нагрузки (Active-Active). Для настройки используется *интерфейс командной строки (CLI) Angie ADC*.

В этом разделе:

- *Режим резервирования (Active-Standby)*
- *Режим распределения нагрузки (Active-Active)*

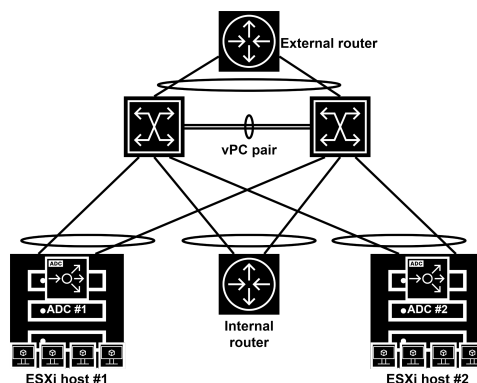


Рис. 1: Рис. 1

6.1.1 Резервирование с помощью протокола BGP (Active-Standby)

6.1.1.1 Введение

В этом разделе описана настройка высокой доступности с использованием протокола BGP в режиме резервирования (Active-Standby). Для настройки используется *интерфейс командной строки (CLI) Angie ADC*.

В качестве протокола динамической маршрутизации используется BGPv4 для адресного семейства IPv4 unicast. Отказоустойчивость в рамках IPv6 unicast настраивается аналогично.

В рассматриваемом примере Angie ADC #1 выполняет функции активной системы балансировки трафика, а Angie ADC #2 остается в резерве.

Пример типового участка сети с системами балансировки Angie ADC (ADC #1 и ADC #2) представлен на схеме:



Рис. 1

Добавим IP-адресацию:

- В сегменте сети, куда подключаются External router, ADC #1 и ADC #2, будет использоваться подсеть 192.168.0.0/24.
- В сегменте сети, куда подключаются Internal router, ADC #1 и ADC #2, будет использоваться подсеть 192.168.1.0/24.
- Четвертый октет адреса для каждого устройства (в рамках соответствующей подсети) показан на схеме ниже.

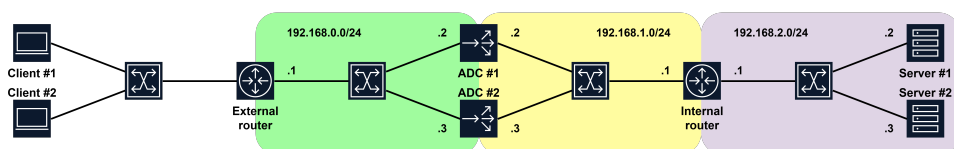


Рис. 2

Адресация клиентов не приводится, так как в данном случае мы будем считать, что их адреса формально не определены, то есть они могут подключаться из любой сети, включая сеть интернет.

Для этих целей External router будет анонсировать маршрут по умолчанию по BGP для ADC #1 и ADC #2. В данном документе будем использовать только одну автономную систему BGP №65001.

Примечание

Автономная система BGP — логическое объединение устройств в сети для маршрутизации через протокол BGP.

6.1.1.2 BGP-сессии

Устанавливаемые BGP-сессии представлены на схеме:

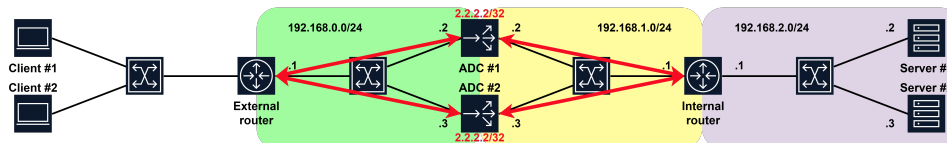


Рис. 3

Красным цветом на схеме отмечены IP-адреса виртуальных серверов. Это те адреса, к которым подключаются клиенты снаружи. Системы балансировки должны анонсировать маршруты до этих виртуальных серверов в сторону External router, причем маршрут через ADC #1 должен быть более предпочтительным. Добиться этого можно разными способами, манипулируя атрибутами пути в BGP.

Так как все BGP-взаимодействие производится строго в рамках одной автономной системы, мы будем использовать атрибут Local preference (чем больше, тем лучше) для выбора наиболее оптимального пути. Трафик от серверов в сторону клиента должен проходить через ту же систему балансировки, что и трафик от клиентов к серверам. Добиться этого можно также путем манипулирования значением атрибута Local preference для префикса 0.0.0.0/0.

6.1.1.3 Обработка типовых сбоев

Ниже описаны сценарии работы Angie ADC при сбоях.

Полный отказ ADC #1

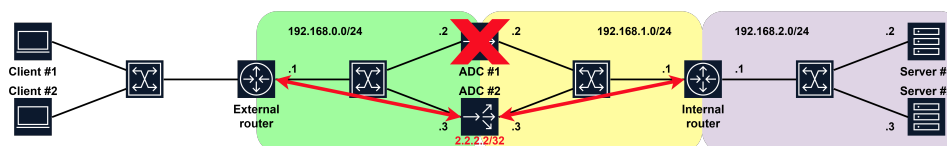


Рис. 4

При полном отказе или отключении от сети ADC #1 разрываются обе BGP-сессии с ADC #1. Через ADC #1 не проходят никакие объявления маршрутов. Остается связь по BGP только с ADC #2. Переключение на резервную систему ADC #2 производится автоматически. Никакие дополнительные настройки не требуются.

Отказ внешнего интерфейса ADC #1

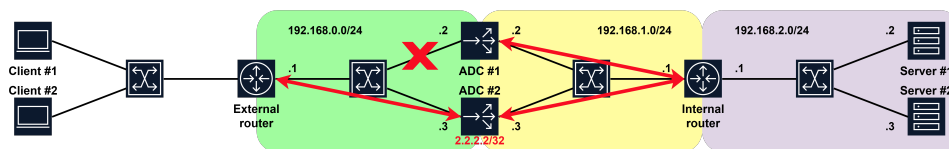


Рис. 5

В этом случае разрывается BGP-сессия между ADC #1 и External router. ADC #1 теперь не может объявить маршрут до адреса виртуального сервера в сторону External router. Несмотря на наличие BGP-сессии между ADC #1 и Internal router, ADC #1 не отправляет маршрут по умолчанию в сторону Internal router, так как сама не получает его от External router из-за сбоя линка. Таким образом, Internal router видит маршрут по умолчанию только через ADC #2, а External router получает маршрут до адреса виртуального сервера только через полностью работающий ADC #2.

Отказ линка между ADC #1 и Internal router

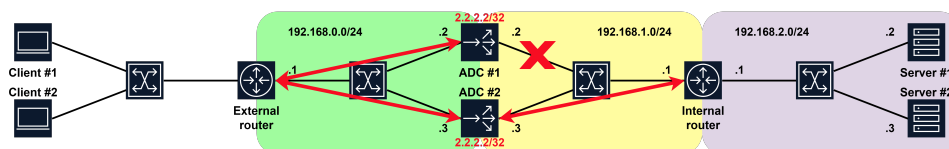


Рис. 6

Если BGP-сессия между External router и ADC #1 сохраняется, то ADC #1 будет продолжать анонсировать маршрут до адреса виртуального сервера, хотя реальные серверы для ADC #1 уже недоступны. Таким образом, мы получаем классический black hole для трафика. Чтобы избежать этого, анонсирование маршрутов до виртуальных серверов должно быть условным, то есть ADC #1 анонсирует маршрут до адреса виртуального сервера, только если в BGP-таблице присутствуют префиксы, соответствующие подсетям реальных серверов.

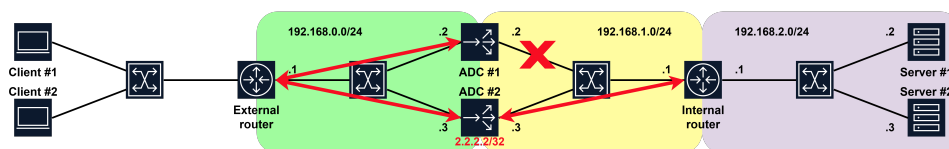


Рис. 7

6.1.1.4 Настройка маршрутизации

Приведем ключевые настройки маршрутизации для External router, Internal router, ADC #1 и ADC #2.

Ключевые настройки маршрутизации для External router

```
hostname external
interface GigabitEthernet0/0
 ip address 192.168.0.1 255.255.255.0
!Интерфейс в сторону систем балансировки
router bgp 65001
 neighbor 192.168.0.2 remote-as 65001
!Сессия с ADC#1
```

```
neighbor 192.168.0.2 default-originate
!Анонсирование маршрута по умолчанию
neighbor 192.168.0.3 remote-as 65001
!Сессия с ADC#2
neighbor 192.168.0.3 default-originate
!Анонсирование маршрута по умолчанию
end
```

Ключевые настройки маршрутизации для Internal router

```
hostname internal
interface GigabitEthernet0/0
 ip address 192.168.1.1 255.255.255.0
!Интерфейс в сторону систем балансировки
interface GigabitEthernet1/0
 ip address 192.168.2.1 255.255.255.0
!Интерфейс в сторону подсети с реальными серверами
router bgp 65001
 network 192.168.2.0 mask 255.255.255.0
!Добавление информации о подсети с реальными серверами в BGP
 neighbor 192.168.1.2 remote-as 65001
!Сессия с ADC#1
 neighbor 192.168.1.3 remote-as 65001
!Сессия с ADC#2
end
```

Ключевые настройки маршрутизации Angie ADC #1

```
hostname adc1
!Имя устройства
ip prefix-list vip seq 5 permit 2.2.2.2/32
!Префикс-лист, описывающий адреса виртуальных серверов
ip prefix-list servers seq 5 permit 192.168.2.0/24
!Префикс-лист, описывающий адреса реальных серверов.
interface ens37
 ip address 192.168.0.2/24
!Интерфейс в сторону External router
exit
interface ens38
 ip address 192.168.1.2/24
!Интерфейс в сторону Internal router
exit
interface lo
 ip address 2.2.2.2/32
!Loopback-интерфейс для адресов виртуальных серверов.
exit
router bgp 65001
 neighbor 192.168.0.1 remote-as 65001
!Сессия с External router
 neighbor 192.168.1.1 remote-as 65001
!Сессия с Internal router
address-family ipv4 unicast
 network 2.2.2.2/32
!Добавление префикса для адреса виртуального сервера в BGP
 neighbor 192.168.0.1 route-reflector-client
!Разрешаем передачу префиксов между двумя iBGP-клиентами
```

```

neighbor 192.168.0.1 route-map fromclients in
!Манипуляции над маршрутами, полученными от External router, меняем local preference
neighbor 192.168.0.1 route-map 2clients out
!Манипуляции над маршрутами, отправляемыми в сторону External router, меняем local
↪preference
neighbor 192.168.0.1 advertise-map 2clients exist-map servers
!Условная отправка префиксов: если префикс есть в route-map servers, то будет отправка
neighbor 192.168.1.1 route-reflector-client
!Разрешаем передачу префиксов между двумя iBGP-клиентами
neighbor 192.168.1.1 next-hop-self force
!Подменяем адрес атрибута next-hop на собственный даже для отраженных префиксов
exit-address-family
exit
route-map 2clients permit 10
match ip address prefix-list vip
set local-preference 150
!Для префиксов, соответствующих адресам виртуальных серверов, увеличиваем Local
↪preference
exit
route-map 2clients deny 20
!Блокируем отправку всех остальных префиксов
exit
route-map fromclients permit 10
set local-preference 150
!Увеличиваем Local preference для маршрутов, получаемых от External router
exit
route-map servers permit 10
match ip address prefix-list servers
!Условие для условной отправки: если в BGP-таблице есть маршрут,
!соответствующий префикс-листу servers, то условие считается выполненным
exit
end

```

Ключевые настройки маршрутизации Angie ADC #2

```

hostname adc2
!Имя устройства
ip prefix-list vip seq 5 permit 2.2.2.2/32
!Префикс-лист, описывающий адреса виртуальных серверов
ip prefix-list servers seq 5 permit 192.168.2.0/24
!Префикс-лист, описывающий адреса реальных серверов.
interface ens37
ip address 192.168.0.3/24
!Интерфейс в сторону External router
exit
interface ens38
ip address 192.168.1.3/24
!Интерфейс в сторону Internal router
exit
interface lo
ip address 2.2.2.2/32
!Loopback-интерфейс для адресов виртуальных серверов.
exit
router bgp 65001
neighbor 192.168.0.1 remote-as 65001
!Сессия с External router
neighbor 192.168.1.1 remote-as 65001

```

```
!Сессия с Internal router
address-family ipv4 unicast
    network 2.2.2.2/32
!Добавление префикса для адреса виртуального сервера в BGP
    neighbor 192.168.0.1 route-reflector-client
!Разрешаем передачу префиксов между двумя iBGP-клиентами
    neighbor 192.168.0.1 route-map fromclients in
!Манипуляции над маршрутами, полученными от External router, меняем local preference
    neighbor 192.168.0.1 route-map 2clients out
!Манипуляции над маршрутами, отправляемыми в сторону External router, меняем local
->preference
    neighbor 192.168.0.1 advertise-map 2clients exist-map servers
!Условная отправка префиксов: если префикс есть в route-map servers, то будет отправка
    neighbor 192.168.1.1 route-reflector-client
!Разрешаем передачу префиксов между двумя iBGP-клиентами
    neighbor 192.168.1.1 next-hop-self force
!Подменяем адрес атрибута next-hop на собственный даже для отраженных префиксов
exit-address-family
exit
route-map 2clients permit 10
    match ip address prefix-list vip
    set local-preference 50
!Для префиксов, соответствующих адресам виртуальных серверов, уменьшаем Local
->preference
exit
route-map 2clients deny 20
!Блокируем отставку всех остальных префиксов
exit
route-map fromclients permit 10
    set local-preference 50
!Уменьшаем Local preference для маршрутов, получаемых от External router
exit
route-map servers permit 10
    match ip address prefix-list servers
!Условие для условной отправки: если в BGP-таблице есть маршрут,
!соответствующий префикс-листу servers, то условие считается выполненным
exit
end
```

6.1.1.5 Проверка работы

Выполним проверку работы системы маршрутизации, когда ADC #1 и ADC #2 находятся в полностью работоспособном состоянии.

Примечание

В листингах ниже отсутствуют префиксы, не относящиеся к обсуждаемым вопросам отказоустойчивости.

Маршрутизатор External router видит маршрут к адресу виртуального сервера через обе системы балансировки, однако маршрут через ADC #1 является более предпочтительным:

```
external#sho ip ro
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
```

```

E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
+ - replicated route, % - next hop override

Gateway of last resort is not set

2.0.0.0/32 is subnetted, 1 subnets
B    2.2.2.2 [200/0] via 192.168.0.2, 02:06:11
    192.168.0.0/24 is variably subnetted, 2 subnets, 2 masks
C    192.168.0.0/24 is directly connected, GigabitEthernet0/0
L    192.168.0.1/32 is directly connected, GigabitEthernet0/0
external#sho ip bgp
BGP table version is 9, local router ID is 11.11.11.11
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
               x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

      Network          Next Hop          Metric LocPrf Weight Path
      0.0.0.0          0.0.0.0              0         150      0 i
*>i 2.2.2.2/32        192.168.0.2          0         150      0 i
* i                   192.168.0.3          0         50       0 i

```

Маршрутизатор Internal router видит маршрут по умолчанию через обе системы балансировки, однако маршрут через ADC #1 является более предпочтительным:

```

internal#sho ip ro
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       + - replicated route, % - next hop override

Gateway of last resort is 192.168.1.2 to network 0.0.0.0

B*    0.0.0.0/0 [200/0] via 192.168.1.2, 02:12:37
    192.168.1.0/24 is variably subnetted, 2 subnets, 2 masks
C    192.168.1.0/24 is directly connected, GigabitEthernet0/0
L    192.168.1.1/32 is directly connected, GigabitEthernet0/0
    192.168.2.0/24 is variably subnetted, 2 subnets, 2 masks
C    192.168.2.0/24 is directly connected, GigabitEthernet1/0
L    192.168.2.1/32 is directly connected, GigabitEthernet1/0
internal#sho ip bgp
BGP table version is 12, local router ID is 33.33.33.33
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
               x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

      Network          Next Hop          Metric LocPrf Weight Path
*>i 0.0.0.0          192.168.1.2          0         150      0 i

```

```
* i          192.168.1.3          0    50    0 i
*> 192.168.2.0/24 0.0.0.0        0          32768 i
```

Отключим теперь на ADC #1 интерфейс в сторону Internal router и проверим маршрутизацию:

```
adc1# sho int bri
Interface      Status  VRF      Addresses
-----
ens37          up      default  192.168.0.2/24
ens38          down    default  192.168.1.2/24
lo             up      default  2.2.2.2/32

adc1# conf t
adc1(config)# int ens38
adc1(config-if)# shut
adc1# sho int bri
Interface      Status  VRF      Addresses
-----
ens37          up      default  192.168.0.2/24
ens38          down    default  192.168.1.2/24
lo             up      default  2.2.2.2/32
```

Изучим теперь таблицу маршрутизации и BGP-таблицу на маршрутизаторе External router. Несмотря на наличие BGP-сессии до каждой из систем балансировки, только по одной сессии прилетает маршрут на адрес виртуального сервера – от ADC #2:

```
external#sho ip bg su
BGP router identifier 11.11.11.11, local AS number 65001
BGP table version is 8, main routing table version 8
2 network entries using 296 bytes of memory
2 path entries using 128 bytes of memory
2/1 BGP path/bestpath attribute entries using 272 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 696 total bytes of memory
BGP activity 3/1 prefixes, 4/2 paths, scan interval 60 secs
Neighbor      V      AS MsgRcvd MsgSent  TblVer  InQ OutQ Up/Down  State/
↪PfxRcd
192.168.0.2    4        1    156    150      8    0    0 00:07:29    0
192.168.0.3    4        1    157    150      8    0    0 00:07:29    1

external#sho ip bg
BGP table version is 8, local router ID is 11.11.11.11
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
               x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

      Network      Next Hop      Metric LocPrf Weight Path
      0.0.0.0      0.0.0.0              0 i
*>i 2.2.2.2/32      192.168.0.3      0    50    0 i

external#sho ip ro
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
```



```

+ - replicated route, % - next hop override
Gateway of last resort is not set
  2.0.0.0/32 is subnetted, 1 subnets
B       2.2.2.2 [200/0] via 192.168.0.3, 00:04:36
        192.168.0.0/24 is variably subnetted, 2 subnets, 2 masks
C       192.168.0.0/24 is directly connected, GigabitEthernet0/0
L       192.168.0.1/32 is directly connected, GigabitEthernet0/0

```

И теперь маршрутная информация со стороны Internal router:

```

internal#sho ip bgp
BGP table version is 8, local router ID is 33.33.33.33
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
               x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

   Network          Next Hop           Metric LocPrf Weight Path
*>i 0.0.0.0          192.168.1.3             0      50      0 i
*> 192.168.2.0/24    0.0.0.0                 0          32768 i
internal#sho ip ro
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       + - replicated route, % - next hop override
Gateway of last resort is 192.168.1.3 to network 0.0.0.0
B*    0.0.0.0/0 [200/0] via 192.168.1.3, 00:00:49
C      192.168.1.0/24 is directly connected, GigabitEthernet0/0
L      192.168.1.1/32 is directly connected, GigabitEthernet0/0
C      192.168.2.0/24 is directly connected, GigabitEthernet1/0
L      192.168.2.1/32 is directly connected, GigabitEthernet1/0

```

6.1.1.6 Применение с IP transparency

Все описанное выше относится к режиму работы Angie ADC с включенной опцией IP transparency. При включенной опции IP Transparency Angie ADC сохраняет исходный адрес отправителя в IP-пакетах, позволяя серверам на сетевом уровне различать клиентские подключения.

Очень часто администраторы систем балансировки отключают эту опцию, в этом случае Angie ADC будет выполнять не только dNAT, но sNAT над трафиком, приходящим от клиентов. Если система балансировки Angie ADC выполняет sNAT, то в этом случае маршрутизация несколько упрощается, так как в этом случае нет необходимости анонсировать на Internal router маршрут по умолчанию. Если ADC #1 и ADC #2 выполняют sNAT в разные адреса, то в этом случае нет необходимости заботиться о том, с какими значениями Local preference (и иными атрибутами пути) маршруты до соответствующих адресов попадают на Internal router.

6.1.2 Распределение нагрузки с помощью протокола BGP (Active-Active)

6.1.2.1 Введение

В этом разделе описана настройка высокой доступности с использованием протокола BGP в режиме распределения нагрузки (Active-Active). Для настройки используется *командная строка Angie ADC CLI*.

В качестве протокола динамической маршрутизации используется BGPv4 для адресного семейства IPv4 unicast. Отказоустойчивость в рамках IPv6 unicast настраивается аналогично.

В рассматриваемом примере устройства Angie ADC #1 и Angie ADC #2 выполняют функции активной системы балансировки трафика одновременно. Для динамического исключения одного из Angie ADC из работы в случае сбоя (технические проблемы, потери связности и так далее) можно использовать *механизм RHI (Route Health Injection)*.

Пример типового участка сети с системами балансировки Angie ADC (ADC #1 и ADC #2) представлен на схеме:



Рис. 1

Добавим IP-адресацию:

- В сегменте сети, куда подключаются External router, ADC #1 и ADC #2, будет использоваться подсеть 192.168.0.0/24.
- В сегменте сети, куда подключаются Internal router, ADC #1 и ADC #2, будет использоваться подсеть 192.168.1.0/24.
- Четвертый октет адреса для каждого устройства (в рамках соответствующей подсети) показан на схеме ниже.

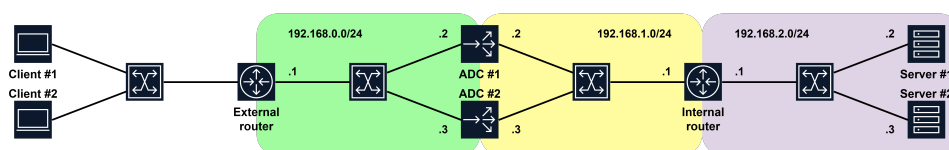


Рис. 2

Адресация клиентов не приводится, так как в данном случае мы будем считать, что их адреса формально не определены, то есть они могут подключаться из любой сети, включая сеть интернет. Для этих целей External router будет анонсировать маршрут по умолчанию по BGP для ADC #1 и ADC #2.

6.1.2.2 BGP-сессии

Устанавливаемые BGP-сессии представлены на схеме:

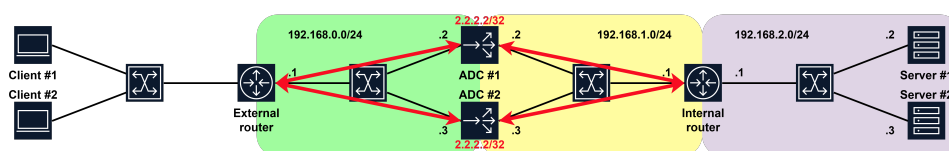


Рис. 3

Красным цветом на схеме отмечены IP-адреса виртуальных серверов. Это те адреса, к которым подключаются клиенты снаружи. ADC #1 и ADC #2 должны анонсировать маршруты до этих виртуальных серверов в сторону External router, причем маршруты через ADC #1 и ADC #2 должны иметь схожие атрибуты пути, чтобы маршрутизатор External router мог выполнить распределение трафика между ними.

Трафик внутри пользовательской сессии от серверов должен возвращаться строго через ту же систему балансировки, через которую он проходил до этого. Добиться этого можно с помощью механизма *SNAT*, когда адрес клиента подменяется на адрес интерфейса системы балансировки.

6.1.2.3 Обработка типовых сбоев

Ниже описаны сценарии работы Angie ADC при сбоях.

Полный отказ ADC #1

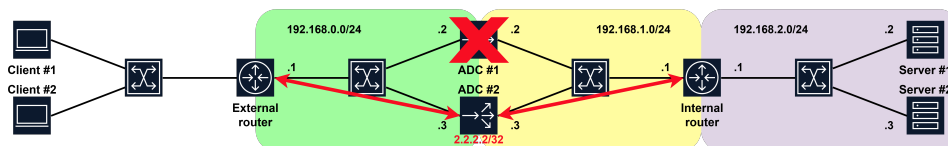


Рис. 4

При полном отказе или отключении от сети ADC #1 разрываются обе BGP-сессии с ADC #1. Через ADC #1 не проходят никакие объявления маршрутов. Остается связь по BGP только с ADC #2. Отключение ADC #1 происходит автоматически. Никакие дополнительные действия не требуются.

Отказ внешнего интерфейса ADC #1

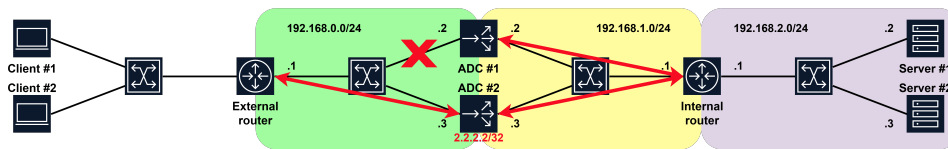


Рис. 5

В этом случае разрывается BGP-сессия между ADC #1 и External router. ADC #1 теперь не может объявить маршрут до адреса виртуального сервера в сторону External router, а пользовательский трафик будет распределяться только на систему балансировки ADC #2. Так как клиентские сессии устанавливаются снаружи, трафик всех вновь устанавливаемых сессий будет проходить только через ADC #2.

Отказ линка между ADC #1 и Internal router

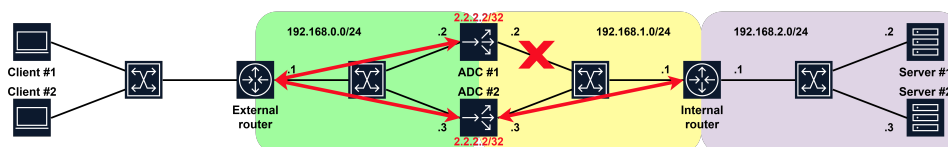


Рис. 6

Если BGP-сессия между External router и ADC #1 сохраняется, то ADC #1 будет продолжать анонсировать маршрут до адреса виртуального сервера, хотя реальные серверы для ADC #1 уже недоступны. Таким образом, мы получаем классический black hole для трафика. Чтобы избежать этого, анонсирование маршрутов до виртуальных серверов должно быть условным. То есть ADC #1 должен отозвать маршрут до адреса виртуального сервера при падении тестов RHI:

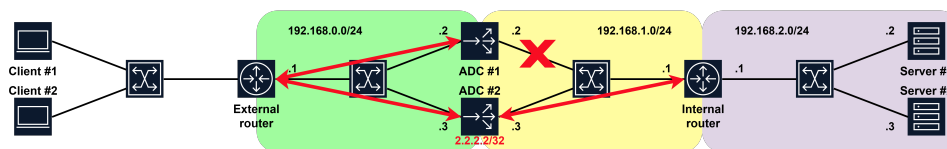


Рис. 7

6.1.2.4 Настройка маршрутизации

При настройке BGP-маршрутизации необходимо указывать собственный номер автономной системы.

Примечание

Автономная система BGP — логическое объединение устройств в сети для маршрутизации через протокол BGP.

Ниже приведены примеры трех распространенных топологий настройки маршрутизации во внешнем сегменте сети. Настройка маршрутизации во внутреннем сегменте выполняется аналогично.

Примечание

В список обязательных команд для некоторых платформ необходимо включить команду задания ограничения на максимальное количество путей, между которыми производится ECMP-балансировка.

Пример такой команды для маршрутизаторов Cisco на базе IOS и IOS-XE:

```
platform loadbalance max-paths 2
```

6.1.2.5 Топология #1

В топологии #1 все три устройства (External router, ADC #1 и ADC #2) принадлежат одной автономной системе:

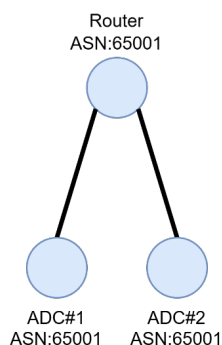


Рис. 8

Обычно на ADC #1 не нужны маршруты, объявляемые со стороны ADC #2. Если для конкретной сети вам нужно, чтобы ADC#1 знал маршруты, которые объявляет ADC #2 (или наоборот), то тогда на маршрутизаторе необходимо создать RR-кластер (Route Reflector cluster). Это позволит обойти ограничение на передачу префиксов, накладываемое правилом BGP split-horizon.

Примечание

Правило BGP split-horizon запрещает отправлять полученные от iBGP-соседа маршруты в сторону другого iBGP-соседа, защищая таким образом от заикливания маршрутов.

Создание RR-кластера поможет также в ситуации, когда перед маршрутизатором External router расположен другой маршрутизатор, находящийся в той же самой автономной системе.

Ключевые настройки маршрутизации для External router

На примере маршрутизатора Cisco Systems на базе IOS или IOS-XE:

```
!Имя устройства
hostname external
interface GigabitEthernet0/0
!Интерфейс в сторону систем балансировки
ip address 192.168.0.1 255.255.255.0
router bgp 65001
!Сессия с ADC#1
neighbor 192.168.0.2 remote-as 65001
!Сессия с ADC#2
neighbor 192.168.0.3 remote-as 65001
address-family ipv4
!Включение соседа
neighbor 192.168.0.2 activate
!Включение соседа
neighbor 192.168.0.3 activate
!Анонсирование маршрута по умолчанию
neighbor 192.168.0.2 default-originate
!Анонсирование маршрута по умолчанию
neighbor 192.168.0.3 default-originate
!Добавление соседа в RR-кластер
neighbor 192.168.0.2 route-reflector-client
!Добавление соседа в RR-кластер
neighbor 192.168.0.3 route-reflector-client
!Указание максимального количества iBGP путей, между которыми будет выполняться↵
↵балансировка
maximum-paths ibgp 2
end
```

Ключевые настройки маршрутизации Angie ADC #1

```
!Имя устройства
hostname adc1
!Интерфейс в сторону External router
interface ens33
ip address 192.168.0.2/24
exit
!Loopback-интерфейс для адресов виртуальных серверов
interface lo
ip address 2.2.2.2/32
ip address 3.3.3.3/32
exit
router bgp 65001
!Сессия с External router
neighbor 192.168.0.1 remote-as 65001
address-family ipv4 unicast
!Добавление префикса для адреса виртуального сервера в BGP
```

```
network 2.2.2.2/32
network 3.3.3.3/32
exit-address-family
exit
end
```

Ключевые настройки маршрутизации Angie ADC #2

```
!Имя устройства
hostname adc2
!Интерфейс в сторону External router
interface ens33
ip address 192.168.0.3/24
exit
!Loopback-интерфейс для адресов виртуальных серверов
interface lo
ip address 2.2.2.2/32
ip address 4.4.4.4/32
exit
router bgp 65001
!Сессия с External router
neighbor 192.168.0.1 remote-as 65001
address-family ipv4 unicast
!Добавление префикса для адреса виртуального сервера в BGP
network 2.2.2.2/32
network 4.4.4.4/32
exit-address-family
exit
end
```

Проверка работы

```
external#sho ip bgp
BGP table version is 4, local router ID is 192.168.0.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found
Network Next Hop Metric LocPrf Weight Path
0.0.0.0 0.0.0.0 0 i
*mi 2.2.2.2/32 192.168.0.2 0 100 0 i
*>i 192.168.0.3 0 100 0 i
external#sho ip ro
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
+ - replicated route, % - next hop override
2.0.0.0/32 is subnetted, 1 subnets
B 2.2.2.2 [200/0] via 192.168.0.2, 00:12:23
[200/0] via 192.168.0.3, 00:12:23
192.168.0.0/24 is variably subnetted, 2 subnets, 2 masks
```

```
C 192.168.0.0/24 is directly connected, GigabitEthernet0/0
L 192.168.0.1/32 is directly connected, GigabitEthernet0/0
```

Выводы просмотревших команд сокращены для краткости. Для тестирования работы RR-кластера мы завели на ADC #1 адрес 3.3.3.3/32, а на ADC #2 — 4.4.4.4/32.

Ниже представлены BGP-таблица и таблица маршрутизации от ADC #2:

```
adc2# sho ip bgp
BGP table version is 8, local router ID is 4.4.4.4, vrf id 0
Default local pref 100, local AS 65001
Status codes: s suppressed, d damped, h history, u unsorted, * valid, > best, =  

↳ multipath,
i internal, r RIB-failure, S Stale, R Removed
Next hop codes: @NNN nexthop's vrf id, < announce-nh-self
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found
Network Next Hop Metric LocPrf Weight Path
*>i 0.0.0.0/0 192.168.0.1 0 100 0 i
*> 2.2.2.2/32 0.0.0.0(adc2) 0 32768 i
* i 192.168.0.2 0 100 0 i
*>i 3.3.3.3/32 192.168.0.2 0 100 0 i
*> 4.4.4.4/32 0.0.0.0(adc2) 0 32768 i
Displayed 4 routes and 5 total paths
adc2# sho ip bgp 3.3.3.3
BGP routing table entry for 3.3.3.3/32, version 7
Paths: (1 available, best #1, table default)
Not advertised to any peer
Local
192.168.0.2 from 192.168.0.2 (3.3.3.3)
Origin IGP, metric 0, localpref 100, valid, internal, bestpath-from-AS Local, best  

↳ (First path received)
Originator: 3.3.3.3, Cluster list: 192.168.0.1
Last update: Mon Apr 21 19:47:21 2025
adc2# sho ip ro
Codes: K - kernel route, C - connected, L - local, S - static,
R - RIP, O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
T - Table, v - VNC, V - VNC-Direct, F - PBR,
f - OpenFabric, t - Table-Direct,
> - selected route, * - FIB route, q - queued, r - rejected, b - backup
t - trapped, o - offload failure
B 0.0.0.0/0 [200/0] via 192.168.0.1, ens33, weight 1, 00:03:35
L * 2.2.2.2/32 is directly connected, lo, 00:40:12
C>* 2.2.2.2/32 is directly connected, lo, 00:40:12
B>* 3.3.3.3/32 [200/0] via 192.168.0.2, ens33, weight 1, 00:03:35
L * 4.4.4.4/32 is directly connected, lo, 00:40:12
C>* 4.4.4.4/32 is directly connected, lo, 00:40:12
C>* 192.168.0.0/24 is directly connected, ens33, 00:40:13
L>* 192.168.0.3/32 is directly connected, ens33, 00:40:13
```

6.1.2.6 Топология #2

В топологии #2 все системы балансировки (Angie ADC #1 и Angie ADC #2) принадлежат одной автономной системе, а маршрутизатор External router – другой:

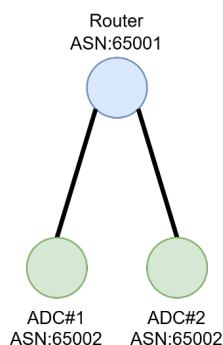


Рис. 9

Обычно на ADC #1 не нужны маршруты, объявляемые со стороны ADC #2. Если для конкретной сети вам нужно, чтобы ADC #1 знал маршруты, которые объявляет ADC #2 (или наоборот), то тогда на системе балансировки нагрузки надо разрешить прием маршрутов, содержащих собственный номер автономной системы в атрибуте AS-PATH.

Ключевые настройки маршрутизации для External router

На примере маршрутизатора Cisco Systems на базе IOS или IOS-XE:

```

!Имя устройства
hostname external
!Интерфейс в сторону систем балансировки
interface GigabitEthernet0/0
ip address 192.168.0.1 255.255.255.0
router bgp 65001
!Сессия с ADC#1
neighbor 192.168.0.2 remote-as 65002
!Сессия с ADC#2
neighbor 192.168.0.3 remote-as 65002
address-family ipv4
!Включение соседа
neighbor 192.168.0.2 activate
!Включение соседа
neighbor 192.168.0.3 activate
!Анонсирование маршрута по умолчанию
neighbor 192.168.0.2 default-originate
!Анонсирование маршрута по умолчанию
neighbor 192.168.0.3 default-originate
!Указание максимального количества eBGP путей, между которыми будет выполняться
↪балансировка
maximum-paths 2
end
  
```

Ключевые настройки маршрутизации Angie ADC #1

```

!Имя устройства
hostname adc1
!Интерфейс в сторону External router
interface ens33
ip address 192.168.0.2/24
  
```

```

exit
!Loopback-интерфейс для адресов виртуальных серверов
interface lo
ip address 2.2.2.2/32
ip address 3.3.3.3/32
exit
router bgp 65002
!Сессия с External router
neighbor 192.168.0.1 remote-as 65001
address-family ipv4 unicast
!Добавление префикса для адреса виртуального сервера в BGP
network 2.2.2.2/32
!Прием маршрутов, AS-PATH которых содержит собственный ASN
!Опция origin указывает на то, что приниматься будут только те маршруты,
!которые порождены в автономной системе с собственным ASN
neighbor 192.168.0.1 allowas-in origin
exit-address-family
exit
end

```

Ключевые настройки маршрутизации Angie ADC #2

```

!Имя устройства
hostname adc2
!Интерфейс в сторону External router
interface ens33
ip address 192.168.0.3/24
exit
!Loopback-интерфейс для адресов виртуальных серверов
interface lo
ip address 2.2.2.2/32
ip address 4.4.4.4/32
exit
router bgp 65002
!Сессия с External router
neighbor 192.168.0.1 remote-as 65001
address-family ipv4 unicast
!Добавление префикса для адреса виртуального сервера в BGP
network 2.2.2.2/32
!Прием маршрутов, AS-PATH которых содержит собственный ASN.
!Опция origin указывает на то, что приниматься будут только те маршруты,
!которые порождены в автономной системе с собственным ASN
neighbor 192.168.0.1 allowas-in origin
exit-address-family
exit
end

```

Проверка работы

```

external#sho ip bgp
BGP table version is 11, local router ID is 192.168.0.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

```



```

Network Next Hop Metric LocPrf Weight Path
0.0.0.0 0.0.0.0 0 i
*> 2.2.2.2/32 192.168.0.2 0 0 65002 i
*m 192.168.0.3 0 0 65002 i
external#sho ip ro
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
+ - replicated route, % - next hop override
Gateway of last resort is not set
2.0.0.0/32 is subnetted, 1 subnets
B 2.2.2.2 [20/0] via 192.168.0.2, 00:13:18
[20/0] via 192.168.0.3, 00:13:18
192.168.0.0/24 is variably subnetted, 2 subnets, 2 masks
C 192.168.0.0/24 is directly connected, GigabitEthernet0/0
L 192.168.0.1/32 is directly connected, GigabitEthernet0/0

```

Выводы просмотревых команд сокращены для краткости.

Для тестирования команды `allowas-in` мы завели на ADC #1 адрес 3.3.3.3/32, а на ADC #2 — 4.4.4.4/32.

Ниже представлены BGP-таблица и таблица маршрутизации от ADC #2:

```

adc2# sho ip bgp
BGP table version is 4, local router ID is 2.2.2.2, vrf id 0
Default local pref 100, local AS 65002
Status codes: s suppressed, d damped, h history, u unsorted, * valid, > best, =_
↔multipath,
i internal, r RIB-failure, S Stale, R Removed
Nexthop codes: @NNN nexthop's vrf id, < announce-nh-self
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found
Network Next Hop Metric LocPrf Weight Path
*> 0.0.0.0/0 192.168.0.1 0 65001 i
*> 2.2.2.2/32 0.0.0.0(adc2) 0 32768 i
* 192.168.0.2 0 65001 65002 i
*> 3.3.3.3/32 192.168.0.2 0 65001 65002 i
*> 4.4.4.4/32 0.0.0.0(adc2) 0 32768 i
* 192.168.0.1 0 65001 65002 i
Displayed 4 routes and 6 total paths
adc2# sho ip ro
Codes: K - kernel route, C - connected, L - local, S - static,
R - RIP, O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
T - Table, v - VNC, V - VNC-Direct, F - PBR,
f - OpenFabric, t - Table-Direct,
> - selected route, * - FIB route, q - queued, r - rejected, b - backup
t - trapped, o - offload failure
B 0.0.0.0/0 [20/0] via 192.168.0.1, ens33, weight 1, 00:07:26
L * 2.2.2.2/32 is directly connected, lo, 00:12:14
C>* 2.2.2.2/32 is directly connected, lo, 00:12:14
B>* 3.3.3.3/32 [20/0] via 192.168.0.2, ens33, weight 1, 00:01:02
L * 4.4.4.4/32 is directly connected, lo, 00:06:43
C>* 4.4.4.4/32 is directly connected, lo, 00:06:43
C>* 192.168.0.0/24 is directly connected, ens33, 00:12:15

```

```
L>* 192.168.0.3/32 is directly connected, ens33, 00:12:15
```

6.1.2.7 Топология #3

Все три устройства (External router, ADC #1 и ADC #2) принадлежат разным автономным системам:

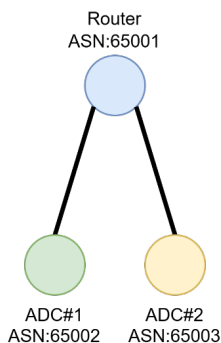


Рис. 10

Обычно маршрутизаторы Cisco не балансируют трафик до подсети, известной через соседей из разных автономных систем. Чтобы обойти данное ограничение, необходимо изменить поведение по умолчанию маршрутизатора BGP с помощью следующей команды:

```
bgp bestpath as-path multipath-relax
```

Ключевые настройки маршрутизации для External router

На примере маршрутизатора Cisco Systems на базе IOS или IOS-XE:

```

!Имя устройства
hostname external
!Интерфейс в сторону систем балансировки
interface GigabitEthernet0/0
ip address 192.168.0.1 255.255.255.0
router bgp 65001
!Команда, разрешающая выполнять балансировку для префиксов, полученных из разных ASN
bgp bestpath as-path multipath-relax
!Сессия с ADC#1
neighbor 192.168.0.2 remote-as 65002
!Сессия с ADC#2
neighbor 192.168.0.3 remote-as 65003
address-family ipv4
!Включение соседа
neighbor 192.168.0.2 activate
!Включение соседа
neighbor 192.168.0.3 activate
!Анонсирование маршрута по умолчанию
neighbor 192.168.0.2 default-originate
!Анонсирование маршрута по умолчанию
neighbor 192.168.0.3 default-originate
!Указание максимального количества eBGP путей, между которыми будет выполняться
↪балансировка
maximum-paths 2
end
  
```

Ключевые настройки маршрутизации Angie ADC #1

```
!Имя устройства
hostname adc1
!Интерфейс в сторону External router
interface ens33
ip address 192.168.0.2/24
exit
!Loopback-интерфейс для адресов виртуальных серверов
interface lo
ip address 2.2.2.2/32
exit
router bgp 65002
!Сессия с External router
neighbor 192.168.0.1 remote-as 65001
address-family ipv4 unicast
!Добавление префикса для адреса виртуального сервера в BGP
network 2.2.2.2/32
exit-address-family
exit
end
```

Ключевые настройки маршрутизации Angie ADC #2

```
!Имя устройства
hostname adc2
!Интерфейс в сторону External router
interface ens33
ip address 192.168.0.3/24
exit
!Loopback-интерфейс для адресов виртуальных серверов
interface lo
ip address 2.2.2.2/32
exit
router bgp 65003
!Сессия с External router
neighbor 192.168.0.1 remote-as 65001
address-family ipv4 unicast
!Добавление префикса для адреса виртуального сервера в BGP
network 2.2.2.2/32
exit-address-family
exit
end
```

Проверка работы

```
external#sho ip bgp
BGP table version is 11, local router ID is 192.168.0.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found
Network Next Hop Metric LocPrf Weight Path
0.0.0.0 0.0.0.0 0 i
*> 2.2.2.2/32 192.168.0.2 0 0 65002 i
```

```
*m 192.168.0.3 0 0 65003 i
external#sho ip ro
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
+ - replicated route, % - next hop override
Gateway of last resort is not set
2.0.0.0/32 is subnetted, 1 subnets
B 2.2.2.2 [20/0] via 192.168.0.2, 00:10:22
[20/0] via 192.168.0.3, 00:16:07
192.168.0.0/24 is variably subnetted, 2 subnets, 2 masks
C 192.168.0.0/24 is directly connected, GigabitEthernet0/0
L 192.168.0.1/32 is directly connected, GigabitEthernet0/0
```

Выводы просмотревших команд сокращены для краткости.

6.1.2.8 Применение с IP transparency

Все описанное выше относится к режиму работы Angie ADC с выключенной опцией IP transparency. В этом случае Angie ADC будет выполнять проксирование клиентских запросов с подменой адресов источника и получателя. Если Angie ADC подменяет адрес клиента, то при маршрутизации нет необходимости анонсировать на Internal router маршрут по умолчанию. Если ADC #1 и ADC #2 используют разные адреса для подмены адреса клиента, то нет необходимости заботиться о том, с какими значениями Local preference (и иными атрибутами пути) маршруты до соответствующих адресов попадают на Internal router.

При включении опции IP Transparency Angie ADC будет сохранять исходный адрес отправителя в IP-пакетах, позволяя серверам на сетевом уровне различать клиентские подключения.

6.1.2.9 Заключение

Система балансировки нагрузки Angie ADC может работать в любой из перечисленных топологий. Каждый вариант имеет свои особенности настройки и работы, о чем следует помнить при проектировании и настройке информационной системы.

6.2 OSPF-маршрутизация

В этом разделе описана настройка OSPF-маршрутизации для обеспечения высокой доступности в режиме резервирования (Active-Standby) и в режиме распределения нагрузки (Active-Active). Для настройки используется *интерфейс командной строки (CLI) Angie ADC*.

В этом разделе:

- *Режим резервирования (Active-Standby)*
- *Режим распределения нагрузки (Active-Active)*

6.2.1 Резервирование с помощью протокола OSPF (Active-Standby)

6.2.1.1 Введение

В этом разделе описана настройка высокой доступности с использованием протокола OSPF в режиме резервирования (Active-Standby). Для настройки OSPF используется *интерфейс командной строки (CLI) Angie ADC*.

В качестве протокола динамической маршрутизации используется OSPFv2 для адресного семейства IPv4 unicast. Отказоустойчивость в рамках IPv6 unicast настраивается аналогично.

Для обеспечения отказоустойчивости нужны две системы балансировки Angie ADC — основная и резервная. Резервная система Angie ADC будет принимать нагрузку при отказе основной.

Пример типового участка сети с системами балансировки Angie ADC (ADC #1 и ADC #2):



Рис. 1

Добавим IP-адресацию:

- В сегменте сети, куда подключаются External router, ADC #1 и ADC #2, будет использоваться подсеть 192.168.0.0/24.
- В сегменте сети, куда подключаются Internal router, ADC #1 и ADC #2, будет использоваться подсеть 192.168.1.0/24.
- Четвертый октет адреса для каждого устройства (в рамках соответствующей подсети) показан на схеме ниже.

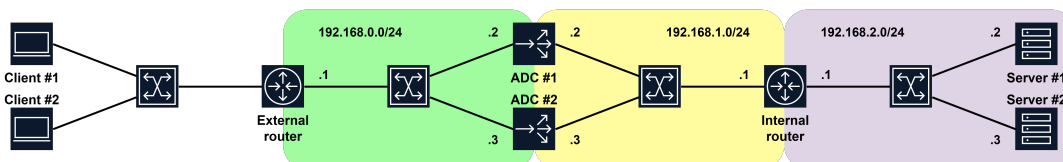


Рис. 2

Адресация клиентов не приводится, так как в данном случае мы будем считать, что их адреса формально не определены, то есть, они могут подключаться из любой сети, включая сеть интернет. Для этих целей External router будет анонсировать маршрут по умолчанию по OSPF для ADC #1 и ADC #2. Мы будем использовать OSPF-домен, включающий в себя четыре устройства: External router, Internal router, ADC #1 и ADC #2.

Красным текстом на схеме ниже отмечены IP-адреса виртуальных серверов. Это те адреса, к которым подключаются клиенты снаружи. Системы балансировки Angie ADC должны анонсировать маршруты до этих виртуальных серверов в сторону External router, причем маршрут через ADC #1 должен быть более предпочтительным. Добиться этого можно разными способами, например, манипулируя значением метрики внешнего маршрута в OSPF. Чем больше значение метрики, тем менее предпочтительный маршрут. В данном документе мы будем вручную задавать косты для интерфейсов системы балансировки.

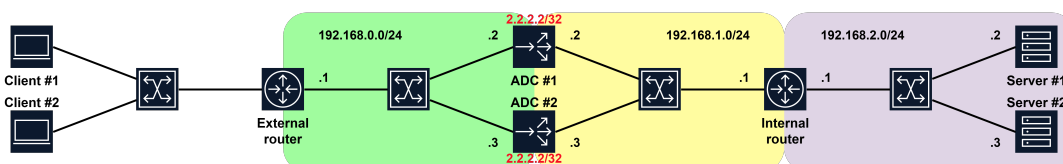


Рис. 3

Трафик от серверов в сторону клиента должен проходить через ту же систему балансировки, что и трафик от клиентов к серверам. Добиться этого можно также путем манипулирования значением метрики для префикса 0.0.0.0/0. Манипуляции с маршрутом по умолчанию нужны при включении опции IP transparency. В противном случае трафик и так будет возвращаться через ту же систему балансировки, так как поведение ADC напоминает операцию sNAT.

6.2.1.2 Обработка типовых сбоев

Ниже описаны сценарии работы Angie ADC при сбоях.

Полный отказ ADC #1

В этом случае оба OSPF-соседства с ADC #1 разрываются, никакие объявления маршрутов через ADC #1 не пролетают. Остается связь по OSPF только с ADC #2. То есть, переключение на резервную систему производится полностью автоматически, никакие дополнительные настройки не требуются.

Красными стрелками обозначены рабочие OSPF-соседства:

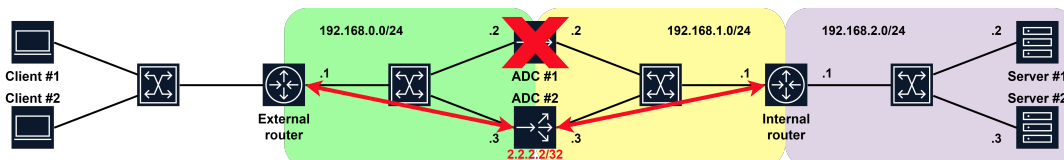


Рис. 4

Отказ внешнего интерфейса ADC #1

Предлагаемая схема не предполагает возможности отслеживания частичных отказов и реагирования на них. Пример такого отказа представлен на схеме ниже.

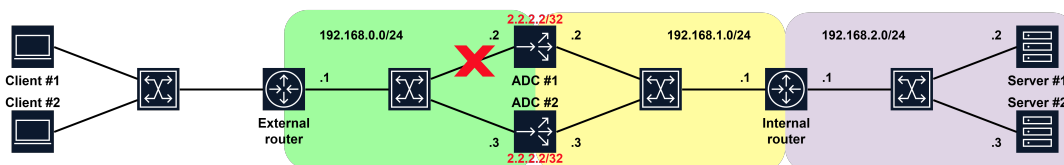


Рис. 5

В случае обозначенного на схеме отказа линка разрывается OSPF-сессия между ADC #1 и External router. ADC #1 теперь не может объявить маршрут до адреса виртуального сервера в сторону External router напрямую. Internal router рассчитывает маршрут по умолчанию только через ADC #1, а External router считает маршрут до адреса виртуального сервера только через полностью работающий ADC #2.

Отказ линка между ADC #1 и Internal router

Отказ линка между ADC #1 и Internal router более сложная ситуация: если OSPF-соседство между External router и ADC #1 сохраняется, то ADC #1 будет продолжать анонсировать маршрут до адреса виртуального сервера, хотя реальные сервера для системы балансировки ADC #1 уже недостижимы.

Таким образом, мы получаем классический пример black hole для трафика:

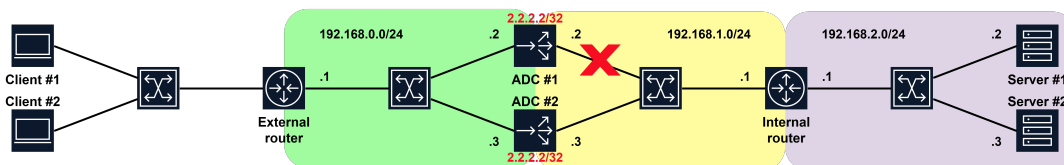


Рис. 6

Чтобы избежать этой проблемы, необходимо, чтобы ситуация отказа только одного из линков была невозможной, т.е. нужно обеспечить отказоустойчивое подключение к сети для хоста виртуализации, на котором работает ADC.

Рекомендованная схема подключения:

- Объединить в одну LAG-группу все физические интерфейсы хоста виртуализации и использовать этот виртуальный LAG-интерфейс в качестве транка, то есть передавать по нему агрегированными несколько виртуальных сетей (один VLAN для external сети, второй VLAN для internal сети).
- Для резервирования коммутаторов можно использовать технологию vPC или стекирование, пример представлен на схеме ниже.

Из соображений простоты на схеме не отображено резервирование внешнего и внутреннего маршрутизаторов.

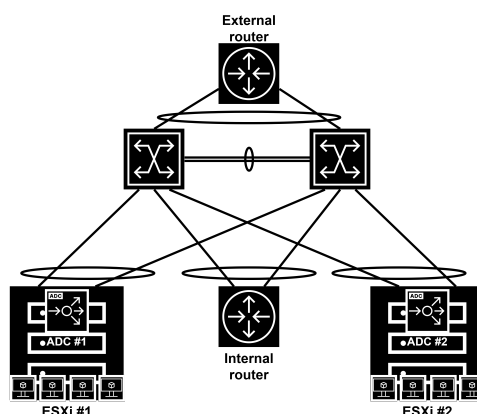


Рис. 7

6.2.1.3 Настройки маршрутизации

Приведем ключевые настройки маршрутизации для External router, Internal router, ADC #1 и ADC #2.

Ключевые настройки маршрутизации для External router

```
hostname external
!Имя устройства
interface GigabitEthernet0/0
 ip address 192.168.0.1 255.255.255.0
!Интерфейс в сторону систем балансировки
router ospf 1
 network 192.168.0.0 0.0.0.255 area 0
!Включение OSPF на интерфейсе в сторону систем балансировки
 default-information originate always
!Анонсирование маршрута по умолчанию
end
```

Ключевые настройки маршрутизации для Internal router

```
hostname internal
!Имя устройства
interface GigabitEthernet0/0
 ip address 192.168.1.1 255.255.255.0
```

```
!Интерфейс в сторону систем балансировки
interface GigabitEthernet1/0
ip address 192.168.2.1 255.255.255.0
!Интерфейс в сторону подсети с реальными серверами
router ospf 1
 redistribute connected subnets
!добавление в OSPF информации о подсети с серверами
 network 192.168.1.0 0.0.0.255 area 0
!Включение OSPF на интерфейсе в сторону систем балансировки
end
```

Ключевые настройки маршрутизации системы балансировки ADC #1

```
hostname adc1
!Имя устройства
interface ens33
!Интерфейс в сторону маршрутизатора external router
 description external
 ip address 192.168.0.2/24
 ip ospf 1 area 0
!Включение OSPF на интерфейсе
 ip ospf cost 10
!Назначение стоимости интерфейса в OSPF вручную
exit
interface ens37
!Интерфейс в сторону маршрутизатора internal router
 description internal
 ip address 192.168.1.2/24
 ip ospf 1 area 0
!Включение OSPF на интерфейсе
exit
interface lo
!Loopback-интерфейс для назначения адресов виртуальных серверов
 ip address 2.2.2.2/32
!Адрес тестового виртуального сервера
exit
ip prefix-list c2o seq 5 permit 2.2.2.2/32
!Префикс-лист, который используется для отбора префиксов для добавления в OSPF
route-map c2o permit 10
 match ip address prefix-list c2o
!Route-мар с помощью которой производится отбор префиксов для добавления в OSPF
exit
router ospf 1
!Настройки процесса маршрутизации OSPF
 ospf router-id 2.2.2.2
!Необходимо вручную задать идентификатор маршрутизатора для OSPF
!Если этого не сделать, то автоматика может выбрать одинаковые для ADC#1 и ADC#2
 redistribute connected metric 10 route-map c2o
!Добавление информации о подключенных сетях в OSPF
!Метрика 10, чтобы стать более предпочтительным next-hop по сравнению с ADC#2
exit
end
```


Ключевые настройки маршрутизации системы балансировки ADC #2

```
hostname adc2
!Имя устройства
interface ens33
!Интерфейс в сторону маршрутизатора external router
description external
ip address 192.168.0.3/24
ip ospf 1 area 0
!Включение OSPF на интерфейсе
ip ospf cost 100
!Назначение стоимости интерфейса в OSPF вручную
exit
interface ens37
!Интерфейс в сторону маршрутизатора internal router
description internal
ip address 192.168.1.3/24
ip ospf 1 area 0
!Включение OSPF на интерфейсе
exit
interface lo
!Loopback-интерфейс для назначения адресов виртуальных серверов
ip address 2.2.2.2/32
!Адрес тестового виртуального сервера
exit
ip prefix-list c2o seq 5 permit 2.2.2.2/32
!Префикс-лист, который используется для отбора префиксов для добавления в OSPF
route-map c2o permit 10
match ip address prefix-list c2o
!Route-map с помощью которой производится отбор префиксов для добавления в OSPF
exit
router ospf 1
!Настройки процесса маршрутизации OSPF
ospf router-id 22.22.22.22
!Необходимо вручную задать идентификатор маршрутизатора для OSPF
!Если этого не сделать, то автоматика может выбрать одинаковые для ADC#1 и ADC#2
redistribute connected metric 20 route-map c2o
!Добавление информации о подключенных сетях в OSPF
!Метрика 20, чтобы стать менее предпочтительным next-hop по сравнению с ADC#1
exit
end
```

Проверка работы

Выполним проверку работы системы маршрутизации, когда ADC #1 и ADC #2 находятся в полностью работоспособном состоянии.

Примечание

В листингах ниже отсутствуют префиксы, не относящиеся к обсуждаемым вопросам отказоустойчивости.

Маршрутизатор External router видит маршрут к адресу виртуального сервера через обе системы балансировки, однако маршрут через ADC #1 является более предпочтительным.

```
external#sho ip ro os
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
```

```

D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
+ - replicated route, % - next hop override
Gateway of last resort is not set
  2.0.0.0/32 is subnetted, 1 subnets
O E2    2.2.2.2 [110/10] via 192.168.0.2, 00:41:28, GigabitEthernet0/0
external#sho ip os da ex 2.2.2.2
      OSPF Router with ID (192.168.0.1) (Process ID 1)
        Type-5 AS External Link States
Routing Bit Set on this LSA in topology Base with MTID 0

LS age: 898
Options: (No TOS-capability, No DC, Upward)
LS Type: AS External Link
Link State ID: 2.2.2.2 (External Network Number )
Advertising Router: 2.2.2.2
LS Seq Number: 80000003
Checksum: 0xAA0C
Length: 36
Network Mask: /32
    Metric Type: 2 (Larger than any link state path)
    MTID: 0
    Metric: 10
    Forward Address: 0.0.0.0
    External Route Tag: 0

LS age: 885
Options: (No TOS-capability, No DC, Upward)
LS Type: AS External Link
Link State ID: 2.2.2.2 (External Network Number )
Advertising Router: 22.22.22.22
LS Seq Number: 80000003
Checksum: 0xB4A7
Length: 36
Network Mask: /32
    Metric Type: 2 (Larger than any link state path)
    MTID: 0
    Metric: 20
    Forward Address: 0.0.0.0
    External Route Tag: 0

```

Маршрутизатор Internal router видит маршрут по умолчанию (путь через ADC #1 является более предпочтительным).

```

internal#sho ip ro os
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
+ - replicated route, % - next hop override
Gateway of last resort is 192.168.1.2 to network 0.0.0.0

```

```

0*E2 0.0.0.0/0 [110/1] via 192.168.1.2, 00:43:21, GigabitEthernet0/0
internal#sho ip os da ext 0.0.0.0
      OSPF Router with ID (192.168.2.1) (Process ID 1)
      Type-5 AS External Link States
Routing Bit Set on this LSA in topology Base with MTID 0
LS age: 992
Options: (No TOS-capability, DC, Upward)
LS Type: AS External Link
Link State ID: 0.0.0.0 (External Network Number )
Advertising Router: 192.168.0.1
LS Seq Number: 80000003
Checksum: 0x2521
Length: 36
Network Mask: /0
      Metric Type: 2 (Larger than any link state path)
      MTID: 0
      Metric: 1
      Forward Address: 0.0.0.0
      External Route Tag: 1

```

Отключим теперь ADC #1 полностью и проверим маршрутизацию.

Маршрутизатор Internal router:

```

internal#sho ip ro os
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       + - replicated route, % - next hop override
Gateway of last resort is 192.168.1.3 to network 0.0.0.0
0*E2 0.0.0.0/0 [110/1] via 192.168.1.3, 00:01:08, GigabitEthernet0/0

```

Маршрутизатор External router:

```

external#sho ip ro os
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       + - replicated route, % - next hop override
Gateway of last resort is not set
      2.0.0.0/32 is subnetted, 1 subnets
O E2      2.2.2.2 [110/10] via 192.168.0.3, 00:00:10, GigabitEthernet0/0

```

После возвращения в строй системы балансировки ADC #1 маршрутизация через это устройство восстанавливается.

6.2.1.4 Заключение

Все описанное выше относится к режиму работы системы балансировки трафика с включенной опцией IP transparency. При опции IP Transparency Angie ADC сохраняет исходный адрес отправителя в IP-пакетах, позволяя серверам на сетевом уровне различать клиентские подключения. Очень часто администраторы систем балансировки отключают эту опцию, в этом случае Angie ADC будет выполнять не только dNAT, но sNAT над трафиком, приходящим от клиентов. Если система балансировки выполняет sNAT, то в этом случае маршрутизация несколько упрощается, так как в этом случае нет необходимости анонсировать в сторону Internal router маршрут по умолчанию. Если ADC #1 и ADC #2 выполняют sNAT в разные адреса, то в этом случае нет необходимости заботиться о том, с какими значениями метрики маршруты до соответствующих адресов попадают на маршрутизатор Internal router, так как анонсы будут уникальными.

6.2.2 Распределение нагрузки с помощью протокола OSPF (Active-Active)

6.2.2.1 Введение

В этом разделе описана настройка высокой доступности с использованием протокола OSPF в режиме распределения нагрузки (Active-Active). Для настройки OSPF используется *интерфейс командной строки (CLI) Angie ADC*.

В качестве протокола динамической маршрутизации используется OSPFv2 для адресного семейства IPv4 unicast. Отказоустойчивость в рамках IPv6 unicast настраивается аналогично.

Для обеспечения распределения нагрузки между ADC нужны две системы балансировки Angie, работающие параллельно и одновременно. Пример типового участка сети с системами балансировки Angie ADC (ADC #1 и ADC #2):

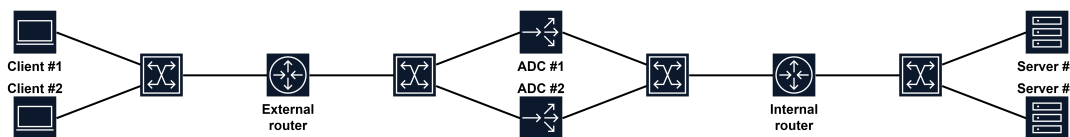


Рис. 1

Добавим IP-адресацию:

- В сегменте сети, куда подключаются External router, ADC #1 и ADC #2, будет использоваться подсеть 192.168.0.0/24.
- В сегменте сети, куда подключаются Internal router, ADC #1 и ADC #2, будет использоваться подсеть 192.168.1.0/24.
- Четвертый октет адреса для каждого устройства (в рамках соответствующей подсети) показан на схеме ниже.

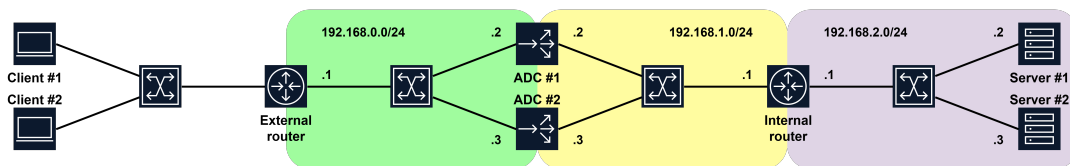


Рис. 2

Адресация клиентов не приводится, так как в данном случае мы будем считать, что их адреса формально не определены, то есть, они могут подключаться из любой сети, включая сеть интернет. Для этих целей External router будет анонсировать маршрут по умолчанию по OSPF для ADC #1 и ADC #2. Мы будем использовать OSPF-домен, включающий в себя четыре устройства: External router, Internal router, ADC #1 и ADC #2.

Красным текстом на схеме ниже отмечены *IP-адреса виртуальных серверов*. Это те адреса, к которым подключаются клиенты снаружи. Чтобы маршрутизатор External router мог выполнить распределение трафика между системами балансировки Angie ADC (см. ECMP – Equal-Cost Multi-Path), они должны одинаково анонсировать маршруты до виртуальных серверов в сторону External router (должны совпадать типы маршрутов и косты).

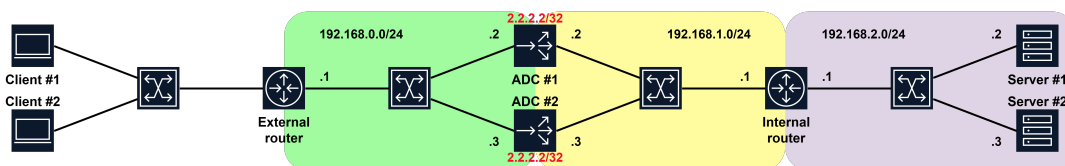


Рис. 3

Трафик внутри пользовательской сессии от серверов должен возвращаться строго через ту же систему балансировки, через которую он проходил до этого. Добиться этого можно с помощью механизма *SNAT*, когда адрес клиента подменяется на адрес интерфейса системы балансировки.

6.2.2.2 Обработка типовых сбоев

Ниже описаны сценарии работы Angie ADC при сбоях.

Полный отказ ADC #1

В этом случае оба OSPF-соседства с ADC #1 разрываются, никакие объявления маршрутов через ADC #1 не пролетают. Остается связь по OSPF только с ADC #2. То есть, переключение на резервную систему производится полностью автоматически, никакие дополнительные настройки не требуются.

Красными стрелками обозначены рабочие OSPF-соседства:

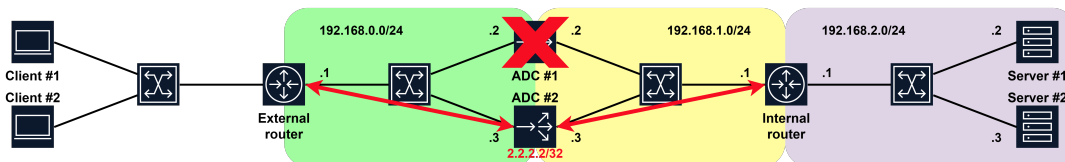


Рис. 4

Отказ внешнего интерфейса ADC #1

Пример такого отказа представлен на схеме ниже.

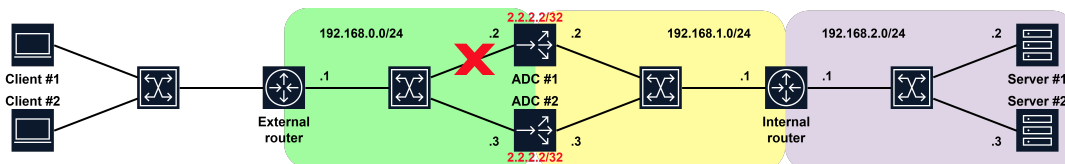


Рис. 5

В случае обозначенного на схеме отказа линка разрывается OSPF-сессия между ADC #1 и External router. ADC #1 теперь не может объявить маршрут до адреса виртуального сервера в сторону External router напрямую. Internal router получает маршруты до адресов, которые используются в SNAT от обеих систем балансировки, однако трафик от клиентов, прошедший через ADC #2,

будет возвращаться тем же путем (так же через ADC #2), так как ADC #1 и ADC #2 используют уникальные адреса для SNAT.

Отказ линка между ADC #1 и Internal router

Отказ линка между ADC #1 и Internal router более сложная ситуация: если OSPF-соседство между External router и ADC #1 сохраняется, то ADC #1 будет продолжать анонсировать маршрут до адреса виртуального сервера, хотя реальные сервера для системы балансировки ADC #1 уже недоступны.

Таким образом, мы получаем классический пример black hole для трафика:

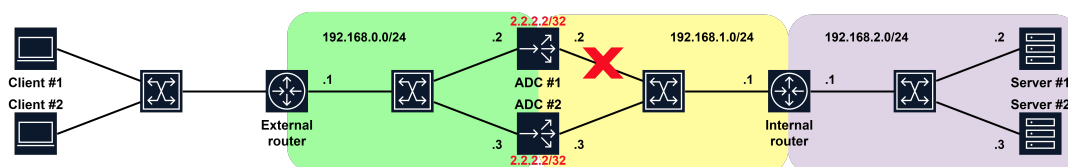


Рис. 6

Чтобы избежать этой проблемы, необходимо, чтобы ситуация отказа только одного из линков была невозможной, т.е. нужно обеспечить отказоустойчивое подключение к сети для хоста виртуализации, на котором работает ADC.

Вторым способом решить указанную проблему является *технология RHI*, позволяющая отозвать маршрут при недоступности апстрим-серверов.

Рекомендованная схема подключения:

- Объединить в одну LAG-группу все физические интерфейсы хоста виртуализации и использовать этот виртуальный LAG-интерфейс в качестве транка, то есть передавать по нему агрегированными несколько виртуальных сетей (один VLAN для внешней (external) сети, второй VLAN для внутренней (internal) сети).
- Для резервирования коммутаторов можно использовать такие технологии как vPC, VSS или стекирование, пример представлен на схеме ниже.

Из соображений простоты на схеме не отображено резервирование внешнего и внутреннего маршрутизаторов.

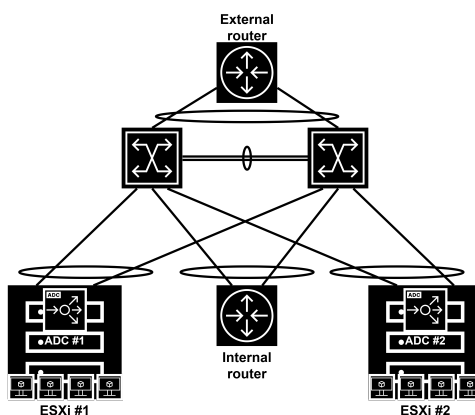


Рис. 7

Предупреждение

Маршрутизатор, выполняющий ECMP-балансировку, и обе системы Angie ADC могут располагаться в одной OSPF-зоне или в разных. Если при этом маршрутизатор является OSPF ABR (Area Border Router), а ADC #1 и ADC #2 находятся в разных OSPF-зонах (видны через разные интерфейсы ABR), то ECMP-балансировка может не поддерживаться некоторыми вендорами сетевого оборудования.

Такой дизайн сети не рекомендуется:

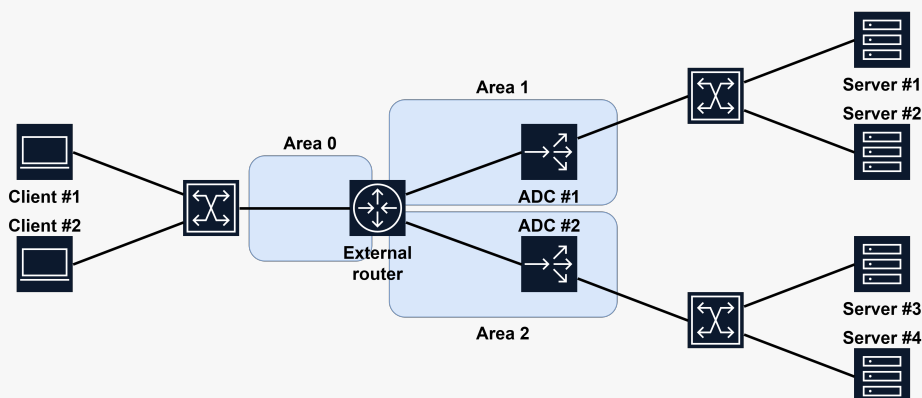


Рис. 8

6.2.2.3 Настройки маршрутизации

Приведем ключевые настройки маршрутизации для External router, Internal router, ADC #1 и ADC #2.

Ключевые настройки маршрутизации для External router

```
!Имя устройства
hostname external
!Интерфейс в сторону систем балансировки
interface GigabitEthernet0/0
ip address 192.168.0.1 255.255.255.0
router ospf 1
!Включение OSPF на интерфейсе в сторону систем балансировки
network 192.168.0.0 0.0.0.255 area 0
!Анонсирование маршрута по умолчанию
default-information originate always
end
```

Ключевые настройки маршрутизации для Internal router

```
!Имя устройства
hostname internal
!Интерфейс в сторону систем балансировки
interface GigabitEthernet0/0
ip address 192.168.1.1 255.255.255.0
!Интерфейс в сторону подсети с реальными серверами
interface GigabitEthernet1/0
ip address 192.168.2.1 255.255.255.0
router ospf 1
!Добавление в OSPF информации о подсети с серверами
redistribute connected subnets
!Включение OSPF на интерфейсе в сторону систем балансировки
```

```
network 192.168.1.0 0.0.0.255 area 0
end
```

Ключевые настройки маршрутизации системы балансировки ADC #1

```
!Имя устройства
hostname adc1
!Интерфейс в сторону маршрутизатора external router
interface ens33
description external
ip address 192.168.0.2/24
!Включение OSPF на интерфейсе
ip ospf 1 area 0
exit
!Интерфейс в сторону маршрутизатора internal router
interface ens37
description internal
ip address 192.168.1.2/24
!Включение OSPF на интерфейсе
ip ospf 1 area 0
exit
!Loopback-интерфейс для назначения адресов виртуальных серверов
interface lo
!Адрес тестового виртуального сервера
ip address 2.2.2.2/32
exit
!Префикс-лист, который используется для отбора префиксов для добавления в OSPF
ip prefix-list c2o seq 5 permit 2.2.2.2/32
!Route-мар, с помощью которой производится отбор префиксов для добавления в OSPF
route-map c2o permit 10
match ip address prefix-list c2o
exit
!Настройки процесса маршрутизации OSPF
router ospf 1
!Необходимо вручную задать идентификатор маршрутизатора для OSPF
!Иначе автоматика может выбрать одинаковые идентификаторы для ADC#1 и ADC#2
ospf router-id 2.2.2.2
!Добавление информации о подключенных сетях в OSPF
redistribute connected route-map c2o
exit
end
```

Ключевые настройки маршрутизации системы балансировки ADC #2

```
!Имя устройства
hostname adc2
!Интерфейс в сторону маршрутизатора external router
interface ens33
description external
ip address 192.168.0.3/24
!Включение OSPF на интерфейсе
ip ospf 1 area 0
exit
!Интерфейс в сторону маршрутизатора internal router
interface ens37
description internal
```



```

ip address 192.168.1.3/24
!Включение OSPF на интерфейсе
ip ospf 1 area 0
exit
!Loopback-интерфейс для назначения адресов виртуальных серверов
interface lo
!Адрес тестового виртуального сервера
ip address 2.2.2.2/32
exit
!Префикс-лист, который используется для отбора префиксов для добавления в OSPF
ip prefix-list c2o seq 5 permit 2.2.2.2/32
!Route-map, с помощью которой производится отбор префиксов для добавления в OSPF
route-map c2o permit 10
match ip address prefix-list c2o
exit
!Настройки процесса маршрутизации OSPF
router ospf 1
!Необходимо вручную задать идентификатор маршрутизатора для OSPF
!Иначе автоматика может выбрать одинаковые идентификаторы для ADC#1 и ADC#2
ospf router-id 22.22.22.22
!Добавление информации о подключенных сетях в OSPF
redistribute connected route-map c2o
exit
end

```

6.2.2.4 Проверка работы

Выполним проверку работы системы маршрутизации, когда ADC #1 и ADC #2 находятся в полностью работоспособном состоянии.

Примечание

В листингах ниже отсутствуют префиксы, не относящиеся к обсуждаемым вопросам отказоустойчивости.

Маршрутизатор External router видит маршрут к адресу виртуального сервера через обе системы балансировки.

```

external#sho ip ro os
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
+ - replicated route, % - next hop override
Gateway of last resort is not set
2.0.0.0/32 is subnetted, 1 subnets
O E2      2.2.2.2 [110/20] via 192.168.0.2, 00:41:28, GigabitEthernet0/0
via 192.168.0.3, 00:41:15, GigabitEthernet0/0
external#sho ip os da ex 2.2.2.2
OSPF Router with ID (192.168.0.1) (Process ID 1)
Type-5 AS External Link States
Routing Bit Set on this LSA in topology Base with MTID 0

```

```
LS age: 898
Options: (No TOS-capability, No DC, Upward)
LS Type: AS External Link
Link State ID: 2.2.2.2 (External Network Number )
Advertising Router: 2.2.2.2
LS Seq Number: 80000003
Checksum: 0xAA0C
Length: 36
Network Mask: /32
Metric Type: 2 (Larger than any link state path)
MTID: 0
Metric: 20
Forward Address: 0.0.0.0
External Route Tag: 0
```

```
LS age: 885
Options: (No TOS-capability, No DC, Upward)
LS Type: AS External Link
Link State ID: 2.2.2.2 (External Network Number )
Advertising Router: 22.22.22.22
LS Seq Number: 80000003
Checksum: 0xB4A7
Length: 36
Network Mask: /32
Metric Type: 2 (Larger than any link state path)
MTID: 0
Metric: 20
Forward Address: 0.0.0.0
External Route Tag: 0
```

По умолчанию Angie ADC будет устанавливать соединение с апстрим-сервером с адреса того интерфейса, через который этот сервер достижим, в соответствии с таблицей маршрутизации (192.168.1.2 для ADC #1 и 192.168.1.3 для ADC #2). Оба адреса принадлежат подсети 192.168.1.0/24, которая для маршрутизатора Internal router является непосредственно подключенной (Directly Connected). Анонсировать другие адреса не нужно. Если используется SNAT pool, то каждая из систем балансировки объявит в сеть дополнительные адреса, соответствующие пулу.

Отключим теперь ADC #1 полностью и проверим маршрутизацию.

Маршрутизатор External router:

```
external#sho ip ro os
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
+ - replicated route, % - next hop override
Gateway of last resort is not set
2.0.0.0/32 is subnetted, 1 subnets
O E2      2.2.2.2 [110/20] via 192.168.0.3, 01:40:10, GigabitEthernet0/0
```

После возвращения в строй ADC #1 маршрутизация через это устройство восстанавливается.

6.2.2.5 Применение с IP transparency

Все описанное выше относится к режиму работы системы балансировки трафика с отключенной опцией IP transparency. При включении опции IP Transparency Angie ADC сохраняет исходный адрес отправителя в IP-пакетах, позволяя серверам на сетевом уровне различать клиентские подключения. Однако в этом случае маршрутизатор Internal router будет возвращать трафик через неправильную систему балансировки. Исходный адрес клиента может быть передан серверу либо с помощью заголовка X-FORWARDED-FOR, либо с помощью специального proxy protocol.

6.3 VRRP-маршрутизация

VRRP (Virtual Router Redundancy Protocol) — это протокол для виртуального резервирования устройств. VRRP позволяет переключиться на резервный Angie ADC, который автоматически принимает на себя IP-адрес и роль мастера, если основное устройство Angie ADC отказало. Таким образом обеспечивается отказоустойчивость для подключенных узлов в сети при статической маршрутизации. Поддерживаемые версии: VRRPv2 (для IPv4) и VRRPv3 (для IPv4 и IPv6).

Примечание

Настройка VRRP осуществляется через *командную строку (CLI)* на порту 2022.

Важно

Настройки VRRP не отображаются в веб-интерфейсе. Не рекомендуется одновременно использовать ручную настройку высокой доступности по VRRP и настройку VRRP *в рамках пары высокой доступности*.

В этом разделе:

- *Предварительная настройка для VMware ESXi*
- *Режим резервирования (Active-Standby)*
- *Режим распределения нагрузки (Active-Active)*
- *Команды VRRP*
- *Создание и удаление VRRP-групп*

6.3.1 Предварительная настройка для VMware ESXi

Если Angie ADC развернут на VMware ESXi, то для работы VRRP необходимо дополнительно настроить сетевые политики безопасности ESXi и разрешить передачу трафика с портов, участвующих в VRRP.

6.3.1.1 Настройка VMware ESXi до версии 6.7

Для VMware ESXi версии до версии 6.7 необходимо создать отдельную порт-группу (Port Group) для портов, участвующих в VRRP, и включить для нее настройки **Promiscuous mode** и **Forged Transmits**.

1. Создайте отдельную порт-группу (Port Group) для портов, участвующих в VRRP.

Пример пути для создания порт-группы: Host → Configure → Networking → Port Groups → Add Port Group.

2. Задайте следующие значения для передачи трафика при изменении MAC-адреса для новой порт-группы:
 - Promiscuous Mode – Accept.

- Forged transmits – Accept.

Пример пути для настройки порт-группы: Host → Configure → Networking → Port Groups → <порт_группа> → Edit Settings → Security.

3. Обеспечьте размещение виртуальных машин Angie ADC на разных ESXi-хостах кластера. Для этого запретите DRS-перемещение виртуальных машин Angie ADC на один хост с помощью запрещающего правила Anti-Affinity.

Пример пути для настройки правила: Host → Configure → Configuration → VM/Host Rules → Add.

В открывшемся окне заполните необходимые поля и укажите виртуальные машины Angie ADC.

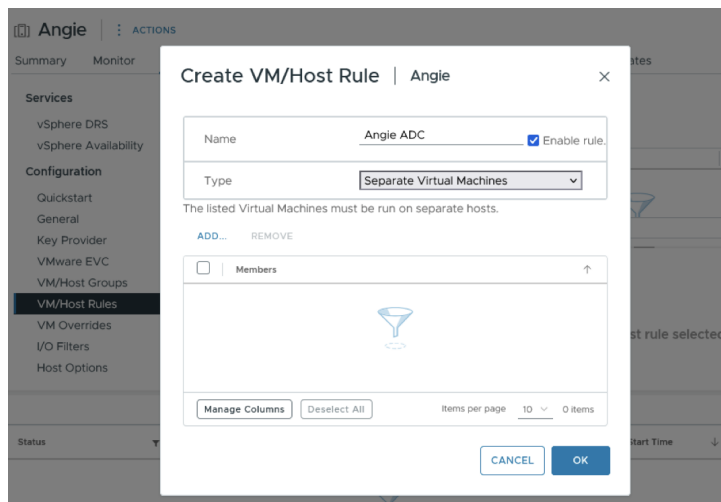


Рис. 1: создание запрещающего правила для ESXi до версии 6.7.

Общая схема будет такой: устройства Angie ADC подключены через отдельную порт-группу, на которой включен режим Promiscuous:

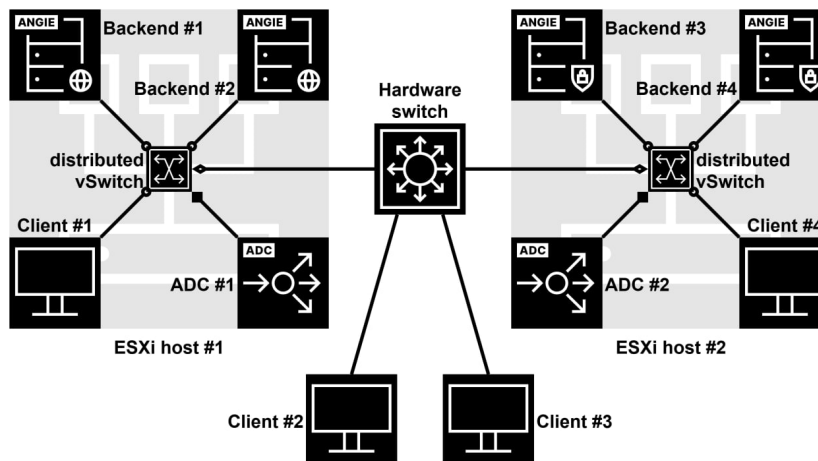


Рис. 2: подключение Angie ADC для ESXi до версии 6.7.

6.3.1.2 Настройка VMware ESXi версии 6.7+

Для VMware ESXi версии 6.7+ необходимо настроить обычную порт-группу (Port Group) для портов, участвующих в VRRP, и включить для нее настройки **Forged Transmits** и **MAC Learning**.

1. Подключитесь к vCenter Server и перейдите Inventory → Networks.
2. Выберите DSwitch и щелкните правой кнопкой мыши по нужной порт-группе. В контекстном меню выберите пункт Edit Settings → Security.

3. Задайте следующие значения для передачи трафика при изменении MAC-адреса:

- Promiscuous Mode – Reject.
- MAC address changes – Reject.
- Forged transmits – Accept.

4. В блоке MAC Learning установите Status в значение Enabled.

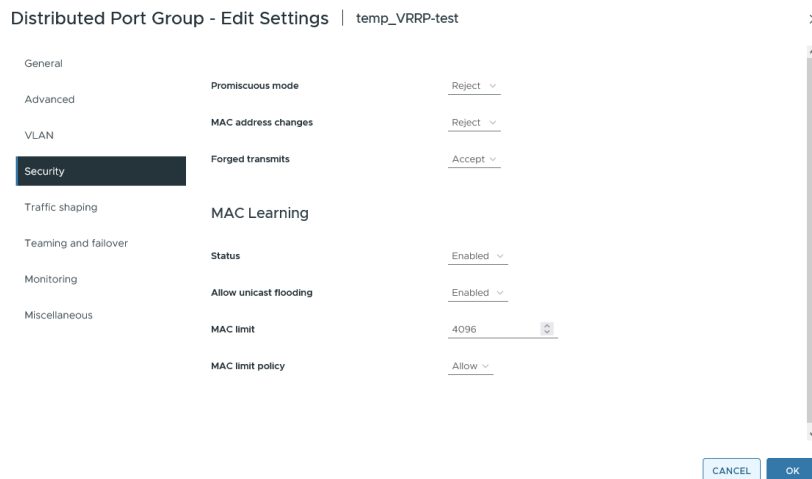


Рис. 3: настройка порт-группы на ESXi 6.7+.

Общая схема будет такой: устройства Angie ADC подключены через обычные порт-группы, при этом включена политика, разрешающая подмену MAC-адресов в Distributed Virtual Port Group (DVPG.macManagementPolicy=yes):

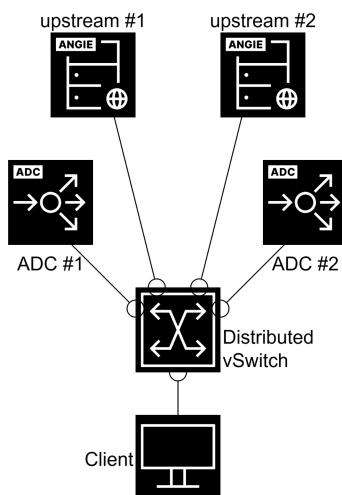


Рис. 4: подключение Angie ADC для ESXi версии 6.7+.

6.3.2 Режим резервирования (Active-Standby)

В этом разделе описана настройка высокой доступности с использованием протокола VRRP в режиме резервирования (Active-Standby). Для настройки используется *интерфейс командной строки (CLI) Angie ADC*.

Также можно посмотреть следующие статьи:

- *Настройка режима распределения нагрузки (Active-Active) с помощью протокола VRRP;*
- *Особенности создания и удаления VRRP-групп;*

- Команды VRRP.

6.3.2.1 Введение

Для обеспечения отказоустойчивости нужны две системы балансировки Angie ADC — основная и резервная. Резервная система Angie ADC будет принимать нагрузку при отказе основной. Автоматическое переключение между двумя системами балансировки Angie ADC можно реализовать с помощью протокола VRRP. Для настройки VRRP используется *командная строка Angie ADC CLI*. Пример настройки приведен ниже.

Примечание

Резервирование других сетевых компонентов (маршрутизаторы, коммутаторы, межсетевые экраны) выходит за рамки настоящей статьи.

Пример типового участка сети с системами балансировки Angie ADC (ADC #1 и ADC #2):

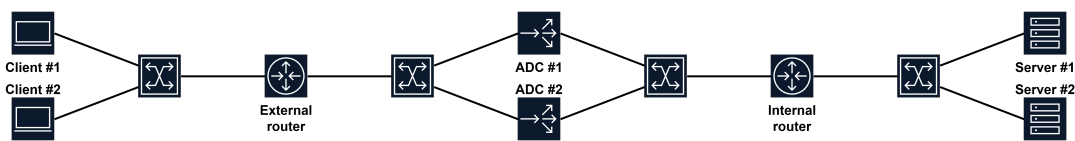


Рис. 1

Добавим IP-адресацию:

- В сегменте сети, куда подключаются External router, ADC #1 и ADC #2, будет использоваться подсеть 192.168.0.0/24.
- В сегменте сети, куда подключаются Internal router, ADC #1 и ADC #2, будет использоваться подсеть 192.168.1.0/24.
- Четвертый октет адреса для каждого устройства (в рамках соответствующей подсети) показан на схеме ниже.

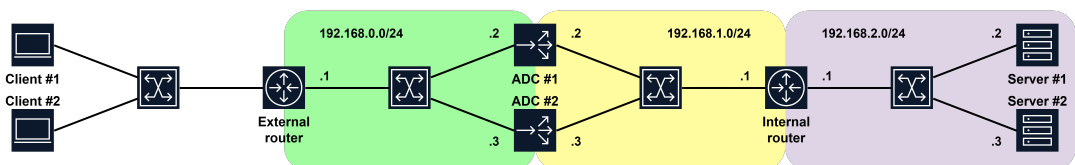


Рис. 2

Адресация клиентов не приводится, так как в данном случае мы будем считать, что их адреса формально не определены, то есть, они могут подключаться из любой сети, включая сеть интернет. Для этого на ADC #1 и ADC #2 необходимо указать адрес правого интерфейса маршрутизатора External router (192.168.0.1) в качестве пшюза по умолчанию.

Виртуальный адрес, который соответствует сервису, может быть размещен на loopback-интерфейсе каждой системы балансировки, либо это может быть виртуальный адрес, используемый протоколом VRRP. Первый способ будет более гибким.

При размещении адреса сервиса на loopback-интерфейсе (на схеме ниже это адрес 2.2.2.2) необходимо добавить статический маршрут на этот адрес на маршрутизаторе External router. Причем в качестве next-hop необходимо использовать виртуальный адрес VRRP, за счет чего достигается отказоустойчивость решения.

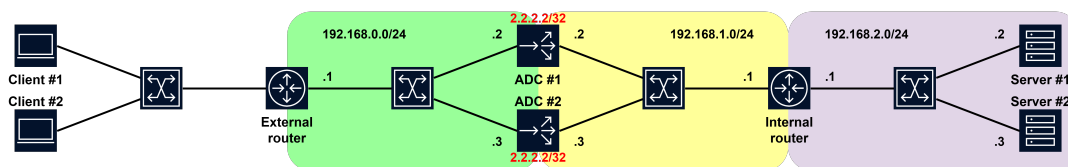


Рис. 3

При использовании виртуального адреса VRRP в качестве адреса сервиса (на схеме ниже это 192.168.0.10) настройка соответствующих статических маршрутов на маршрутизаторе External router не требуется. Виртуальный адрес VRRP находится в одной подсети с интерфейсами маршрутизатора с точки зрения маршрутизатора External router.

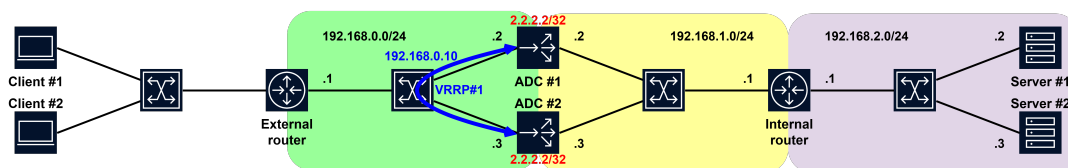


Рис. 4

Также допускается использование VRRP и при работе системы балансировки Angie ADC в «одно-руком» режиме.

6.3.2.2 Обработка типовых сбоев

Ниже описаны сценарии работы Angie ADC при сбоях.

Полный отказ ADC #1

При полном отказе или отключении от сети ADC #1 происходят пере выборы VRRP-мастера. Переключение на резервную систему ADC #2 производится автоматически. Никакие дополнительные настройки не требуются.

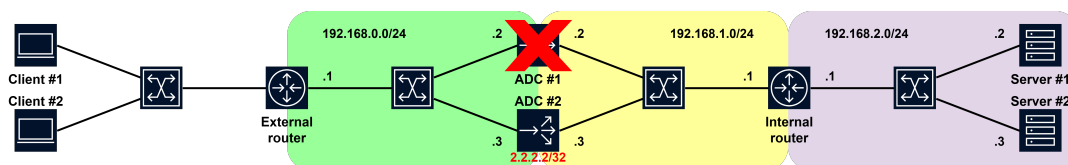


Рис. 5

Отказ внешнего интерфейса ADC #1

Предлагаемая схема не предполагает возможности отслеживания частичных отказов и реагирования на них (пример ниже).

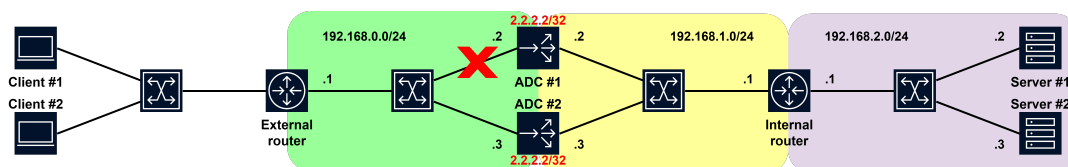


Рис. 6

В случае частичного отказа, как на схеме выше, ADC #1 перестает рассылать сообщения VRRP. ADC #2 начинает считать себя мастером, что заставляет виртуальную машину отвечать на ARP-запросы об адресе 192.168.0.10, а также обрабатывать пользовательский трафик. Обратный трафик (от серверов в сторону клиентов) будет проходить через ADC #2, так как ADC #2 устанавливает сессии до серверов с адреса своего внутреннего интерфейса.

Примечание

IP transparency на данном этапе обеспечить невозможно. Для работы IP transparency требуется наличие маршрута по умолчанию на маршрутизаторе Internal router. Если прописать маршрут по умолчанию через ADC #1, то отказ внешнего интерфейса ADC #1 приведет к невозможности доставки трафика и его отбрасыванию (black hole). Решить эту проблему можно за счет использования протокола VRRP как на внешнем, так и на внутреннем интерфейсах. Причем состояние групп должно быть согласованным, то есть если для внешней сети мастером является система балансировки ADC #2, то и для внутренней сети ADC #2 также должна являться мастером. Добиться этого можно с помощью опции VRRP tracking, которая будет выпущена в ближайших релизах Angie ADC.

Отказ линка между ADC #1 и Internal router

Отказ линка между ADC #1 и маршрутизатором Internal router более сложная ситуация: если ADC #1 сохраняет за собой роль мастера, то трафик от клиентов до серверов будет все еще передаваться через ADC #1, хотя реальные сервера для системы балансировки ADC #1 уже недостижимы. Таким образом, мы получаем классический пример black hole:

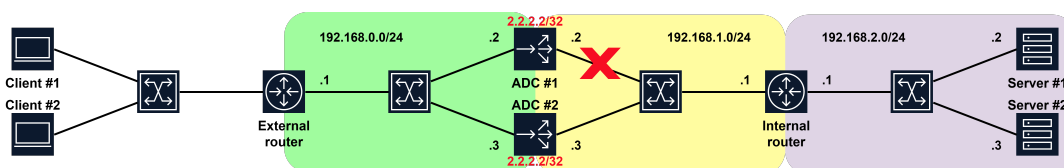


Рис. 7

Чтобы избежать этой проблемы, необходимо, чтобы ситуация отказа только одного из линков была невозможной, т.е. нужно обеспечить отказоустойчивое подключение к сети для хоста виртуализации, на котором работает ADC.

Рекомендованная схема подключения:

- Объединить в одну LAG-группу все физические интерфейсы хоста виртуализации и использовать этот виртуальный LAG-интерфейс в качестве транка, то есть передавать по нему агрегированными несколько виртуальных сетей (один VLAN для external сети, второй VLAN для internal сети).
- Для резервирования коммутаторов можно использовать технологию vPC или стекирование, пример представлен на схеме ниже.

Из соображений простоты на схеме не отображено резервирование внешнего и внутреннего маршрутизаторов.

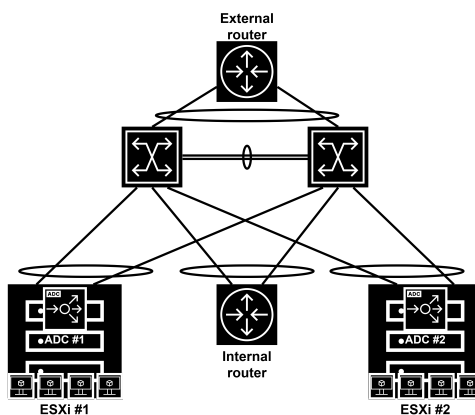


Рис. 8

Примечание

В ближайших релизах Angie ADC появится опция VRRP tracking, которая позволит автоматически переводить систему балансировки ADC #2 в режим VRRP backup при возникновении обсуждаемого отказа.

6.3.2.3 Настройка VRRP

Базовая настройка

Для настройки VRRP необходимо указать виртуальный IP-адрес, все остальные настройки опциональны:

```
#! Задать виртуальный IP-адрес 192.168.0.10/24 для первой VRRP группы внешнего
↪ интерфейсе (ens33).
#! Задается для ADC #1 и для ADC #2.
# vrrp 1 int ens33 ip 192.168.0.10/24
```

Указанная выше настройка одинакова для ADC #1 и ADC #2.

Примечание

Полный список поддерживаемых команд VRRP можно посмотреть в *справочнике по командам VRRP*.

Изменение приоритета

Если требуется, чтобы система балансировки ADC #1 выигрывала выборы и принимала на себя роль мастера, то ее приоритет необходимо сделать выше, чем приоритет у ADC #2. Значение приоритета по умолчанию – 100. Пример настройки ADC #1 представлен ниже:

```
#! Задается для ADC #1
#vrrp 1 int ens33 priority 110
```

Аналогичная настройка делается для ADC #2, но с меньшим значением приоритета:

```
#! Задается для ADC #2
#vrrp 1 int ens33 priority 90
```

Использование preempt

Опция VRRP **preempt** включена по умолчанию. Эта опция заставляет включившуюся систему балансировки инициировать пере выборы мастера, если ее собственный приоритет выше приоритета текущего мастера.

Включить **preempt** можно с помощью следующей команды:

```
vrrp 1 int ens33 preempt
```

Выключение **preempt** реализовано через **no**:

```
no vrrp 1 int ens33 preempt
```

Прописывание маршрута

Для указания статического маршрута, например на внешнем маршрутизаторе с указанием виртуального адреса VRRP в качестве next-hop, используется стандартная команда:

```
External(config)#ip route 2.2.2.2 255.255.255 192.168.0.10
```

6.3.2.4 Проверка работы

Убедимся, что ADC #1 является мастером:

```
# show vrrp int ens33 1 sum
Interface VRID Priority IPv4 IPv6 State (v4) State (v6)
-----
ens33 1 110 1 0 Master Backup
```

Убедимся, что ADC #2 не обладает ролью мастера:

```
# show vrrp int ens33 1 sum
Interface VRID Priority IPv4 IPv6 State (v4) State (v6)
-----
ens33 1 90 1 0 Backup Backup
```

Отключим теперь систему балансировки ADC #1 и убедимся, что ADC #2 приняла на себя роль мастера:

```
# show vrrp int ens33 1 sum
Interface VRID Priority IPv4 IPv6 State (v4) State (v6)
-----
ens33 1 90 1 0 Master Backup
```

Примечание

В момент переезда роли Master на вторую систему балансировки ADC возможны кратковременные перерывы обработки трафика, вызванные задержками обнаружения падения активной ноды. Характерное время переключения ~3 секунды. Ускорить процедуру переключения можно за счет редактирования параметра **advertisement-interval** (см. [справочник по командам VRRP](#)). В будущем планируется использование протокола BFD для ускорения процедуры обнаружения отказов.

Заново включим систему балансировки ADC #1. Поведение устройства зависит от состояния опции VRRP **preempt**. Если опция **preempt** активирована (значение по умолчанию), то ADC #1 инициирует пере выборы и заберет на себя роль мастера. Если опция **preempt** отключена, то ADC #1 не будет пытаться отобрать роль мастера у системы балансировки ADC #2.

Проверить текущее состояние опции **preempt** можно с помощью следующей команды:

```
# show vrrp int ens33 1
Virtual Router ID 1
Protocol Version 3
Autoconfigured No
Shutdown No
Interface ens33
VRRP interface (v4) vrrp4-ens33-1
VRRP interface (v6) None
Primary IP (v4) 192.168.0.3
Primary IP (v6) ::
Virtual MAC (v4) 00:00:5e:00:01:01
Virtual MAC (v6) 00:00:5e:00:02:01
Status (v4) Master
Status (v6) Initialize
Priority 110
Effective Priority (v4) 110
Effective Priority (v6) 110
Preempt Mode No
Accept Mode Yes
Checksum with IPv4 Pseudoheader Yes
Advertisement Interval 1000 ms
Master Advertisement Interval (v4) Rx 1000 ms (stale)
Master Advertisement Interval (v6) Rx 0 ms (stale)
Advertisements Tx (v4) 1163
Advertisements Tx (v6) 0
Advertisements Rx (v4) 0
Advertisements Rx (v6) 0
Gratuitous ARP Tx (v4) 1
Neigh. Adverts Tx (v6) 0
State transitions (v4) 2
State transitions (v6) 0
Skew Time (v4) 600 ms
Skew Time (v6) 0 ms
Master Down Interval (v4) 3600 ms
Master Down Interval (v6) 0 ms
IPv4 Addresses 1
..... 192.168.0.10
IPv6 Addresses 0
```

Вне зависимости от того, какая система балансировки активна в данный момент, на маршрутизаторах в ARP-записи об адресе 192.168.0.10 будет один и тот же MAC-адрес: 00-00-5e-00-01-01 (при использовании первой VRRP-группы).

```
External#sho ip arp
Protocol Address Age (min) Hardware Addr Type Interface
Internet 192.168.0.10 - 0000.5e00.0101 ARPA GigabitEthernet0/0
```

Это означает, что при отказе одной из систем балансировки и аварийном переключении на вторую, единственное изменение, которое должно произойти в сети – переобучение коммутаторов, то есть потребуется актуализировать таблицы коммутации для адреса 00-00-5e-00-01-01.

6.3.2.5 Заключение

С помощью протокола VRRP можно построить отказоустойчивое решение по балансировке нагрузки на сервера там, где использование протоколов динамической маршрутизации невозможно по той или иной причине.

Ряд дополнительных опций, расширяющих возможности протокола VRRP, находятся в данный момент в стадии разработки и будут доступны в ближайших релизах Angie ADC.

6.3.3 Режим распределения нагрузки (Active-Active)

В этом разделе описана настройка высокой доступности с использованием протокола VRRP в режиме распределения нагрузки (Active-Active). Для настройки используется *интерфейс командной строки (CLI) Angie ADC*.

Также можно посмотреть следующие статьи:

- *Настройка режима резервирования (Active-Standby) с помощью протокола VRRP;*
- *Особенности создания и удаления VRRP-групп;*
- *Команды VRRP.*

6.3.3.1 Введение

Реализация полноценной схемы распределения нагрузки (Active-Active) не предусмотрена протоколом VRRP, но это ограничение можно обойти. Для этого рассмотрим два сценария подключения Angie ADC:

- между клиентами и системами балансировки установлен хотя бы один маршрутизатор;
- между клиентами и системами балансировки нет маршрутизатора.

6.3.3.2 Подключение Angie ADC с маршрутизатором

Пример сети с маршрутизатором между клиентами и системами балансировки Angie ADC (ADC #1 и ADC #2):

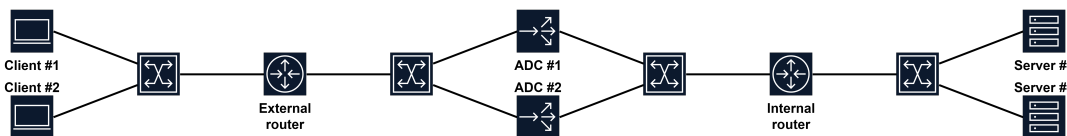


Рис. 1

Добавим IP-адресацию:

- В сегменте сети, куда подключаются External router, ADC #1 и ADC #2, будет использоваться подсеть 192.168.0.0/24.
- В сегменте сети, куда подключаются Internal router, ADC #1 и ADC #2, будет использоваться подсеть 192.168.1.0/24.
- Четвертый октет адреса для каждого устройства (в рамках соответствующей подсети) показан на схеме ниже.

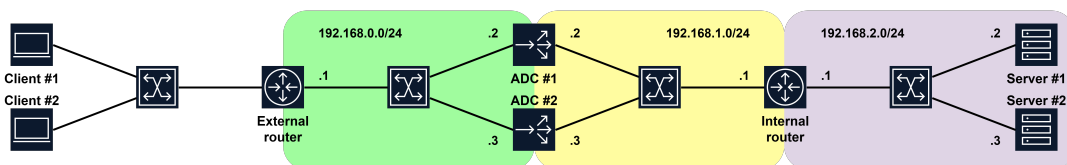


Рис. 2

Адресация клиентов не приводится, так как в данном случае мы будем считать, что их адреса формально не определены, то есть, они могут подключаться из любой сети, включая сеть интернет. Для этого на ADC #1 и ADC #2 необходимо указать адрес правого интерфейса маршрутизатора External router (192.168.0.1) в качестве шлюза по умолчанию.

Виртуальный адрес, который соответствует сервису, может быть размещен на loopback-интерфейсе каждой системы балансировки, либо это может быть виртуальный адрес, используемый протоколом VRRP.

- Первый способ (рис. 3 – сервис размещен на loopback-интерфейсах и IP-адреса этих интерфейсов совпадают) будет более гибким, так как позволяет использовать только один IP-адрес для сервиса.
- Во втором случае (используется виртуальный адрес протокола VRRP) необходимо сообщить клиенту, что сервис работает на двух адресах (например, с помощью DNS).
- В третьем случае, если сервис размещен на loopback-интерфейсах, но IP-адреса этих интерфейсов не совпадают, также потребуется сообщить клиенту, что сервис работает на двух адресах.

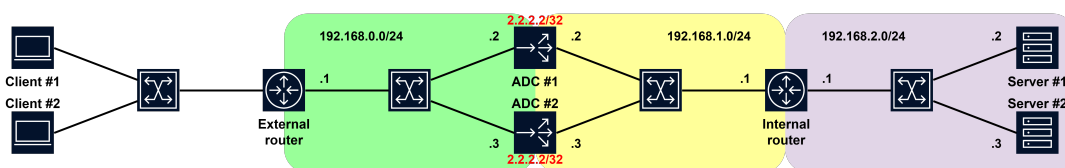


Рис. 3

При размещении адреса сервиса на loopback-интерфейсе (на схеме ниже это адрес 2.2.2.2) необходимо добавить статический маршрут на этот адрес на маршрутизаторе External router. Причем в качестве next-hop необходимо использовать виртуальный адрес VRRP – 192.168.0.10, за счет чего достигается отказоустойчивость решения.

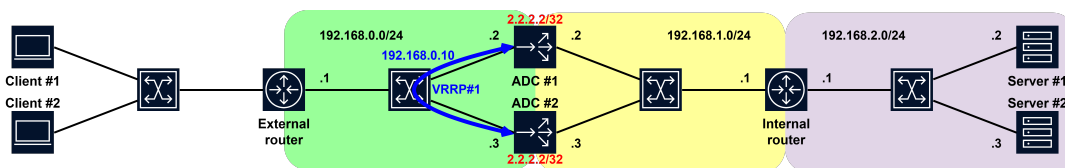


Рис. 4

Настройка режима Active-Active

Чтобы перейти от режима работы Active-Standby к режиму Active-Active, необходимо на том же интерфейсе систем балансировки настроить вторую VRRP группу таким образом, чтобы мастером в ней являлась другая система Angie ADC (достигается за счет опции priority). Для второй группы должен быть настроен виртуальный IP-адрес, отличающийся от адреса первой группы.

На схеме ниже вторая группа имеет виртуальный IP-адрес 192.168.0.11:

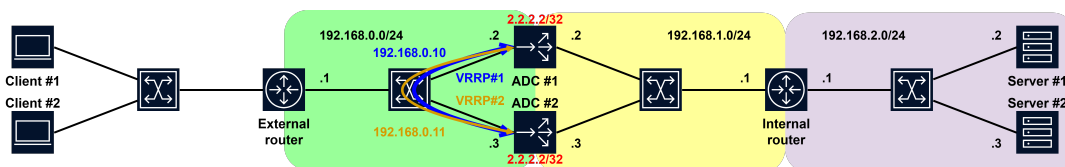


Рис. 5

На стороне маршрутизатора External router должно быть прописано два статических маршрута до префикса 2.2.2.2/32:

- один с next-hop 192.168.0.10;
- второй – 192.168.0.11.

Маршрутизатор External router должен выполнять ECMP-балансировку трафика (часть сессий отправляется в сторону ADC #1, а оставшиеся – в сторону ADC #2).

Ниже приведен пример настройки маршрутизатора Cisco:

```
External(config)#ip route 2.2.2.2 255.255.255.255 192.168.0.10
External(config)#ip route 2.2.2.2 255.255.255.255 192.168.0.11
External(config)#exi
External#sho ip ro 2.2.2.2
Routing entry for 2.2.2.2/32
Known via "static", distance 1, metric 0
Routing Descriptor Blocks:
192.168.0.11
Route metric is 0, traffic share count is 1
* 192.168.0.10
Route metric is 0, traffic share count is 1
External#
```

У VRRP-группы есть виртуальный MAC-адрес (00:00:5e:00:01:n, где n – номер группы VRRP, номер виртуального маршрутизатора, в шестнадцатеричном виде). По MAC-адресу коммутатор определяет, для какой из систем балансировки предназначен трафик. Первая группа будет иметь виртуальный MAC-адрес 00:00:5e:00:01:01, вторая – 00:00:5e:00:01:02.

Убедиться в этом можно, просмотрев ARP-таблицу на маршрутизаторе External router:

```
External#sho ip arp
Protocol Address Age (min) Hardware Addr Type Interface
Internet 192.168.0.1 - ca01.0950.0008 ARPA GigabitEthernet0/0
Internet 192.168.0.10 0 0000.5e00.0101 ARPA GigabitEthernet0/0
Internet 192.168.0.11 0 0000.5e00.0102 ARPA GigabitEthernet0/0
External#
```

ARP-таблица маршрутизатора External router не меняется при отказе одной из систем балансировки нагрузки Angie ADC. Изменения будут на стороне коммутатора. Ниже приводится часть таблицы коммутации, когда обе системы балансировки «живы» и выполняют свои функции:

```
switch#sho mac address-table dynamic vlan 51
Mac Address Table
-----
Vlan Mac Address Type Ports
-----
51 0000.5e00.0101 DYNAMIC Gi0/7
51 0000.5e00.0102 DYNAMIC Gi0/8
```

Система балансировки ADC #1 подключена к интерфейсу Gi0/7 коммутатора, а система балансировки ADC #2 – к интерфейсу Gi0/8.

При отказе одной из систем балансировки оставшаяся будет иметь обе VRRP-группы в статусе Master. С точки зрения коммутатора это будет выглядеть так, словно оба виртуальных MAC-адреса доступны через один интерфейс.

```
switch#sho mac address-table dynamic vlan 51
Mac Address Table
-----
Vlan Mac Address Type Ports
-----
51 0000.5e00.0101 DYNAMIC Gi0/7
51 0000.5e00.0102 DYNAMIC Gi0/7
```

6.3.3.3 Подключение Angie ADC без маршрутизатора

Пример схемы сети представлен на рисунке ниже.

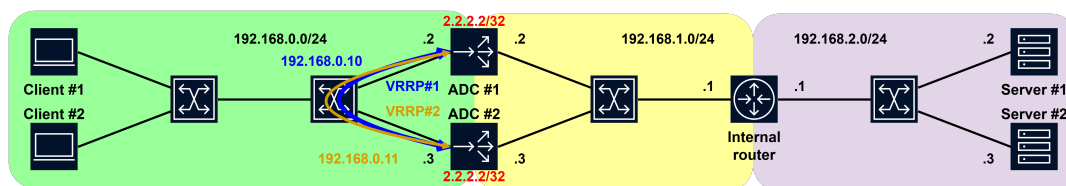


Рис. 6

В данном случае клиенты должны находиться в той же подсети, что и внешние интерфейсы систем балансировки Angie ADC (192.168.0.0/24).

Если клиенты Client #1 и Client #2 не имеют выхода в интернет, то тогда на них будет достаточно настроить один из адресов 192.168.0.10 и 192.168.0.11 в качестве шлюза по умолчанию. Тогда половина клиентов будет использовать адрес 192.168.0.10 в качестве шлюза по умолчанию, а вторая половина – 192.168.0.11.

Если для выхода в интернет используется какое-то отдельное устройство, то на клиентах придется прописать статический маршрут до префикса 2.2.2.2/32. На половине клиентов в качестве адреса следующего перехода (next-hop) необходимо настроить виртуальный адрес первой группы – 192.168.0.10, а на оставшихся клиентах – адрес второй группы, 192.168.0.11.

На схеме ниже желтой и красной стрелками указаны направления на шлюзы по умолчанию.

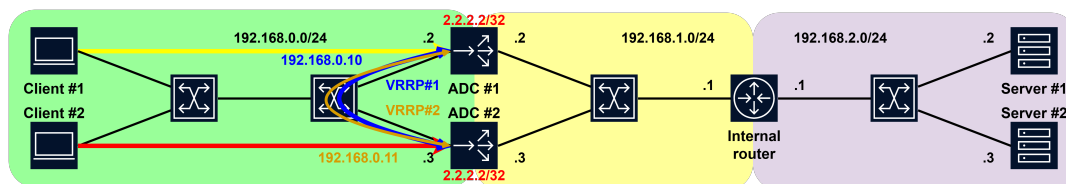


Рис. 7

Использовать ECMP-подход здесь нецелесообразно, так как некоторые клиентские операционные системы могут не поддерживать такую балансировку.

6.3.3.4 Настройка VRRP

Настройка VRRP на системах балансировки не зависит от того, какая схема подключения используется, с маршрутизатором или без.

Для настройки VRRP необходимо:

- на каждой системе балансировки создать две группы;
- для каждой из групп указать виртуальный IP-адрес и приоритет.

Все остальные настройки опциональны.

Примечание

Полный список поддерживаемых команд VRRP можно посмотреть в *справочнике по командам VRRP*.

Пример настройки VRRP для ADC #1:

```
#! Задать виртуальный IP-адрес 192.168.0.10/24 для первой VRRP группы внешнего
↪интерфейсе (ens33).
# vrrp 1 int ens33 ip 192.168.0.10/24
#! Задать приоритет 110 для первой VRRP группы внешнего интерфейсе (ens33).
# vrrp 1 int ens33 priority 110
#! Задать виртуальный IP-адрес 192.168.0.11/24 для второй VRRP группы внешнего
↪интерфейсе (ens33).
# vrrp 2 int ens33 ip 192.168.0.11/24
#! Задать приоритет 90 для второй VRRP группы внешнего интерфейсе (ens33).
# vrrp 2 int ens33 priority 90
```

Настройка VRRP для ADC #2 симметричная:

```
#! Задать виртуальный IP-адрес 192.168.0.10/24 для первой VRRP группы внешнего
↪интерфейсе (ens33).
# vrrp 1 int ens33 ip 192.168.0.10/24
#! Задать приоритет 90 для первой VRRP группы внешнего интерфейсе (ens33).
# vrrp 1 int ens33 priority 90
#! Задать виртуальный IP-адрес 192.168.0.11/24 для второй VRRP группы внешнего
↪интерфейсе (ens33).
# vrrp 2 int ens33 ip 192.168.0.11/24
#! Задать приоритет 110 для второй VRRP группы внешнего интерфейсе (ens33).
# vrrp 2 int ens33 priority 110
```

6.3.3.5 Проверка работы

Убедимся, что в стабильном и полностью рабочем состоянии ADC #1 является мастером для первой группы VRRP, а для второй группы мастером является система ADC #2.

Вывод с системы балансировки ADC #1:

```
# show vrrp int ens33 sum
Interface VRID Priority IPv4 IPv6 State (v4) State (v6)
-----
ens33 1 110 1 0 Master Backup
ens33 2 90 1 0 Backup Backup
```

Вывод с системы балансировки ADC #2:

```
# show vrrp int ens33 sum
Interface VRID Priority IPv4 IPv6 State (v4) State (v6)
-----
ens33 1 90 1 0 Backup Backup
ens33 2 110 1 0 Master Backup
```

При отказе, например, системы балансировки ADC #2, вывод информации о группах VRRP с системы балансировки ADC #1 будет выглядеть так:

```
# show vrrp int ens33 sum
Interface VRID Priority IPv4 IPv6 State (v4) State (v6)
-----
ens33 1 110 1 0 Master Backup
ens33 2 90 1 0 Master Backup
##
```


6.3.3.6 Заключение

С помощью протокола VRRP можно построить отказоустойчивое решение по балансировке нагрузки на сервера там, где использование протоколов динамической маршрутизации невозможно по той или иной причине. Стоит отдельно отметить, что поддерживается работа систем балансировки как в режиме Active-Standby, так и в режиме Active-Active.

Ряд дополнительных опций, расширяющих возможности протокола VRRP, находятся сейчас в разработке и будут доступны в ближайших релизах Angie ADC.

6.3.4 Команды VRRP

Здесь описываются команды VRRP, поддерживаемые в Angie ADC, и особенности их работы. Настройка VRRP осуществляется через *командную строку*.

Также вы можете посмотреть следующие статьи:

- *Настройка режима резервирования (Active-Standby) с помощью протокола VRRP;*
- *Настройка режима распределения нагрузки (Active-Active) с помощью протокола VRRP;*
- *Особенности создания и удаления VRRP-групп.*

Обобщенный пример команды VRRP:

```
vrrp n int ens33 command
```

Параметры:

- **n** - номер VRRP-группы (так же VRID — Virtual Router ID). Поддерживаются номера от 1 до 255.
- **ens33** - имя физического интерфейса. Если требуется создать VRRP-группы для другого физического интерфейса, то в команде необходимо указать имя этого интерфейса. VRRP можно независимо использовать для нескольких интерфейсов одновременно.
- **command** - команда VRRP (список поддерживаемых команд см. ниже).

6.3.4.1 Команды настройки

Команда	Описание	Значение по умолчанию
<code>vrrp n int <interface> ip <ip-address></code>	Команда добавляет IPv4-адрес для Angie ADC. Добавление IP-адреса автоматически активирует его. Задается в формате <code><ip_address>/<subnet_mask></code> . Примечание Маска подсети IP-адреса должна соответствовать маске физического интерфейса.	n/a
<code>vrrp n int <interface> version v</code>	Команда задает версию протокола VRRP (v равно 2 или 3). Если версия протокола не задана, по умолчанию используется VRRPv3. Примечание Версии протокола VRRP на двух Angie ADC, работающих вместе, должны совпадать, иначе оба Angie ADC VA будут считать себя мастером.	3
<code>vrrp n int <interface> advertisement-inte i</code>	Команда устанавливает значение интервала отправки анонсов для VRRP-группы n. Значение задается в миллисекундах, но должно быть кратно 10, так как VRRP использует единицы измерения в сотых долях секунды (i от 10 до 40950 в мс).	1000 мс (1 с)
<code>vrrp n int <interface> preempt</code>	Команда включает режим preempt , разрешающий запуск выборов нового мастера. При включенном режиме preempt резервный Angie ADC может автоматически стать мастером, если его приоритет выше, чем у текущего Angie ADC с ролью master .	Enabled (режим preempt включен)
<code>vrrp n int <interface> priority p</code>	Команда устанавливает приоритет p для устройств в VRRP-группе n на интерфейсе <interface> (p от 1 до 254). Значение приоритета 255 зарезервировано для текущего устройства в роли master , чей адрес интерфейса совпадает с виртуальным адресом группы. Значение 255 нельзя задать вручную. Устройство с наивысшим приоритетом становится master . Если у нескольких устройств в VRRP-группе одинаковый приоритет и режим preempt включен, мастером становится устройство с наибольшим основным IP-адресом.	100
<code>vrrp n int <interface> shutdown</code>	Команда выключает VRRP-группу n на интерфейсе <interface>. Используйте эту команду, если нужно завершить работу VRRP-группы без ее удаления. Чтобы обратно включить VRRP для этой группы, необходимо отменить текущую команду с помощью <code>no</code> (см. следующую таблицу).	Disabled (режим shutdown не активирован, группа включена)
<code>vrrp n int <interface> checksum-with-ipv4</code>	Команда включает использование псевдо-заголовка IPv4 при расчете контрольной суммы VRRPv3 для VRRP-группы n. Эта команда не влияет на VRRPv2 и IPv6.	Enabled (использование псевдо-заголовков IPv4 включено)

6.3.4.2 Команды удаления и отмены

Команда	Описание
<code>no vrrp n int <interface> ip <ip-address></code>	Команда удаляет IP-адрес, настроенный для VRRP-группы n .
<code>no vrrp n int <interface></code>	Команда удаляет все пользовательские настройки, относящиеся к VRRP-группе n для интерфейса <interface> . Выставляются значения по умолчанию.
i Примечание Сама группа следом удаляется.	
<code>no vrrp n int <interface> priority p</code>	Команда удаляет пользовательскую настройку приоритета для VRRP-группы n .
<code>no vrrp n int <interface> version v</code>	Команда удаляет версию протокола VRRP, настроенную для VRRP-группы n . Выставляется значение по умолчанию.
<code>no vrrp n int <interface> advertisement-interval i</code>	Команда удаляет значение интервала отправки анонсов, настроенное для VRRP-группы n . Выставляется значение по умолчанию.
<code>no vrrp n int <interface> shutdown</code>	Команда включает ранее выключенную VRRP-группу n на интерфейсе <interface> .
i Примечание После перезагрузки Angie ADC при включенном режиме vrrp default shutdown все VRRP-группы переводятся в выключенное состояние, в том числе те, которые были включены вручную. Чтобы заново включить конкретную группу после перезагрузки, введите последовательно следующие две команды: <code>vrrp n int <interface> shutdown</code> <code>no vrrp n int <interface> shutdown</code>	
<code>no vrrp n int <interface> preempt</code>	Команда выключает режим preempt для группы n , выборы не будут запускаться даже при более высоком приоритете группы n .
<code>no vrrp n int <interface> checksum-with-ipv4-pseudo</code>	Команда выключает использование псевдо-заголовка IPv4 при расчете контрольной суммы VRRPv3 для VRRP-группы n .

6.3.4.3 Дефолтные команды для настройки и отмены

Команда	Описание	Значение по умолчанию
<code>vrrp default advertisement-inte</code>	Команда устанавливает значение интервала отправки анонсов для всех новых VRRP-групп при их создании. Значение задается в миллисекундах, но должно быть кратно 10, так как VRRP использует единицы измерения в сотых долях секунды (i от 10 до 40950 в мс).	1000 мс (1 с)
<code>vrrp default shutdown</code>	Команда устанавливает значение "выключено" для всех новых VRRP-групп при их создании, то есть переводит их в выключенное состояние сразу после загрузки или применения конфигурации. Режим shutdown можно выключить глобально (<code>no vrrp default shutdown</code>).	Enabled (режим default shutdown включен, новые группы выключены)
 Примечание После перезагрузки Angie ADC при включенном режиме <code>vrrp default shutdown</code> все VRRP-группы переводятся в выключенное состояние, в том числе те, которые были включены вручную.		
<code>vrrp default checksum-with-ipv4</code>	Команда включает использование псевдо-заголовка IPv4 при расчете контрольной суммы VRRPv3 для всех новых VRRP-групп при их создании. Эта команда не влияет на VRRPv2 и IPv6.	Enabled (дефолтное использование включено)
<code>no vrrp default advertisement-inte</code>	Команда удаляет пользовательское значение интервала отправки анонсов, настроенное для всех новых групп VRRP. Выставляется значение по умолчанию 1000 мс (1 сек.).	n/a
<code>no vrrp default shutdown</code>	Команда отключает настройку, по которой все новые VRRP-группы создаются выключенными. Новые VRRP-группы будут создаваться сразу включенными.	n/a
<code>no vrrp default checksum-with-ipv4</code>	Команда отключает настройку, по которой для всех новых VRRP-групп при расчете контрольной суммы используется псевдо-заголовок IPv4.	n/a

6.3.4.4 Команды для просмотра информации

Команда	Описание
<code>show vrrp</code>	Команда выводит информацию о состоянии всех VRRP-групп для всех интерфейсов.
<code>show vrrp int <interface></code>	Команда выводит информацию о всех VRRP-группах для интерфейса <interface>.
<code>show vrrp int <interface> n</code> или <code>show vrrp n int <interface></code>	Команда выводит информацию о VRRP-группе n для интерфейса <interface> (n от 1 до 255).
<code>show vrrp int <interface> n sum</code>	Команда выводит краткую информацию о VRRP-группе n для интерфейса <interface> (n от 1 до 255).

6.3.5 Создание и удаление VRRP-групп

Здесь описываются особенности создания и удаления VRRP-групп. Настройка VRRP осуществляется через *интерфейс командной строки (CLI) Angie ADC*.

Также можно посмотреть следующие статьи:

- *Настройка режима резервирования (Active-Standby) с помощью протокола VRRP;*
- *Настройка режима распределения нагрузки (Active-Active) с помощью протокола VRRP;*
- *Справочник по VRRP-командам.*

VRRP-группа — это логическая сущность, привязанная к физическому интерфейсу или его подынтерфейсу. Создание и удаление VRRP-групп вручную в Angie ADC не требуется. VRRP-группы создаются автоматически при введении команд, изменяющих конфигурацию по умолчанию, и удаляются также автоматически, если для группы не остается параметров, значения которых отличаются от значений по умолчанию. Это позволяет сохранять минимальную конфигурацию VRRP по умолчанию.

Вы можете просмотреть список всех VRRP-групп с помощью команды **show vrrp**.

Ниже приведены примеры ситуаций, в которых VRRP-группа будет создана или удалена.

6.3.5.1 Создание VRRP-группы

Новая VRRP-группа создается автоматически, если изменяется значение по умолчанию любого из ее параметров.

Например, если для VRRP-группы 10, привязанной к интерфейсу **ens33**, указать IP-адрес (по умолчанию не указан), то указанная группа будет создана автоматически:

```
vrrp 10 int ens33 ip 192.168.0.1/24
```

Если для VRRP-группы 11, привязанной к интерфейсу **ens33**, указать значение параметра **preempt** равным **disabled** (по умолчанию **enabled**), то указанная группа также будет создана автоматически:

```
no vrrp 11 int ens33 preempt
```

6.3.5.2 Удаление VRRP-группы

Рассмотрим пример. Для интерфейса **ens33** создано две VRRP-группы, с параметром **preempt** со значением **disabled** (выключено):

```
no vrrp 10 int ens33 preempt
```

```
no vrrp 11 int ens33 preempt
```

Группа удаляется в следующих случаях:

- Администратор ввел команду, возвращающую все параметры группы к значениям по умолчанию, в результате чего группа удаляется. Пример удаления группы 10:

```
no vrrp 10 int ens33
```

- Администратор вернул параметр группы к значению по умолчанию, при этом другие параметры группы уже имеют значения по умолчанию. Группа будет автоматически удалена, так как ее настройка больше не содержит ни одного параметра, значение которого отличается от значения по умолчанию.

Пример удаления группы 11 (параметру **preempt** присваивается значение по умолчанию **enabled**):

```
vrrp 11 int ens33 preempt
```

6.4 Использование протокола BFD для уменьшения времени реакции

6.4.1 Введение

Стандартные таймеры протоколов маршрутизации слишком велики и не подходят для современных сетей. Уменьшить время реакции протокола на какое-либо событие (непрямое падение, indirect failure) можно двумя способами:

- за счет уменьшения значения таймеров (может привести к увеличению нагрузки);
- с помощью специализированного протокола — BFD (предпочтительный способ).

BFD (Bidirectional Forwarding Detection) — это легковесный протокол, используемый для быстрого обнаружения отказов сетевых каналов между двумя устройствами. Он работает независимо и комплементарно к протоколам маршрутизации и помогает быстро определить, что связь между узлами нарушена. Если в инфраструктуре важна быстрая реакция на сбой, BFD значительно уменьшает время сходимости сети, так как позволяет получить субсекундное время реакции.

Как работает BFD?

1. Установление сессии: два сетевых устройства обмениваются контрольными пакетами BFD для установления связи.
2. Мониторинг: оба узла отправляют друг другу небольшие пакеты с фиксированным интервалом.
3. Обнаружение отказа: если один из узлов перестает получать пакеты от соседа в течение заданного времени, соединение объявляется "down".
4. Реакция: протоколы маршрутизации отслеживают состояние BFD-сессий и разрывают свои соединения, что позволяет переключиться на резервный канал менее, чем за секунду после сбоя.

Примечание

Для настройки используется *командная строка Angie ADC CLI*.

6.4.2 Настройка с OSPF

Для протокола OSPF поддержка BFD активируется на интерфейсе с помощью команды `ip ospf bfd`.

Пример конфигурации интерфейса:

```
interface ens33
ip address 192.168.0.2/24
ip ospf 1 area 0
ip ospf bfd
ip ospf bfd profile test
```

Если параметры работы протокола BFD на каком-либо интерфейсе должны отличаться от стандартных, то новые параметры можно указать путем привязки профиля.

Протокол OSPF осуществляет динамическое обнаружение соседей на интерфейсе, а затем протокол BFD пытается установить с ними соседство:

```

angie-va# sho bfd peers
BFD Peers:
peer 192.168.0.3 vrf default interface ens33
ID: 192599120
Remote ID: 626157519
Active mode
Status: up
Uptime: 4 second(s)
Diagnostics: ok
Remote diagnostics: ok
Peer Type: dynamic
RTT min/avg/max: 0/0/0 usec
Local timers:
Detect-multiplier: 3
Receive interval: 300ms
Transmission interval: 300ms
Echo receive interval: 50ms
Echo transmission interval: disabled
Remote timers:
Detect-multiplier: 3
Receive interval: 300ms
Transmission interval: 300ms
Echo receive interval: 50ms

```

Протокол BFD может установить сессию с каким-либо соседом через определенный интерфейс и одновременно с этим не установить сессию с другим соседом. Это ожидаемое поведение для соседей, не обладающих поддержкой BFD. OSPF-соединение при этом не разрывается.

Если протокол BFD смог установить сессию с каким-либо из OSPF-соседей, то доступность этого соседа будет определяться с помощью протокола BFD.

Если протокол BFD не смог установить сессию с каким-либо из OSPF-соседей, то доступность этого соседа будет определяться с помощью встроенных механизмов OSPF (протокол Hello). При этом команда `sho bfd peers` будет отображать такого соседа в статусе `down`:

```

angie-va# sho bfd peers
BFD Peers:
peer 192.168.0.3 vrf default interface ens33
ID: 192599120
Remote ID: 0
Active mode
Status: down
Downtime: 4 minute(s), 0 second(s)
Diagnostics: ok
Remote diagnostics: ok
Peer Type: dynamic
RTT min/avg/max: 0/0/0 usec
Local timers:
Detect-multiplier: 3
Receive interval: 300ms
Transmission interval: 300ms
Echo receive interval: 50ms
Echo transmission interval: disabled
Remote timers:
Detect-multiplier: 3
Receive interval: 1000ms
Transmission interval: 1000ms
Echo receive interval: disabled

```


6.4.3 Настройка с BGP

По умолчанию при указании BGP-соседа BFD-сессия между маршрутизаторами не устанавливается.

Пример такой конфигурации:

```
angie-v# sho run
! Часть конфига, не относящаяся к BGP/BFD, удалена для краткости.
router bgp 3
no bgp ebgp-requires-policy
neighbor 192.168.0.2 remote-as 2
exit
!
angie-v# sho ip bg su
IPv4 Unicast Summary:
BGP router identifier 3.3.3.3, local AS number 3 VRF default vrf-id 0
BGP table version 0
RIB entries 0, using 0 bytes of memory
Peers 1, using 24 KiB of memory
Neighbor      V      AS  MsgRcvd  MsgSent  TblVer  InQ OutQ  Up/Down State/
↪PfxRcd  PfxSnt Desc
192.168.0.2    4        2         8         9         0    0    0 00:05:35
↪  0         0 N/A
! BGP сессия успешно установлена, хотя никаких BFD-соседей у системы нет.
Total number of neighbors 1

angie-v# sho bfd pe
BFD Peers:
angie-v#
```

Чтобы активировать установление BFD-сессии с BGP-соседом, установите в секции настройки маршрутизатора BGP опцию `bfd` для соседа:

```
router bgp 2
no bgp ebgp-requires-policy
neighbor 192.168.0.3 remote-as 3
neighbor 192.168.0.3 bfd
exit
```

Чтобы BFD-сессия поднялась, необходимо включить поддержку BFD в настройках процесса маршрутизации BGP с обеих сторон. Еще одним способом является создание BFD-соседа вручную.

BGP-сессия успешно устанавливается и обновления передаются даже в ситуации, когда BFD-сессия не была установлена. Однако, если BFD-сессия была установлена, а затем разорвана, то автоматически будет разорвана и соответствующая BGP-сессия (после чего будут предприняты попытки переустановить BGP-сессию).

```
angie-v# sho bfd peers
BFD Peers:
peer 192.168.0.3 local-address 192.168.0.2 vrf default interface ens33
ID: 2213541843
Remote ID: 0
Active mode
Status: down
Downtime: 5 second(s)
Diagnostics: neighbor signaled session down
Remote diagnostics: neighbor signaled session down
Peer Type: dynamic
RTT min/avg/max: 0/0/0 usec
```

```

Local timers:
Detect-multiplier: 3
Receive interval: 300ms
Transmission interval: 300ms
Echo receive interval: 50ms
Echo transmission interval: disabled
Remote timers:
Detect-multiplier: 3
Receive interval: 300ms
Transmission interval: 300ms
Echo receive interval: 50ms

angie-v# sho ip bg su

IPv4 Unicast Summary:
BGP router identifier 2.2.2.2, local AS number 2 VRF default vrf-id 0
BGP table version 0
RIB entries 0, using 0 bytes of memory
Peers 1, using 24 KiB of memory

Neighbor      V      AS  MsgRcvd  MsgSent  TblVer  InQ OutQ  Up/Down State/
↪PfxRcd  PfxSnt Desc
192.168.0.3    4      3       27      30       0    0    0 00:00:06
↪  0      0 N/A

Total number of neighbors 1

```

6.4.4 Настройка с VRRP

На данный момент нет интеграции с BFD.

6.4.5 Настройка со статическими маршрутами

Angie ADC поддерживает условное добавление статического маршрута (в зависимости от BFD-статуса соседа, который указан в качестве next-hop). То есть, если BFD-сессия до устройства, адрес которого указан как next-hop, поднимается, то статический маршрут добавляется в таблицу маршрутизации. Если сессия остается в статусе **down** (или падает в процессе эксплуатации позднее), маршрут не добавляется (удаляется) из таблицы маршрутизации.

Добавить статический маршрут с проверкой доступности next-hop по BFD можно, например, с помощью следующей команды:

```
ip route 3.3.3.3/32 192.168.0.3 bfd
```

Сразу после создания такого статического маршрута соответствующая BFD-сессия находится в статусе **down**:

```

angie-v# sho bfd pee
BFD Peers:
peer 192.168.0.3 vrf default
ID: 1753656330
Remote ID: 0
Active mode
Status: down
Downtime: 6 second(s)
Diagnostics: ok
Remote diagnostics: ok
Peer Type: dynamic

```

```
RTT min/avg/max: 0/0/0 usec
Local timers:
Detect-multiplier: 3
Receive interval: 300ms
Transmission interval: 300ms
Echo receive interval: 50ms
Echo transmission interval: disabled
Remote timers:
Detect-multiplier: 3
Receive interval: 1000ms
Transmission interval: 1000ms
Echo receive interval: disabled
```

Чтобы сессия поднялась, на соседе нужно либо настроить собственный статический маршрут с указанием адреса исходящего интерфейса в качестве next-hop, либо вручную создать статический пир, соответствующий системе балансировки (см. ниже).

6.4.6 Настройка независимого пира

Во всех приведенных выше случаях пир создавался динамически по требованию протокола динамической маршрутизации или в момент создания соответствующего статического маршрута. Ручное создание BFD-пира может потребоваться, например, в ситуации, когда необходимо поменять параметры работы какого-то одного соседа из группы автоматически обнаруженных.

Переход к созданию и настройке отдельного пира из режима конфигурирования протокола BFD:

```
angie-va# conf t
angie-va(config)# bfd
angie-va(config-bfd)# peer
A.B.C.D IPv4 peer address
X:X::X:X IPv6 peer address
angie-va(config-bfd)# peer
```

Некоторые настройки необходимо задать в момент создания пира (интерфейс, локальный адрес, multihop и vrf), остальные параметры можно будет изменить потом.

```
angie-va(config-bfd)# peer 192.168.0.3
<cr>
interface      Interface information
local-address   Configure local address
multihop        Configure multihop
vrf             Configure VRF
angie-va(config-bfd)# peer 192.168.0.3
angie-va(config-bfd-peer)#
detect-multiplier  Configure peer detection multiplier
echo               Configure peer echo intervals
echo-interval      Configure peer echo intervals
echo-mode          Configure echo mode
end                End current mode and change to enable mode
exit               Exit current mode and down to previous mode
find               Find CLI command matching a regular expression
list               Print command list
minimum-ttl        Expect packets with at least this TTL
no                 Negate a command or set its defaults
output            Direct vtysh output to file
passive-mode       Don't attempt to start sessions
profile            Use BFD profile settings
quit              Exit current mode and down to previous mode
```

```
receive-interval  Configure peer receive interval
shutdown          Disable BFD peer
transmit-interval Configure peer transmit interval
```

После того, как с одной из сторон такой пир сконфигурирован, он будет находиться в статусе **down**, пока не будет создан симметричный пир с "противоположной" стороны.

6.4.7 Профиль

Чтобы создать новый профиль, отредактировать или удалить существующий профиль, необходимо перейти из режима глобальной конфигурации в режим настройки протокола BFD, а дальше уже либо удалять профиль, либо переходить в режим создания или настройки конкретного профиля.

```
angie-va(config)# bfd
angie-va(config-bfd)#
end          End current mode and change to enable mode
exit        Exit current mode and down to previous mode
find        Find CLI command matching a regular expression
list        Print command list
no          Negate a command or set its defaults
output      Direct vtysh output to file
peer        Configure peer
profile     BFD profile.
quit        Exit current mode and down to previous mode
angie-va(config-bfd)# profile
BFDPROF     BFD profile name.
test
angie-va(config-bfd)# no profile test
angie-va(config-bfd)# profile test_new
angie-va(config-bfd-profile)#
detect-multiplier  Configure peer detection multiplier
echo              Configure peer echo intervals
echo-interval     Configure peer echo interval
echo-mode         Configure echo mode
end              End current mode and change to enable mode
exit            Exit current mode and down to previous mode
find            Find CLI command matching a regular expression
list            Print command list
minimum-ttl      Expect packets with at least this TTL
no              Negate a command or set its defaults
output          Direct vtysh output to file
passive-mode     Don't attempt to start sessions
quit            Exit current mode and down to previous mode
receive-interval Configure peer receive interval
shutdown        Disable BFD peer
transmit-interval Configure peer transmit interval
angie-va(config-bfd-profile)#
```

6.5 Настройка RHI

RHI (Route Health Injection) позволяет динамически управлять маршрутами, анонсируемыми протоколами динамической маршрутизации, например BGP и OSPF, на основе данных о состоянии апстримов в балансировщике нагрузки. RHI позволяет автоматически отзываться префиксы при недоступности апстримов, минимизируя возможные циклы объявлений и отзывов. Апстрим считается недоступным, если все его бэкенды не прошли проверки работоспособности (health probes).

Для работы RHI необходимо:

1. Настроить BGP или OSPF в зависимости от свойств сети (через командную строку, примеры см. ниже).
2. Задать проверки работоспособности для балансировщика нагрузки (в веб-интерфейсе Angie ADC, пример см. ниже).
3. Задать конфигурацию RHI (в веб-интерфейсе Angie ADC, инструкцию см. ниже).

6.5.1 Настройка BGP

Настройка BGP проводится через *интерфейс командной строки*.

6.5.1.1 Пример

```
hostname angie-va
no ipv6 forwarding
ip prefix-list myapp1_pair seq 5 permit 2.2.2.2/32
interface lo
 ip address 2.2.2.2/32
exit
router bgp 1
 no bgp ebgp-requires-policy
 neighbor 10.0.0.1 remote-as 2
 address-family ipv4 unicast
  redistribute connected route-map myapp1_pair
 exit-address-family
exit
route-map myapp1_pair permit 10
 match ip address prefix-list myapp1_pair
exit
```

6.5.1.2 Проверка

Angie ADC должен увидеть свои объявления:

```
angie-va# sh ip bgp
BGP table version is 14, local router ID is 2.2.2.2, vrf id 0
Default local pref 100, local AS 1
Status codes:  s suppressed, d damped, h history, u unsorted, * valid, > best, =_
↳multipath,
                i internal, r RIB-failure, S Stale, R Removed
Nexthop codes: @NNN nexthop's vrf id, < announce-nh-self
Origin codes:  i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

   Network          Next Hop              Metric LocPrf Weight Path
*>  2.2.2.2/32      0.0.0.0                  0           32768 ?

Displayed 1 routes and 1 total paths
```

Соседний маршрутизатор должен увидеть объявления от Angie ADC:

```
abstract-router# sh ip bgp
BGP table version is 14, local router ID is 10.0.0.1, vrf id 0
Default local pref 100, local AS 2
Status codes:  s suppressed, d damped, h history, u unsorted, * valid, > best, =_
↳multipath,
                i internal, r RIB-failure, S Stale, R Removed
Nexthop codes: @NNN nexthop's vrf id, < announce-nh-self
```

```
Origin codes:  i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 2.2.2.2/32	10.21.20.6	0		0	1 ?

Displayed 1 routes and 1 total paths

В случае если объявления не видны, можно попробовать сбросить сессию BGP на обоих маршрутизаторах:

```
clear ip bgp * soft
```

6.5.1.3 Отзыв префиксов

Отзыв префиксов реализован через автоматическое изменение префикс-листов.

Например:

```
ip prefix-list myapp1_pair seq 5 deny 2.2.2.2/32
```

6.5.1.4 Объявление префиксов после восстановления работы апстрима

При восстановлении работоспособности апстримов префикс-листы не меняются автоматически.

Чтобы вручную вернуть префикс-лист к изначальному виду, выполните команды:

```
conf t
ip prefix-list myapp1_pair seq 5 permit 2.2.2.2/32
```

Примечание

Если вы хотите, чтобы префиксы автоматически анонсировались после восстановления апстримов, установите флажок **Объявлять префиксы при восстановлении апстримов** в настройках конфигурации RHI.

6.5.2 Настройка OSPF

Настройка OSPF также проводится через *интерфейс командной строки*.

6.5.2.1 Пример

```
hostname angie-va
no ipv6 forwarding
ip prefix-list myapp1_pair seq 5 permit 2.2.2.2/32
interface ens33
ip address 192.168.0.20/24
ip ospf 1 area 0
exit
interface lo
ip address 2.2.2.2/32
exit
router ospf 1
ospf router-id 192.168.0.20
redistribute connected route-map myapp1_pair
exit
route-map myapp1_pair permit 10
```

```
match ip address prefix-list myapp1_pair
exit
```

6.5.2.2 Проверка

Если серверы в бэкенде доступны:

```
angie-v#sho ip ospf 1 database external self-originate
OSPF Instance: 1
    OSPF Router with ID (2.2.2.2)
        AS External Link States
    LS age: 28
    Options: 0x2 : *|-|-|-|-|E|-
    LS Flags: 0xb
    LS Type: AS-external-LSA
    Link State ID: 2.2.2.2 (External Network Number)
    Advertising Router: 192.168.0.20
    LS Seq Number: 80000001
    Checksum: 0x139b
    Length: 36
    Network Mask: /32
        Metric Type: 2 (Larger than any link state path)
        TOS: 0
        Metric: 20
        Forward Address: 0.0.0.0
        External Route Tag: 0
```

Если серверы в бэкенде недоступны:

```
angie-v#sho ip ospf 1 database external self-originate
OSPF Instance: 1
    OSPF Router with ID (192.168.0.20)
        AS External Link States
    LS age: 3600
    Options: 0x2 : *|-|-|-|-|E|-
    LS Flags: 0x8b
    LS Type: AS-external-LSA
    Link State ID: 2.2.2.2 (External Network Number)
    Advertising Router: 2.2.2.2
    LS Seq Number: 80000001
    Checksum: 0x139b
    Length: 36
    Network Mask: /32
        Metric Type: 2 (Larger than any link state path)
        TOS: 0
        Metric: 20
        Forward Address: 0.0.0.0
        External Route Tag: 0
```

6.5.2.3 Отзыв префиксов

Отзыв маршрута в OSPF происходит следующим образом: после удаления префикса OSPF изменяет значение возраста LSA в поле **LS age** на 3600, информируя соседей, что LSA устарела и нужно перестать ее использовать. Через некоторое время эта LSA будет полностью удалена из LSDB.

6.5.2.4 Объявление префиксов после восстановления работы апстрима

При восстановлении работоспособности апстримов префикс-листы не меняются автоматически.

Чтобы вручную вернуть префикс-лист к изначальному виду, выполните команды:

```
conf t
ip prefix-list myapp1_pair seq 5 permit 2.2.2.2/32
```

Примечание

Если вы хотите, чтобы префиксы автоматически анонсировались после восстановления апстримов, установите флажок **Объявлять префиксы при восстановлении апстримов** в настройках конфигурации RHI.

6.5.3 Настройка балансировщика нагрузки

Конфигурация балансировщика нагрузки настраивается в веб-интерфейсе Angie ADC (см. [инструкцию](#)).

Для работы RHI необходимо настроить следующие параметры:

- `upstream_probe`;
- параметры для получения информации о состоянии балансировщика нагрузки.

6.5.3.1 Пример

Приведены только строки, добавленные в дефолтную конфигурацию.

```
### Часть конфигурации опущена для краткости
}
### Бэкенды
upstream myapp1 {
    zone myapp1 1m;
    server server1;
    server server2;
}
### Переменная для health probe
map $upstream_status $good {
    200      "1";
}
server {
    ### Часть конфигурации опущена для краткости
    location / {
        proxy_pass http://myapp1;
        # Апстрим-пробы
        upstream_probe myapp1_probe
            interval=1s
            test=$good
            essential
            mode=always;
    }
    ### Часть конфигурации опущена для краткости
}
}
```


6.5.4 Настройка конфигурации RNI

Конфигурация RNI настраивается в веб-интерфейсе Angie ADC.

Чтобы настроить конфигурацию, выполните следующие действия:

1. Откройте веб-интерфейс Angie ADC и перейдите в раздел **Управление трафиком** → **Конфигурация RNI**.

Откроется окно настройки RNI.

2. Задайте префиксы. Для этого нажмите **Добавить** и укажите следующие параметры:

- **VIP-адрес (CIDR)** — виртуальный IP-адрес для апстрима. Указывается в формате **X.X.X.X/mask**.

Примечание

Значение должно быть уникальным в рамках одного префикс-листа.

- **Приоритет** — номер последовательности, определяющий порядок обработки правил. Маршрут с меньшим приоритетом обрабатывается первым.

Примечание

Значение должно быть уникальным в рамках одного префикс-листа.

- **Имя префикс-листа** — имя префикс-листа для маршрутизации.
- **Имя апстрима** — апстрим, для которого настраивается отзыв префикса.

3. Установите флажок **Объявлять префиксы при восстановлении апстримов**, если хотите, чтобы Angie ADC автоматически анонсировал префиксы в маршрутизируемую сеть при восстановлении работоспособности апстримов. По умолчанию флажок снят.

4. Нажмите **Сохранить**.

Настройки RNI будут сохранены и сразу применены.

6.5.4.1 Удаление префиксов

Вы можете удалить неиспользуемые префиксы из конфигурации RNI, нажав на значок удаления напротив этих префиксов.

Примечание

При удалении префиксов через интерфейс Angie ADC они удаляются из базы данных и перестают влиять на объявление и отзыв маршрутов, но сохраняются в списках, отображаемых через командную строку.

Чтобы полностью удалить префикс-лист, необходимо выполнить следующие команды:

```
conf t
no ip prefix-list <имя префикс-листа>
```

Это поведение будет скорректировано в следующих релизах.

ГЛАВА 7

Высокая доступность

В этом разделе:

- Обзор работы
- Создание пары высокой доступности
- Настройка высокой доступности в паре
- Управление парой
- Привязка клиентских сессий

Высокая доступность (High Availability, HA) – это режим, при котором два устройства Angie ADC объединяются в пару высокой доступности, обеспечивающую автоматическую синхронизацию конфигураций двух узлов и сохранение работоспособности системы при сбое одного из узлов пары.

Пример типовой схемы работы пары высокой доступности:

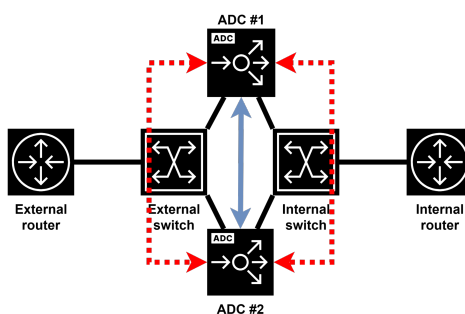


Рис. 1: Рис. 1

Примечание

Рекомендуется резервировать все компоненты инфраструктуры. Отказ сопутствующей инфраструктуры (например выход из строя вышестоящего маршрутизатора или сбой серверов) выходит за рамки решения высокой доступности Angie ADC.

Пример отказоустойчивой инфраструктуры:

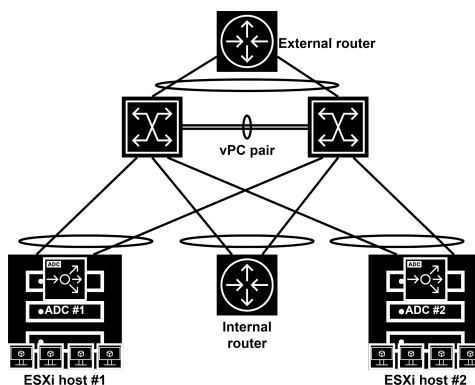


Рис. 2: Рис. 1

7.1 Обзор работы

Вы можете объединить два устройства Angie ADC в пару высокой доступности в режиме Active / Standby. В дальнейшем будут добавлены другие режимы.

Примечание

Перед настройкой пары высокой доступности необходимо создать кластер Angie ADC. Подробнее см. в статье [Создание кластера](#).

7.1.1 Режим Active / Standby

При создании пары одно устройство назначается основным, а второе — резервным. Конфигурации узлов синхронизированы. Основной узел принимает соединения и управляет серверами. Резервный узел находится в режиме ожидания. При выходе из строя основного узла происходит *смена ролей*: резервный узел становится основным, а основной после восстановления переходит в режим ожидания.

Высокая доступность достигается за счет двунаправленных проверок доступности узлов. Узлы отслеживают состояние друг друга, отправляя запросы (UDP-сообщения) на все интерфейсы соседнего узла, участвующие в паре. По умолчанию запросы отправляются раз в 250 мс. Если с момента получения последнего успешного ответа от основного узла прошло больше времени, чем указано в таймауте ожидания, интерфейс будет считаться недоступным. По умолчанию таймаут равен 3 секундам. Если проверка доступности всех интерфейсов основного узла пары завершается неудачно, происходит смена ролей. Параметры опроса можно [настраивать](#).

Поддерживается [привязка клиентских сессий](#) (sticky sessions), позволяющая сохранять клиентские сессии при смене основного узла, принимающего соединения.

Информация о состоянии пары, ролях узлов и событиях в работе пары отображается в веб-интерфейсе Angie ADC. Подробнее см. [Просмотр информации о паре](#).

7.1.2 Развертывание

Развертывание пары высокой доступности включает следующие шаги:

1. [Предварительные действия](#).
2. [Создание пары высокой доступности в режиме Active / Standby](#).
3. [Настройка высокой доступности в паре](#).

7.1.3 Обновление конфигурации и синхронизация

Конфигурации узлов автоматически синхронизируются при создании пары.

Примечание

После изменения конфигурации основного узла необходимо вручную запустить синхронизацию узлов пары.

При синхронизации основной узел передает свою конфигурацию на резервный узел. На резервном узле перезаписываются следующие файлы и папки:

- конфигурация балансировщика нагрузки;
- сертификаты и ключи балансировщика нагрузки;
- конфигурация маршрутизации (VRRP, BGP, OSPF);
- конфигурация GSLB;
- все файлы в папках `/etc/angie/crt` и `/etc/angie-lb/crt`;
- все файлы в папке `/var/transfer`.

Остальные файлы резервного узла, если они находятся вне указанных папок (например файл лицензии), удалены не будут.

Подробнее см. *Обновление конфигурации и синхронизация*.

7.2 Создание пары высокой доступности

Вы можете создать пару высокой доступности *в веб-интерфейсе Angie ADC*.

7.2.1 Требования

Для создания пары высокой доступности должны выполняться следующие условия:

- Оба устройства Angie ADC должны иметь одинаковую версию ПО и лицензию.

7.2.2 Предварительные действия

1. Разверните два устройства Angie ADC из будущей пары.

При этом должны выполняться следующие условия:

- В пару можно поставить устройства Angie ADC, у которых хотя бы одна общая подсеть.
- Интерфейсы самого устройства Angie ADC могут находиться в разных подсетях, но при формировании пар интерфейсов все IP-адреса попарно должны быть из одинаковых подсетей.
- Каждый узел должен иметь собственный набор уникальных IP-адресов. На данный момент поддерживаются только IPv4-адреса. Необходимо прописать IP-адреса для всех физических интерфейсов, которые будут использоваться для обеспечения высокой доступности.

Предупреждение

В дальнейшем не рекомендуется изменять IP-адреса узлов, объединенных в пару высокой доступности, иначе пара потеряет работоспособность.

Мы рекомендуем иметь одинаковое количество интерфейсов на обоих устройствах Angie ADC, чтобы каждый интерфейс получил пару на другом устройстве Angie ADC. Интерфейсы без пары не будут участвовать в обеспечении высокой доступности.

2. Объедините устройства Angie ADC в кластер. Подробнее см. в статье [Создание кластера](#).
3. Настройте авторизацию на резервном узле Angie ADC с использованием локальной учетной записи `admin` (эти учетные данные необходимы для создания пары).
4. Настройте минимальную конфигурацию узла, который будет иметь роль основного. При создании пары запустится синхронизация конфигураций (конфигурация основного узла будет перенесена на резервный узел). Настройку также можно выполнить позже, после создания пары, при этом необходимо будет вручную запустить синхронизацию конфигураций после настройки.

7.2.3 Создание пары высокой доступности в режиме Active / Standby

При создании пары высокой доступности роль основного узла (Active) автоматически назначается тому узлу, с которого создается пара, роль резервного (Standby) – добавляемому узлу.

Важно

Для успешной сборки и работы пары высокой доступности необходимо, чтобы время на узлах пары было синхронизировано, например, по *NTP*.

Чтобы создать пару высокой доступности, выполните следующие действия:

1. Откройте веб-интерфейс Angie ADC. В панели навигации перейдите в раздел **Высокая доступность и кластеризация** → **Высокая доступность**.

Откроется экран подключения резервного узла к текущему узлу.

2. На экране **Создание пары высокой доступности** заполните следующие поля:
 - **IP-адрес** – IP-адрес второго узла пары (IPv4) для удаленного администрирования и мониторинга этого узла. Это может быть выделенный интерфейс только для управления или один из интерфейсов, передающих продуктовый трафик. Этот адрес не распространяется и не синхронизируется.
 - **Логин и Пароль** – данные для авторизации на резервном узле.

Примечание

Используйте логин локальной учетной записи `admin` (см. "Предварительные действия" выше).

3. Нажмите **Подключить**.

Между двумя узлами будет создано соединение. Откроется окно настройки соответствия интерфейсов.

4. В окне **Настройка соответствия интерфейсов** в раскрывающихся списках основного и резервного узла выберите интерфейсы, которые вы хотите поставить в пару. Опрос доступности узлов будет проводиться в рамках каждой добавленной пары.

Чтобы добавить еще одну пару, нажмите значок **Добавить** и повторите действия.

В списках отображаются только интерфейсы в состоянии `up`. Интерфейсы `loopback`, в состоянии `down` и без IP-адреса не отображаются и не доступны для выбора.

Примечание

Адреса двух сопоставляемых интерфейсов должны быть из одной подсети. Минимально достаточно одной пары интерфейсов, но мы рекомендуем задать пары для всех интерфейсов обоих узлов.

5. Нажмите **Сохранить**.

Будет сформирована пара высокой доступности в режиме Active / Standby. Сразу после создания у пары будет отображаться статус **starting** (происходит синхронизация конфигураций). После успешной синхронизации статус поменяется на **on** (пара в рабочем состоянии).

6. После объединения двух устройств Angie ADC в пару необходимо *настроить механизмы высокой доступности* на узлах этой пары.

7.3 Настройка высокой доступности в паре

После объединения двух устройств Angie ADC в пару необходимо настроить механизмы высокой доступности на узлах этой пары.

Для этого необходимо выбрать механизм обеспечения высокой доступности. Поддерживаются следующие механизмы:

- VRRP;
- OSPF;
- BGP.

Можно одновременно использовать несколько механизмов на разных интерфейсах устройства.

Настройку основного узла можно проводить до или после создания пары высокой доступности. При создании пары уже имеющиеся конфигурации синхронизируются, а после создания пары можно запустить ручную синхронизацию конфигураций.

7.3.1 VRRP

Для обеспечения высокой доступности по VRRP необходимо настроить общий VRRP-интерфейс на двух узлах пары. Для этого на каждом узле пары перейдите в *CLI (порт 2022)* и задайте параметры интерфейса:

```
vrrp <номер_группы> int <имя_интфейса> ip <IP_адрес>
```

Также настройки, относящиеся к VRRP, можно продублировать, задав их на одном узле и затем запустив ручную *синхронизацию пары*.

На активном узле виртуальный VRRP-интерфейс будет находиться в состоянии UP, а на резервном узле соответствующий виртуальный интерфейс будет выключен. При смене ролей меняется состояние виртуальных VRRP-интерфейсов на узлах (виртуальный VRRP-интерфейс узла, принявшего на себя роль Active, перейдет в состояние UP). Таким образом можно дополнительно организовать резервирование пары высокой доступности другим устройством или группой.

Пример вывода команды `ip a`, выполненной на активном узле (интерфейс `vrrp4-ens33-1@ens33` находится в состоянии UP):

```
ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
```

```
inet 203.0.113.1/32 brd 203.0.113.1 scope global lo
valid_lft forever preferred_lft forever
inet6 ::1/128 scope host noprefixroute
valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
↳ default qlen 1000
link/ether 00:0c:29:96:0c:dc brd ff:ff:ff:ff:ff:ff
altname enp2s1
inet 10.65.1.7/24 brd 10.65.1.255 scope global ens33
valid_lft forever preferred_lft forever
inet6 fe80::20c:29ff:fe96:cdc/64 scope link
valid_lft forever preferred_lft forever
3: vrrp4-ens33-1@ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
↳ state UP group default qlen 1000
link/ether 00:00:5e:00:01:01 brd ff:ff:ff:ff:ff:ff
inet 10.65.1.5/24 metric 1000 scope global vrrp4-ens33-1
valid_lft forever preferred_lft forever
inet6 fe80::200:5eff:fe00:101/64 scope link
valid_lft forever preferred_lft forever
```

Пример вывода команды `ip a`, выполненной на резервном узле (интерфейс `vrrp4-ens33-1@ens33` находится в состоянии DOWN):

```
ip a

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
↳ qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
valid_lft forever preferred_lft forever
inet 203.0.113.1/32 brd 203.0.113.1 scope global lo
valid_lft forever preferred_lft forever
inet6 ::1/128 scope host noprefixroute
valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
↳ default qlen 1000
link/ether 00:0c:29:c1:36:b4 brd ff:ff:ff:ff:ff:ff
altname enp2s1
inet 10.65.1.8/24 brd 10.65.1.255 scope global ens33
valid_lft forever preferred_lft forever
inet6 fe80::20c:29ff:fec1:36b4/64 scope link
valid_lft forever preferred_lft forever
5: vrrp4-ens33-1@ens33: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue
↳ state DOWN group default qlen 1000
link/ether 00:00:5e:00:01:01 brd ff:ff:ff:ff:ff:ff protodown on protodown_reason <7>
inet 10.65.1.5/24 metric 1000 scope global vrrp4-ens33-1
valid_lft forever preferred_lft forever
inet6 fe80::200:5eff:fe00:101/64 scope link
valid_lft forever preferred_lft forever
```

Далее вы можете настроить высокую доступность двумя способами:

- Назначение адреса сервиса на виртуальный VRRP-интерфейс.
- Назначение адреса сервиса на интерфейс `lo`.

7.3.1.1 Назначение адреса сервиса на виртуальный VRRP-интерфейс

При смене роли узла сервис будет перенесен на тот узел, который взял на себя роль Active. Допускается назначение нескольких IP-адресов одновременно на один виртуальный интерфейс. Для маршрутизатора адрес сервиса находится в подключенной сети, то есть доступен напрямую:

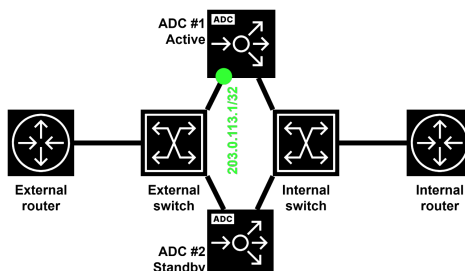


Рис. 3: Рис. 1

Если клиенты находятся в одной IP-сети с адресом сервиса, этот способ будет наиболее удобным:

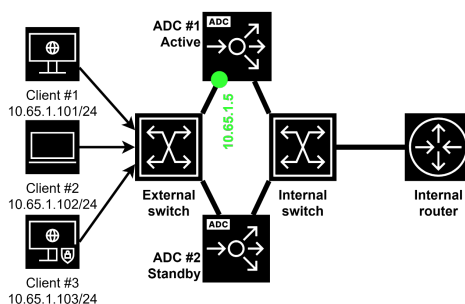


Рис. 4: Рис. 2

7.3.1.2 Назначение адреса сервиса на интерфейс lo

При настройке статического маршрута на маршрутизаторе необходимо указать виртуальный VRRP-адрес как адрес следующего перехода (next-hop) для префикса, содержащего адрес сервиса. На интерфейс lo может быть назначено несколько адресов одновременно. Мы рекомендуем использовать этот способ как более гибкий:

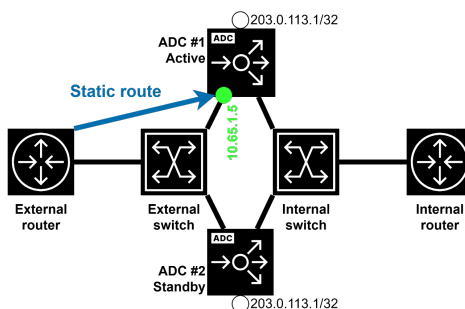


Рис. 5: Рис. 3

7.3.2 OSPF

Если в сети используются протоколы динамической маршрутизации, то вы можете добавить пару высокой доступности Angie ADC в OSPF-домен. При этом только основной узел пары будет анонсировать маршрут на адрес сервиса.

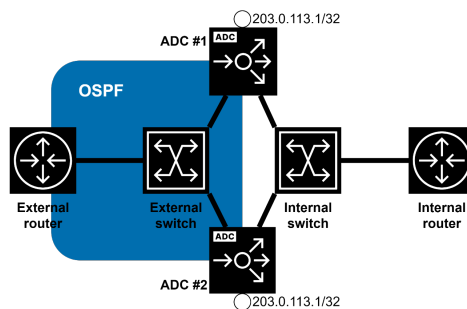


Рис. 6: Рис. 4

7.3.2.1 Пример

Сервис доступен по адресу 203.0.113.1. Необходимо, чтобы основной узел пары анонсировал маршрут на префикс 203.0.113.1/32.

Сначала нужно вручную назначить адрес 203.0.113.1/32 на интерфейсы loopback обоих узлов пары высокой доступности.

Интерфейсы основного узла:

```
ip a

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
   ↪qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
   valid_lft forever preferred_lft forever
inet 203.0.113.1/32 scope global lo
   valid_lft forever preferred_lft forever
inet6 ::1/128 scope host noprefixroute
   valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
   ↪default qlen 1000
link/ether 00:0c:29:96:0c:dc brd ff:ff:ff:ff:ff:ff
altname enp2s1
inet 10.65.1.7/24 brd 10.65.1.255 scope global ens33
   valid_lft forever preferred_lft forever
inet6 fe80::20c:29ff:fe96:cdc/64 scope link
   valid_lft forever preferred_lft forever
```

Интерфейсы резервного узла:

```
ip a

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
   ↪qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
   valid_lft forever preferred_lft forever
inet 203.0.113.1/32 scope global lo
   valid_lft forever preferred_lft forever
inet6 ::1/128 scope host noprefixroute
   valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
   ↪default qlen 1000
link/ether 00:0c:29:c1:36:b4 brd ff:ff:ff:ff:ff:ff
```

```
altname enp2s1
inet 10.65.1.8/24 brd 10.65.1.255 scope global ens33
valid_lft forever preferred_lft forever
inet6 fe80::20c:29ff:fec1:36b4/64 scope link
valid_lft forever preferred_lft forever
```

Далее необходимо добавить поддержку протокола OSPF на интерфейсе **ens33** и включить процесс маршрутизации OSPF, в котором анонсируются все подключенные сети.

```
interface ens33
ip ospf 1 area 0
router ospf 1
redistribute connected
distribute-list %ha%pair% out connected
```

После внесения изменений *синхронизируйте конфигурации в паре.*

Примечание

Последняя строка добавляется в момент выполнения синхронизации. Не нужно вносить эту строку вручную, а также удалять или изменять ее.

7.3.2.2 Проверка

Проверка на маршрутизаторе или L3-коммутаторе (список OSPF-соседей):

```
R1#sho ip ospf nei
```

Neighbor ID	Pri	State	Dead Time	Address	Interface
1.1.1.1	1	FULL/BDR	00:00:39	10.65.1.7	GigabitEthernet0/0
2.2.2.2	1	FULL/DR	00:00:38	10.65.1.8	GigabitEthernet0/0

Соседство по протоколу OSPF устанавливают оба узла пары одновременно:

```
R1#sho ip ro os
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
+ - replicated route, % - next hop override

Gateway of last resort is not set
203.0.113.0/32 is subnetted, 1 subnets
O E2      203.0.113.1 [110/20] via 10.65.1.7, 00:52:39, GigabitEthernet0/0
```

Маршрут анонсируется только одним узлом. В этом можно убедиться, если просмотреть список LSA5 в LSDB маршрутизатора:

```
R1#sho ip os database external
      OSPF Router with ID (10.65.1.140) (Process ID 1)

      Type-5 AS External Link States
Routing Bit Set on this LSA in topology Base with MTID 0
LS age: 1341
```

```
Options: (No TOS-capability, No DC, Upward)
LS Type: AS External Link
Link State ID: 203.0.113.1 (External Network Number )
Advertising Router: 1.1.1.1
LS Seq Number: 80000005
Checksum: 0x575B
Length: 36
Network Mask: /32
Metric Type: 2 (Larger than any link state path)
MTID: 0
Metric: 20
Forward Address: 0.0.0.0
External Route Tag: 0
```

Теперь выключите основной узел и проверьте изменения:

```
R1#sho ip os ne
```

Neighbor ID	Pri	State	Dead Time	Address	Interface
2.2.2.2	1	FULL/DR	00:00:33	10.65.1.8	GigabitEthernet0/0

```
R1#sho ip os da ex
```

```
OSPF Router with ID (10.65.1.140) (Process ID 1)
```

```
Type-5 AS External Link States
```

```
Routing Bit Set on this LSA in topology Base with MTID 0
```

```
LS age: 114
```

```
Options: (No TOS-capability, No DC, Upward)
```

```
LS Type: AS External Link
```

```
Link State ID: 203.0.113.1 (External Network Number )
```

```
Advertising Router: 2.2.2.2
```

```
LS Seq Number: 80000001
```

```
Checksum: 0x139B
```

```
Length: 36
```

```
Network Mask: /32
```

```
Metric Type: 2 (Larger than any link state path)
```

```
MTID: 0
```

```
Metric: 20
```

```
Forward Address: 0.0.0.0
```

```
External Route Tag: 0
```

```
R1#sho ip ro os
```

```
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
```

```
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
```

```
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
```

```
E1 - OSPF external type 1, E2 - OSPF external type 2
```

```
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
```

```
ia - IS-IS inter area, * - candidate default, U - per-user static route
```

```
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
```

```
+ - replicated route, % - next hop override
```

```
Gateway of last resort is not set
```

```
203.0.113.0/32 is subnetted, 1 subnets
```

```
O E2 203.0.113.1 [110/20] via 10.65.1.8, 00:01:57, GigabitEthernet0/0
```

```
R1#
```

Остался только один OSPF-сосед, который стал основным узлом пары. Теперь он анонсирует марш-

пут до префикса 203.0.113.1/32.

7.3.3 BGP

Маршрутизатор устанавливает BGP-сессии с обоими узлами пары. Обе BGP-сессии активны одновременно, но анонс префикса производится только с основного узла.

Пример типовой схемы:

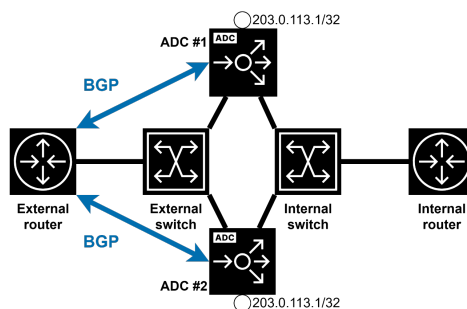


Рис. 7: Рис. 5

7.3.3.1 Пример

Сервис доступен по адресу 203.0.113.1. Основной узел пары будет анонсировать маршрут на префикс 203.0.113.1/32.

Сначала необходимо вручную назначить адрес 203.0.113.1/32 на интерфейсы loopback обоих узлов пары высокой доступности.

Интерфейсы основного узла:

```
ip a

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default_
   ↪qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
   inet 203.0.113.1/32 scope global lo
       valid_lft forever preferred_lft forever
   inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group_
   ↪default qlen 1000
   link/ether 00:0c:29:96:0c:dc brd ff:ff:ff:ff:ff:ff
   altname enp2s1
   inet 10.65.1.7/24 brd 10.65.1.255 scope global ens33
       valid_lft forever preferred_lft forever
   inet6 fe80::20c:29ff:fe96:cdc/64 scope link
       valid_lft forever preferred_lft forever
```

Интерфейсы резервного узла:

```
ip a

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default_
   ↪qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
```

```
valid_lft forever preferred_lft forever
inet 203.0.113.1/32 scope global lo
valid_lft forever preferred_lft forever
inet6 ::1/128 scope host noprefixroute
valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group┐
└─>default qlen 1000
link/ether 00:0c:29:c1:36:b4 brd ff:ff:ff:ff:ff:ff
altname enp2s1
inet 10.65.1.8/24 brd 10.65.1.255 scope global ens33
valid_lft forever preferred_lft forever
inet6 fe80::20c:29ff:fec1:36b4/64 scope link
valid_lft forever preferred_lft forever
```

На основном узле настройте процесс BGP так, чтобы он устанавливал соединение с маршрутизатором (10.65.1.140). Также перераспределите подключенные сети в BGP:

```
router bgp 65001
neighbor 10.65.1.140 remote-as 65001
!
address-family ipv4 unicast
redistribute connected
exit-address-family
exit
```

Запустите *синхронизацию конфигураций* между узлами пары высокой доступности, чтобы настроить процесс маршрутизации BGP на резервном узле.

Примечание

Ручная настройка конфигурации на резервном узле не должна использоваться.

7.3.3.2 Проверка

Пример конфигурации со стороны маршрутизатора:

```
router bgp 65001
bgp log-neighbor-changes
neighbor 10.65.1.7 remote-as 65001
neighbor 10.65.1.8 remote-as 65001
```

Убедитесь, что маршрутную информацию отправляет только основной узел (10.65.1.7):

```
R1#sho ip bg su
BGP router identifier 10.65.1.140, local AS number 65001
BGP table version is 17, main routing table version 17
2 network entries using 296 bytes of memory
2 path entries using 128 bytes of memory
1/1 BGP path/bestpath attribute entries using 136 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 560 total bytes of memory
BGP activity 4/2 prefixes, 13/11 paths, scan interval 60 secs

Neighbor          V            AS MsgRcvd MsgSent   TblVer  InQ  OutQ Up/Down  State/
└─>PfxRcd
10.65.1.7          4            65001    3283    3206      17    0    0 02:43:57      2
```

```

10.65.1.8      4      65001      28      29      17      0      0 00:01:17      0
R1#sho ip bgp
BGP table version is 17, local router ID is 10.65.1.140
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found
Network          Next Hop          Metric LocPrf Weight Path
r>i 10.65.1.0/24   10.65.1.7              0    100      0 ?
*>i 203.0.113.1/32 10.65.1.7              0    100      0 ?
R1#sho ip ro bgp
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
+ - replicated route, % - next hop override

Gateway of last resort is not set
203.0.113.0/32 is subnetted, 1 subnets
B      203.0.113.1 [200/0] via 10.65.1.7, 02:45:34
R1#

```

Теперь выключите основной узел и проверьте изменения:

```

R1#sho ip bg su
BGP router identifier 10.65.1.140, local AS number 65001
BGP table version is 21, main routing table version 21
2 network entries using 296 bytes of memory
2 path entries using 128 bytes of memory
1/1 BGP path/bestpath attribute entries using 136 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 560 total bytes of memory
BGP activity 6/4 prefixes, 15/13 paths, scan interval 60 secs

Neighbor      V      AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/
↪PfxRcd
10.65.1.7      4      65001      0       0        1    0    0 00:07:13 Active
10.65.1.8      4      65001    228     222       21    0    0 00:11:04      2

```

Остался только один BGP-сосед (10.65.1.8), который стал основным узлом пары. Теперь он анонсирует маршрут до префикса 203.0.113.1/32:

```

R1#sho ip bg
BGP table version is 21, local router ID is 10.65.1.140
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found
Network          Next Hop          Metric LocPrf Weight Path
r>i 10.65.1.0/24   10.65.1.8              0    100      0 ?
*>i 203.0.113.1/32 10.65.1.8              0    100      0 ?
R1#sho ip ro bg

```

```
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
+ - replicated route, % - next hop override

Gateway of last resort is not set
203.0.113.0/32 is subnetted, 1 subnets
B      203.0.113.1 [200/0] via 10.65.1.8, 00:05:50
```

За анонс отвечает distribute-list с именем `%ha%pair%`, который автоматически прикрепляется ко всем BGP-соседям, описанным в конфигурации. Соответствующая конфигурационная строка добавляется в момент синхронизации конфигурации между узлами пары:

```
router bgp 65001
neighbor 10.65.1.140 remote-as 65001
!
address-family ipv4 unicast
redistribute connected
neighbor 10.65.1.140 distribute-list %ha%pair% out
exit-address-family
exit
```

Если узел является основным, то distribute-list ссылается на разрешающий список доступа:

```
access-list %ha%pair% seq 5 permit any
```

Если узел является резервным, то distribute-list ссылается на запрещающий список доступа:

```
access-list %ha%pair% seq 5 deny any
```

7.4 Управление парой

В этой статье:

- [Просмотр информации о паре](#)
- [Обновление конфигурации и синхронизация](#)
- [Настройка протокола опроса доступности узлов](#)
- [Удаление пары](#)

См. также [Привязка клиентских сессий](#) (sticky sessions).

7.4.1 Просмотр информации о паре

1. Откройте веб-интерфейс Angie ADC на одном из узлов пары.
2. Перейдите [Высокая доступность и кластеризация](#) → [Высокая доступность](#).

Откроется экран просмотра свойств пары.

Отображаются следующие параметры:

Состояние пары	Состояние пары высокой доступности. Возможны следующие варианты: • on – статус отображается при нормальной работе пары. Высокая доступность обеспечивается. • failed – статус отображается после попытки запуска пары, если пару сформировать не удалось. Синхронизация и обмен данными между узлами в паре невозможны. Высокая доступность не обеспечивается. • starting – пара запускается. Статус отображается сразу после создания пары в процессе ее конфигурации и синхронизации. • not_reachable – статус отображается, когда узел не получает ответ от соседнего узла по всем интерфейсам, участвующим в паре, в течение заданного таймаута (например из-за сетевой проблемы или недоступности соседнего узла). Если резервный узел фиксирует недоступность основного узла, выполняется переключение ролей: резервный узел становится основным. При этом для соседнего узла будет отображаться его последняя известная роль. Примечание Рекомендуется проверить соседний узел и, если требуется, восстановить его связность и работоспособность. Совет Чтобы убрать сообщение о смене статуса в паре, запустите ручную синхронизацию конфигураций по кнопке Синхронизировать.
Последнее изменение	Дата и время последнего изменения в паре, например смены ролей узлов.
Узел	Идентификация узла пары, на котором открыт веб-интерфейс Angie ADC: • Текущий – узел, на котором в данный момент открыт веб-интерфейс Angie ADC со свойствами пары; • Соседний – второй узел пары.
IP-адрес узла	IP-адрес узла, используемый для взаимодействия в паре. Предупреждение Не рекомендуется изменять настройки интерфейсов узлов, объединенных в пару высокой доступности, иначе пара может потерять связность и работоспособность.
Роль узла	Роль узла пары: • Основной – обрабатывает трафик (находится в режиме Active). • Резервный – находится в режиме ожидания (Standby).

7.4.2 Обновление конфигурации и синхронизация

После внесения изменений в конфигурацию основного узла необходимо вручную запустить синхронизацию пары, чтобы перенести внесенные изменения на резервный узел.

Чтобы обновить конфигурацию на узлах пары, выполните следующие действия:

1. Внесите необходимые изменения в конфигурацию основного узла.
2. Откройте веб-интерфейс Angie ADC на любом узле пары.

3. Перейдите **Высокая доступность и кластеризация** → **Высокая доступность**.

Откроется экран просмотра свойств пары.

4. Нажмите **Синхронизировать**.

Конфигурация основного узла будет перенесена на резервный узел, на экране отобразится сообщение **Пара синхронизирована**. Если синхронизация завершится с ошибкой, на экране отобразится сообщение **Ошибка синхронизации**.

На резервном узле перезаписываются следующие файлы и папки:

- конфигурация балансировщика нагрузки;
- сертификаты и ключи балансировщика нагрузки;
- конфигурация маршрутизации (VRRP, BGP, OSPF);
- конфигурация GSLB;
- все файлы в папках `/etc/angie/crt` и `/etc/angie-lb/crt`;
- все файлы в папке `/var/transfer`.

7.4.3 Настройка протокола опроса доступности узлов

Чтобы изменить настройки протокола опроса доступности узлов, выполните следующие действия:

1. Откройте веб-интерфейс Angie ADC на одном из узлов пары.
2. Перейдите **Высокая доступность и кластеризация** → **Высокая доступность**.

Откроется экран просмотра узлов, входящих в пару.

3. В блоке **Проверка доступности узлов** укажите следующие данные:

- **Интервал отправки запросов (мс)**

Интервал отправки запросов резервным узлом для проверки доступности основного узла. По умолчанию запросы отправляются каждые 250 мс. Допустимый диапазон значений – от 200 до 1000 мс.

- **Таймаут ожидания ответа (с)**

Таймаут ожидания ответа от интерфейса основного узла. Если с момента получения последнего успешного ответа прошло больше заданного времени интерфейс будет считаться недоступным. При недоступности всех интерфейсов основного узла произойдет переключение ролей. По умолчанию – 3 секунды. Допустимый диапазон значений – от 1 до 3 секунд.

4. Нажмите **Сохранить**.

Новые настройки будут применены сразу.

7.4.4 Удаление пары

i Примечание

Удаление пары возможно только с основного узла.

Чтобы удалить пару, выполните следующие действия:

1. Откройте веб-интерфейс Angie ADC на узле, который является основным в паре на данный момент.
2. Перейдите **Высокая доступность и кластеризация** → **Высокая доступность**.

Откроется экран просмотра свойств пары.

3. Нажмите **Расформировать** и подтвердите удаление.

Конфигурация пары высокой доступности будет удалена. Конфигурации узлов сохраняются, узлы продолжают функционировать независимо друг от друга.

7.5 Привязка клиентских сессий

В паре высокой доступности поддерживается привязка клиентских сессий (sticky sessions). При переключении узлов все запросы клиента в рамках одной сессии будут по-прежнему перенаправляться на один и тот же бэкенд-сервер, т.е. клиентские сессии будут сохраняться.

Информация о клиентских сессиях хранится в хранилище сессий Angie ADC. Привязка сессий реализуется с помощью директивы *sticky* (режим *learn* с *remote_action*). Для включения функции необходимо указать адрес хранилища сессий Angie ADC (http://127.0.0.1:3380/sticky_session) в конфигурации балансировщика нагрузки.

Пример конфигурации:

```
http {

    upstream sticky_test {

        server 127.0.0.1:8091 sid=web-server-01;
        server 127.0.0.1:8092 sid=web-server-02;

        sticky learn
            lookup=$http_x_userid
            remote_action=/create_session
            remote_result=$upstream_http_x_sid;
    }

    server {

        listen *:80;

        location / {

            proxy_pass http://sticky_test/;

        }

        location /create_session {

            internal;
            proxy_set_header X-Sticky-Sessid $sticky_sessid;
            proxy_set_header X-Sticky-Sid    $sticky_sid;
            proxy_set_header X-Sticky-Last   $msec;
            proxy_method POST;
            proxy_pass http://127.0.0.1:3380/sticky_session;
            proxy_connect_timeout 1s;
            proxy_read_timeout    1s;

        }

    }

}
```

ГЛАВА 8

Кластеризация

Кластер Angie ADC позволяет настраивать несколько устройств Angie ADC, объединенных в кластер, с одного узла. В кластере синхронизируются:

- конфигурации GSLB-узлов;
- сертификаты;
- пользователи SNMP;
- настройки парольной политики.

Конфигурация локального балансировщика не синхронизируется между узлами кластера.

Данные между узлами кластера передаются в зашифрованном виде.

Важно

Для успешной сборки и работы кластера необходимо, чтобы время на всех узлах кластера было синхронизировано, например, по *NTP*.

В этом разделе:

- *Создание кластера*
- *Просмотр состояния кластера*

8.1 Создание кластера

Кластер состоит из нескольких узлов (устройств Angie ADC). Первый узел кластера является точкой входа. Через этот узел осуществляется добавление других узлов в кластер.

Развертывание кластера состоит из следующих этапов:

1. Развертывание устройств Angie ADC в дата-центрах.
2. *Создание точки входа в кластер* на одном из узлов.
3. *Добавление остальных узлов в кластер.*

⚠ Предупреждение

После создания точки входа ее нельзя изменить, кластер нельзя расформировать. Внимательно проверьте все параметры перед созданием кластера.

📌 Важно

В качестве IP-адресов узлов кластера нельзя указывать *IP-адрес интерфейса управления*, если он был настроен.

8.1.1 Создание точки входа в кластер

Чтобы создать точку входа в кластер, выполните следующие действия:

1. Откройте веб-интерфейс Angie ADC на узле, который будет точкой входа.
2. В панели навигации перейдите в раздел **Высокая доступность и кластеризация** → **Кластер**.
Откроется экран с информацией о текущем узле и вариантами его интеграции в кластер.
3. Выберите сценарий **Создать кластер** и заполните следующие поля:
 - Имя текущего узла в кластере (произвольное). Допускаются строчные латинские буквы, цифры и одиночные дефисы внутри имени.
Это имя может отличаться от `hostname` узла и отображает логическую структуру кластера.
 - Секрет для подключения к кластеру.
 - IP-адрес текущего узла (только IPv4).

📌 Важно

В качестве IP-адреса текущего узла нельзя использовать *IP-адрес интерфейса управления*, если он был настроен.

4. Нажмите **Создать**.

Будет создан кластер. Текущий узел будет назначен точкой входа в кластер.

8.1.2 Добавление узлов в кластер

angle-adc1 > Кластер

Настройки кластера

Ожидает подключения Состояние на: 29.06.2026, 13:30

Узел не определён

Создать кластер
Текущий узел станет точкой входа в этот кластер.

Подключить к кластеру
Текущий узел будет подключен к уже существующему кластеру.

IP узла точки входа *

IP адрес

Имя текущего узла *

Секрет кластера *

IP текущего узла *

IP адрес

Подключить

Важно

Для успешной сборки и работы кластера необходимо, чтобы время на всех узлах кластера было синхронизировано, например, по *NTP*.

Для каждого последующего узла кластера выполните следующие действия:

1. Перейдите на узел Angie ADC, который вы хотите добавить в кластер.
2. В веб-интерфейсе Angie ADC перейдите **Высокая доступность и кластеризация** → **Кластер**.
Откроется экран с информацией о текущем узле и вариантами его интеграции в кластер.
3. Выберите сценарий **Подключить к кластеру** и заполните следующие поля:
 - IP-адрес узла точки входа (только IPv4);
 - Имя текущего узла в кластере (произвольное). Допускаются строчные латинские буквы, цифры и одиночные дефисы внутри имени.
Это имя может отличаться от *hostname* узла и отображает логическую структуру кластера.
 - Секрет для подключения к кластеру.
 - IP-адрес текущего узла (только IPv4).

Важно

В качестве IP-адреса текущего узла нельзя использовать *IP-адрес интерфейса управления*, если он был настроен.

4. Нажмите **Подключить**.

Узел будет подключен к кластеру.

8.2 Просмотр состояния кластера

adc-psi-2 > Кластер

Настройки кластера

Подключен
Состояние на: 01.07.2026, 12:42
Текущий узел: cluster

Статус кластера
Активный

1 пир

1 активный

0 проблемных

Пир

Пир	Статус	Upstream	Downstream	Ошибки
clus	активный	активный Последняя синхронизация: 663 мс назад Задержка: 638 мс	активный Последняя синхронизация: 686 мс назад Задержка: 7.1 мс	нет

После создания кластера на странице **Настройки кластера** отображаются:

- Имя текущего узла в кластере.
Имя узла в кластере может отличаться от `hostname` узла и отображает логическую структуру кластера.
- Статус подключения текущего узла к кластеру:
 - Ожидает подключения – узел не входит в кластер.
 - Подключен – узел подключен к кластеру.
 - Подключение... – выполняется подключение узла к кластеру.
- Статус кластера:
 - Активный – кластер запущен.
 - Подключение – выполняется запуск кластера.
- Сводная статистика других узлов кластера (пиров текущего узла).
- Таблица с подробной информацией по каждому пиру:

Пир	Имя узла, входящего в кластер.
Статус	● активный – пир находится в рабочем состоянии, данные синхронизированы. ● сбой – есть проблемы с синхронизацией данных. ● недоступен – пир недоступен.
Upstream	Статус получения данных пиром: ● активный – upstream доступен, данные синхронизированы. ● подключение – выполняется подключение к кластеру. ● синхронизация – выполняется синхронизация (получение данных пиром). ● недоступен – upstream недоступен, получение данных невозможно.
Downstream	Статус передачи данных пиром: ● активный – downstream доступен, данные могут передаваться. ● подключение – выполняется подключение к кластеру. ● недоступен – downstream недоступен, передача данных невозможна.
Последняя синхронизация	Время, прошедшее с последней синхронизации данных.
Задержка	Задержка синхронизации данных между текущим узлом и пиром.
Ошибки	Ошибки, возникающие при синхронизации данных для каждого пира.

ГЛАВА 9

Мониторинг и статистика

Angie ADC предоставляет широкий спектр возможностей мониторинга:

- статистика балансировщика нагрузки *в веб-интерфейсе*;
- детальная статистика балансировщика нагрузки *в веб-консоли Console Light* в режиме реального времени;
- системные метрики Angie ADC *в веб-интерфейсе*.

Дополнительно доступны:

- модуль *Prometheus*;
- модуль *Metric*;
- метрики балансировщика нагрузки *в API*;
- отправка метрик *по протоколу SNMPv2c*;
- экспорт метрик *во внешний Prometheus*.

В этом разделе:

- *Статистика балансировщика*
- *Мониторинг системы*
- *Мониторинг и статистика в Console Light*
- *SNMP*
- *Экспорт метрик*

9.1 Мониторинг балансировщика нагрузки

Angie ADC предоставляет доступ к детализированным графикам со статистикой по балансировщику нагрузки. Просмотреть статистику можно в разделе **Мониторинг** → **Балансировщик**.

Примечание

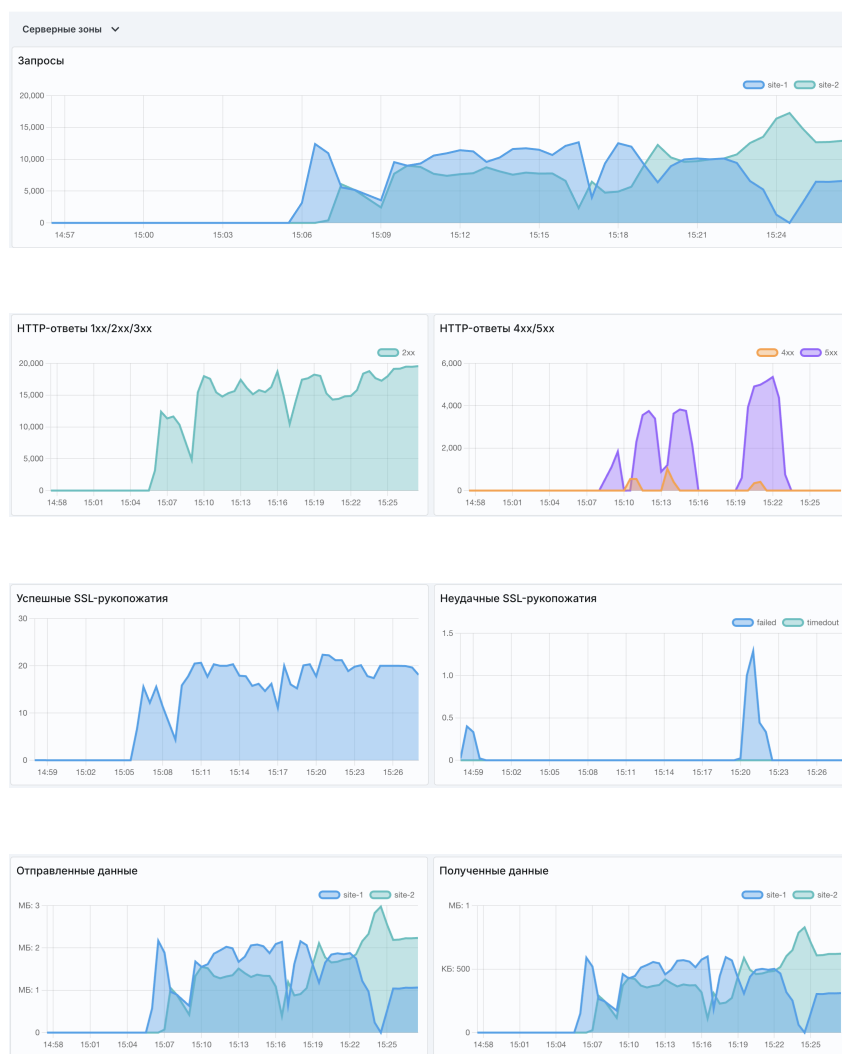
Пользователь с ролью **Оператор** может просматривать только метрики тех зон, которые были выделены для него администратором *при предоставлении доступов* к конфигурации балансировщика нагрузки.

Кнопка **Перейти к интерактивному мониторингу** позволяет перейти в консоль Console Light и посмотреть ключевые показатели нагрузки и производительности сервера в реальном времени. Подробнее см. *Мониторинг и статистика в Console Light*. Кнопка доступна только для пользователей с ролями **Администратор** и **Супервизор**.

В раскрывающемся списке **За 30 минут** в верхней части экрана можно выбрать временной период для отображения статистики. Доступные периоды: 30 минут, 3 часа, 6 часов, 12 часов, 24 часа, 48 часов, неделя, две недели.

9.1.1 Виджет "Серверные зоны"

Виджет **Серверные зоны** отображает статистику по серверным зонам разделяемой памяти:



Серверные зоны

Зона	Запр.	1...	2xx	3xx	4xx	5xx	Отпр.	Пол...
site-1	0	0	108971330		13840	1620473	ГБ: 2.95	МБ: 597.44
site-2	0	0	8770116	0	0	1865952	ГБ: 2.7	МБ: 505.04

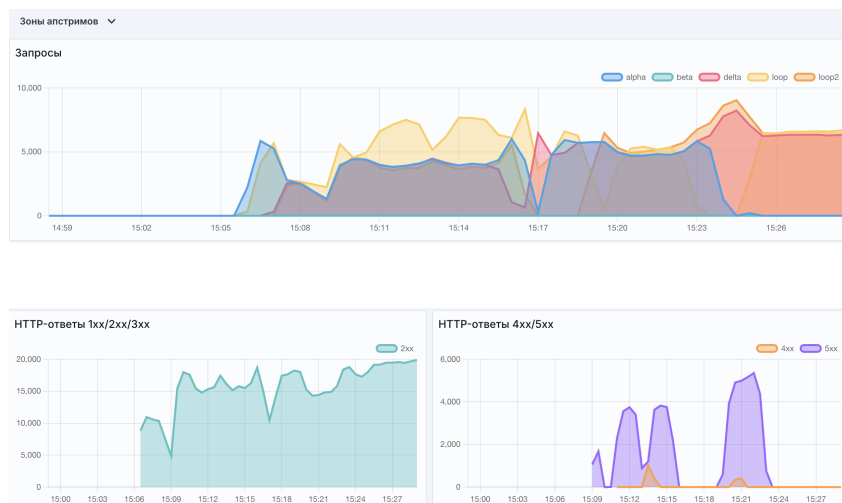
Зоны путей (Location)

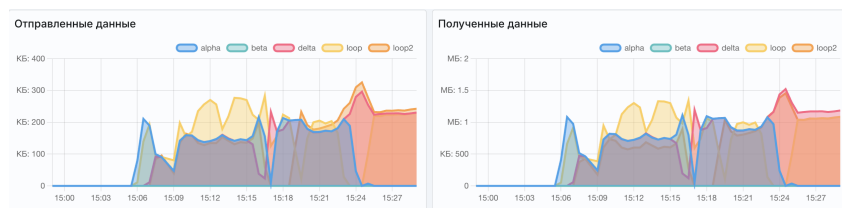
Зона	Зап...	1...	2xx	3xx	4xx	5xx	Отпр.	Пол...
location_alpha	0	0	75275430	0	0		ГБ: 1.33	МБ: 361.33
location_loop	0	0	33695560		13838	34825	МБ: 551.37	МБ: 160.87
location_beta	0	0	57867690	0	0		МБ: 960.61	МБ: 273.19
location_loop2	0	0	29833470	0	0		МБ: 480.3	МБ: 143.21

Запросы	График числа запросов с разделением по отдельным серверам
HTTP-ответы 1xx/2xx/3xx	Графики числа HTTP-ответов с определенными кодами состояния с разделением по группам кодов (1xx, 2xx, 3xx)
HTTP-ответы 4xx/5xx	Графики числа HTTP-ответов с определенными кодами состояния с разделением по группам кодов (4xx, 5xx)
Отправленные данные	График объема отправленных данных с разделением по отдельным серверам
Полученные данные	График объема полученных данных с разделением по отдельным серверам
Успешные SSL-рукопожатия	График числа успешных SSL-рукопожатий с разделением по отдельным серверам
Неудачные SSL-рукопожатия	График числа неудачных SSL-рукопожатий с разделением по отдельным серверам
Серверные зоны	Таблица серверных зон со статистикой запросов, ответов и данных с разделением по отдельным зонам
Зоны путей (Location)	Таблица зон location со статистикой запросов, ответов и данных с разделением по отдельным location

9.1.2 Виджет "Зоны апстримов"

Виджет Зоны апстримов отображает статистику по зонам разделяемой памяти для апстримов:





Запросы	График числа запросов с разделением по отдельным серверам
HTTP-ответы 1xx/2xx/3xx	Графики числа HTTP-ответов с определенными кодами состояния с разделением по группам кодов (1xx, 2xx, 3xx)
HTTP-ответы 4xx/5xx	Графики числа HTTP-ответов с определенными кодами состояния с разделением по группам кодов (4xx, 5xx)
Отправленные данные	График объема отправленных данных с разделением по отдельным серверам
Полученные данные	График объема полученных данных с разделением по отдельным серверам

9.2 Мониторинг системы

Системные метрики Angie ADC позволяют отслеживать состояние системы на разных уровнях:

- на уровне ОС (процессор, память, сеть);
- на уровне оборудования (только для аппаратного балансировщика Angie ADC: показания датчиков напряжения, температуры и пр.);
- на уровне хранения (информация о накопителях и их диагностика).

Примечание

На каждом узле Angie ADC отображаются только метрики этого узла.

Чтобы просмотреть информацию о состоянии системы, выполните следующие действия:

1. Откройте веб-интерфейс Angie ADC.
2. В панели навигации перейдите в раздел **Мониторинг** → **Система**.

Откроется экран мониторинга, отображающий метрики системы и детализированные графики со статистикой за выбранный временной интервал.

В раскрывающемся списке **За 30 минут** в верхней части экрана вы можете изменить временной интервал для отображения статистики. Доступные временные интервалы: 30 минут, 3 часа, 6 часов, 12 часов, 24 часа, 48 часов, неделя, две недели.

9.2.1 Сводная иконография

В верхней части экрана отображается сводная информация о состоянии системы.

RAM (free)	Объем свободной оперативной памяти устройства (ГБ)
ROM (free)	Объем свободной постоянной памяти устройства (ГБ)
Fan (avg.)	Скорость вращения вентиляторов: среднее значение в оборотах в минуту (RPM) по всем вентиляторам устройства
Temp. (avg.)	Средняя температура по всем датчикам устройства (по Цельсию)
Voltage	Текущее напряжение блоков питания

9.2.2 Центральный процессор

Uptime	Время непрерывной работы системы с момента последнего запуска.
CPU (core)	Количество ядер процессора.
Load Avg.	Усредненная загрузка системы за 1 минуту / 5 минут / 15 минут. Интерпретируется относительно числа ядер, например, на сервере с 4 ядрами значение 2.00 означает, что система загружена на 50%.
CPU (load) (график)	График использования процессора в процентах за выбранный временной интервал.
Load Avg. (график)	График усредненной загрузки системы за 1 минуту / 5 минут / 15 минут за выбранный временной интервал. Интерпретируется относительно числа ядер, например, на сервере с 4 ядрами значение 2.00 означает, что система загружена на 50%.

9.2.3 Обработка данных

Received	Среднее количество данных, полученных в секунду (КБ/с)
Transmitted	Среднее количество данных, переданных в секунду (КБ/с)
Collisions (avg.)	Среднее количество коллизий в сети
Input Packet Errors (avg.)	Среднее количество ошибок входящих пакетов
Output Packet Errors (avg.)	Среднее количество ошибок исходящих пакетов
Received (график)	График среднего количества данных, полученных в секунду, за выбранный временной интервал с распределением по интерфейсам
Transmitted (график)	График среднего количества данных, переданных в секунду, за выбранный временной интервал с распределением по интерфейсам

9.2.4 Сетевые интерфейсы

Таблица **Interfaces** отображает список подключенных интерфейсов.

Interface	Имя сетевого интерфейса
Adminstate	Состояние сетевого интерфейса, заданное администратором (up или down)
Operstate	Фактическое состояние сетевого интерфейса на данный момент (up или down)
Collisions, %	Процент коллизий за выбранный временной интервал
Input Packet Errors, %	Процентная доля принятых пакетов, в которых содержатся ошибки, за выбранный временной интервал
Output Packet Errors, %	Процентная доля переданных пакетов, в которых содержатся ошибки, за выбранный временной интервал

9.2.5 ОЗУ

Total	Объем оперативной памяти, установленной на сервере (ГБ)
Available	Объем свободной оперативной памяти (ГБ)
Used	Объем используемой оперативной памяти (ГБ)
ECC correctable errors	Ошибки памяти, которые были обнаружены и автоматически исправлены (шт.)
ECC uncorrectable errors	Обнаруженные ошибки памяти, которые не удалось исправить (шт.)
RAM (used)	График использования оперативной памяти в процентах за выбранный временной интервал

9.2.6 Файловая система

Круговая диаграмма **Review file system** отображает заполнение файловой системы в процентах (**Used** — использованное пространство, **Free** — свободное пространство).

Таблица **File system** содержит список смонтированных файловых систем и их параметры.

filesystem	Имя файловой системы
type	Тип файловой системы
size	Общий размер файловой системы
used	Объем использованного пространства
avail	Объем свободного пространства
use, %	Процент использования файловой системы
mountpoint	Точка монтирования файловой системы

9.2.7 Накопители (ROM)

Таблица **ROM (devices)** отображает состояние накопителей.

smartctl_status	Состояние накопителя по данным SMART (Self-Monitoring, Analysis and Reporting Technology)
power_on	Общее время работы с момента включения
temp	Текущая температура
size	Объем накопителя
device	Имя устройства
interface	Тип интерфейса подключения
model_name	Модель устройства
serial_number	Серийный номер

9.2.8 S.M.A.R.T. атрибуты диска

Таблица **Reallocated_Sector_Ct** отображает количество переназначенных (перемещенных в резервную область) секторов диска:

device	Имя устройства
raw	Необработанный атрибут SMART
thresh	Пороговое значение, при достижении которого состояние атрибута считается критическим
value	Нормализованное текущее значение атрибута
worst	Наихудшее нормализованное значение атрибута за выбранный временной интервал

Таблица **Current_Pending_Sector** отображает количество секторов, которые считаются нестабильными и ожидают переназначения:

device	Имя устройства
raw	Необработанный атрибут SMART
thresh	Пороговое значение, при достижении которого состояние атрибута считается критическим
value	Нормализованное текущее значение атрибута
worst	Наихудшее нормализованное значение атрибута за выбранный временной интервал

9.2.9 Датчики

Таблица IPMI (**temperature**) отображает список датчиков с температурами. Данные получены через интерфейс IPMI (Intelligent Platform Management Interface):

Sensor	Идентификатор датчика температуры
Temp.	Текущее значение температуры

Таблица IPMI (**fan**) отображает информацию о вентиляторах. Данные получены через интерфейс IPMI (Intelligent Platform Management Interface):

Sensor	Идентификатор датчика вентилятора
RPM	Текущая скорость вращения вентилятора в оборотах в минуту (RPM)

Таблица IPMI (**voltage**) отображает параметры напряжения. Данные получены через интерфейс IPMI (Intelligent Platform Management Interface):

Sensor	Идентификатор датчика напряжения
Voltage	Текущее значение напряжения

Таблица IPMI (**power state**) отображает состояние питания устройств. Данные получены через интерфейс IPMI (Intelligent Platform Management Interface):

Sensor	Идентификатор датчика питания
State	Текущее состояние питания устройства (0 — выключено, 1 — включено).

Таблица IPMI (**sensor state**) отображает состояние всех остальных отслеживаемых датчиков, помимо перечисленных выше, в зависимости от типа оборудования. Данные получены через интерфейс IPMI (Intelligent Platform Management Interface):

Sensor	Идентификатор датчика
State	Текущее значение датчика (0 или 1).

9.3 Мониторинг и статистика в Console Light

Console Light — это консоль для мониторинга активности в реальном времени, отображающая ключевые показатели нагрузки и производительности сервера. Консоль основана на возможностях API-интерфейса Angie ADC; данные мониторинга активности генерируются в реальном времени.

Пример развернутой и настроенной консоли: <https://console.angie.software/>

При использовании в составе продукта Angie ADC консоль Console Light устанавливается и настраивается автоматически.

9.3.1 Переход в Console Light

1. Откройте веб-интерфейс Angie ADC. В панели навигации перейдите в раздел **Мониторинг** → **Балансировщик**.

Откроется экран со статистикой для этого балансировщика.

2. В верхней части экрана нажмите кнопку **Перейти к интерактивному мониторингу**.

Откроется облегченная консоль Console Light с данными мониторинга активности в реальном времени для этого балансировщика.

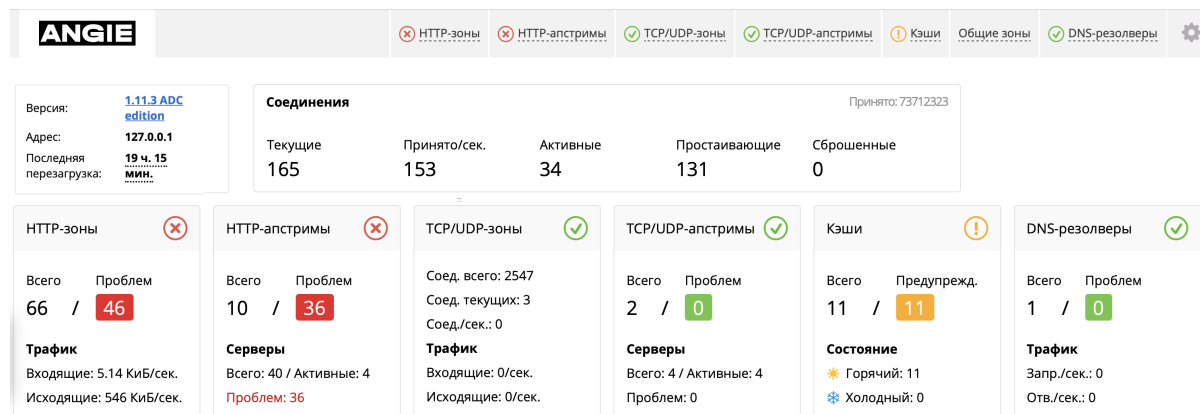
9.3.2 Интерфейс

Console Light представляет собой единый экран с набором вкладок, каждая из которых содержит ряд виджетов с данными мониторинга.

Совет

В разделах ниже описания элементов интерфейса даны в порядке слева направо.

9.3.3 Вкладка "Angie"



Это основная вкладка, где в сводном виде отображаются основные показатели мониторинга Angie ADC, сведенные на основе данных из нескольких разделов API.

Примечание

Виджеты со статистикой отображаются, если настроены соответствующие блоки в *конфигурации балансировщика нагрузки*.

9.3.3.1 Виджет "<версия>"

Здесь выводится номер версии Angie ADC, а также адрес сервера и время последней перезагрузки конфигурации.

9.3.3.2 Виджет "Соединения"

Здесь представлена основная статистика серверных соединений, формируемая на основе раздела API /status/connections/:

Текущие	Текущее количество соединений
Принято/сек.	Число принимаемых за секунду соединений
Активные	Число активных соединений
Простаивающие	Число соединений в состоянии ожидания
Сброшенные	Количество сброшенных соединений

Также доступно:

Принято	Общее число соединений, принятых с последней перезагрузки сервера
---------	---

9.3.3.3 Виджет "HTTP-зоны"

⚠ Предупреждение

Требует задать директиву `status_zone` в контексте `server` или `location`.

Здесь представлена статистика зон разделяемой памяти контекста `http`, формируемая на основе раздела API `/status/http/server_zones/`:

Всего	Общее количество зон
Проблем	Количество зон с какими-либо проблемами
Трафик	Скорость входящего и исходящего трафика в байтах в секунду

9.3.3.4 Виджет "HTTP-апстримы"

⚠ Предупреждение

Требует задать директиву `zone` в блоке `upstream` в контексте `http`.

Здесь представлена статистика апстримов контекста `http`, формируемая на основе раздела API `/status/http/upstreams/`:

Всего	Общее количество апстримов
Проблем	Количество апстримов с какими-либо проблемами
Серверы	Статистика серверов с разделением по состоянию

9.3.3.5 Виджет "TCP/UDP-зоны"

⚠ Предупреждение

Требует задать следующие директивы:

- `status_zone` в контексте `server` или `stream`;
- `limit_conn` в контексте `server` или `stream`;
- `limit_conn_zone` в контексте `stream`.

Пример:

```
stream {
    # ...
    limit_conn_zone $connection zone=limit-conn-stream:10m;

    server {
        # ...
        limit_conn limit-conn-stream 1;
        status_zone foo;
    }
}
```

Здесь представлена статистика зон разделяемой памяти контекста `stream`, формируемая на основе раздела API `/status/stream/server_zones/`:

Соед. всего	Общее количество клиентских соединений
Соед. текущих	Текущее количество клиентских соединений
Соед./сек.	Количество обрабатываемых в секунду соединений

9.3.3.6 Виджет "TCP/UDP-апстрымы"

⚠ Предупреждение

Требуется задать директиву `zone` в блоке `upstream` в контексте `stream`.

Здесь представлена статистика апстримов контекста `stream`, формируемая на основе раздела API `/status/stream/upstreams/`:

Всего	Общее количество апстримов
Проблем	Количество апстримов с какими-либо проблемами
Серверы	Статистика серверов с разделением по состоянию

9.3.4 Вкладка "HTTP-зоны"

⚠ Предупреждение

Требуется задать директиву `status_zone` в контексте `server` или `location`.

9.3.4.1 Раздел "Серверные зоны"

ANGIE ✓ HTTP-зоны ✓ HTTP-апстрымы ✓ TCP/UDP-зоны ✓ TCP/UDP-апстрымы ⚠ Кэши Общие зоны ✓ DNS-резолверы ⚙															
Серверные зоны															
Зона	Запросы			Ответы					Трафик				SSL		
	Текущие	Всего	Запр./сек.	1xx	2xx	3xx	4xx	5xx	Всего	Отпр./сек.	Получ./сек.	Отправлено	Получено	Рукопожатий	Повторных использований
Russia	0	0	0				NaN			0	0	0	0	0	0
India	0	0	0				NaN			0	0	0	0	0	0
China	0	0	0				NaN			0	0	0	0	0	0
South Africa	0	0	0				NaN			0	0	0	0	0	0
Argentina	0	0	0				NaN			0	0	0	0	0	0
Egypt	0	0	0				NaN			0	0	0	0	0	0
Ethiopia	0	0	0				NaN			0	0	0	0	0	0
Iran	0	0	0				NaN			0	0	0	0	0	0
Saudi Arabia	0	0	0				NaN			0	0	0	0	0	0
UAE	0	0	0				NaN			0	0	0	0	0	0

Здесь в сводном виде отображается статистика мониторинга зон разделяемой памяти для контекста `server` в `http`, формируемая на основе раздела API `/status/http/server_zones/`. Для каждой зоны представлены следующие данные:

Зона	Имя зоны
Совет Щелкните стрелку рядом с пунктом Зона, чтобы отсортировать зоны по алфавиту или порядку в конфигурации.	
Запросы	Общее количество запросов, а также число запросов в секунду
Ответы	Количество ответов с разбиением по кодам статуса, а также их общее количество
Трафик	Скорость исходящего и входящего трафика в байтах в секунду, а также общие объемы исходящего и входящего трафика в байтах
SSL	Суммарные показатели количества: успешных SSL-рукопожатий; повторных использований SSL-сессий; SSL-рукопожатий с истекшим таймаутом; неуспешных SSL-рукопожатий

9.3.4.2 Раздел "Зоны путей (Location)"

Зоны путей (Location)

Зона	Запросы		Ответы						Трафик			
	Всего	Запр./сек.	1xx	2xx	3xx	4xx	5xx	Всего	Отпр./сек.	Получ./сек.	Отправлено	Получено
Brasilia 🇧🇷	0	0				NaN			0	0	0	0
Diadema 🇧🇷	0	0				NaN			0	0	0	0
Porto Alegre 🇧🇷	0	0				NaN			0	0	0	0
Salvador 🇧🇷	0	0				NaN			0	0	0	0

Здесь в сводном виде отображается статистика мониторинга зон разделяемой памяти для контекста `location` в `http`, формируемая на основе раздела `API /status/http/location_zones/`. Для каждой зоны представлены следующие данные:

Зона	Имя зоны
Совет Щелкните стрелку рядом с пунктом Зона, чтобы отсортировать зоны по алфавиту или порядку в конфигурации.	
Запросы	Общее количество запросов, а также число запросов в секунду
Ответы	Количество ответов с разбиением по кодам статуса, а также их общее количество
Трафик	Скорость исходящего и входящего трафика в байтах в секунду, а также общие объемы исходящего и входящего трафика в байтах

9.3.4.3 Раздел "Зоны ограничения соединений (Limit Conn)"

Зоны ограничения соединений (Limit Conn)

Зона	Передано	Отклонено	Сброшено	Пропущено
limit-conn-http	0	0	0	0

Здесь приведена статистика зон `limit_conn` в контексте `http`, формируемая на основе раздела `API /status/http/limit_conns/`. Для каждой зоны представлены следующие данные:

Зона	Имя зоны
Щелкните значок рядом с пунктом Зона, чтобы открыть или закрыть график со следующими показателями.	
Передано	Общее количество проксированных соединений
Отклонено	Общее количество сброшенных соединений
Сброшено	Общее количество соединений, сброшенных из-за переполнения хранилища зоны
Пропущено	Общее количество соединений, переданных с нулевым или превосходящим 255 байт ключом

9.3.4.4 Раздел "Зоны ограничения запросов (Limit Req)"

Зоны ограничения запросов (Limit Req)

Зона	Передано	Задержано	Отклонено	Сброшено	Пропущено
limit-req-http	0	0	0	0	0

Здесь приведена статистика зон `limit_reqs` в контексте `http`, формируемая на основе раздела API `/status/http/limit_reqs/`. Для каждой зоны представлены следующие данные:

Зона	Имя зоны
Щелкните значок рядом с пунктом Зона, чтобы открыть или закрыть график со следующими показателями.	
Передано	Общее количество проксированных соединений
Задержано	Общее количество задержанных соединений
Отклонено	Общее количество сброшенных соединений
Сброшено	Общее количество соединений, сброшенных из-за переполнения хранилища зоны
Пропущено	Общее количество соединений, переданных с нулевым или превосходящим 255 байт ключом

9.3.5 Вкладка "HTTP-апстрымы"

ANGIE

HTTP-зоны

HTTP-апстрымы

TCP/UDP-зоны

TCP/UDP-апстрымы

Кэши

Общие зоны

DNS-резолверы

HTTP-апстрымы

Показать список апстрымов

Только проблемные

black

Загрузка памяти: 15 %

Показать все

Сервер	Запросы			Ответы	Соединения		Трафик		Проверки сервера		Проверки работоспособности		Время ответа							
Имя	Простой	Вес	Всего	Запр./сек.	...	4xx	5xx	Актив.	Отгр.	Отгр./сек.	Получ./сек.	Отправлено	Получено	Ошибок	Недоступно	Проверок	Ошибок	Последняя	Заголовки	Ответ
10.19.127.1:80 10.19.127.1	16.95 сек.	1	106985	4		403	495	3	10	353 Б	136 КиБ	7.88 МиБ	3.58 ГиБ	408	0	6	1	9 ч. 54 мин.	457 мс.	457 мс.
10.19.127.2:80 10.19.127.2	0 мс.	1	107149	3		399	485	3	10	307 Б	110 КиБ	7.89 МиБ	3.59 ГиБ	374	0	3	0	9 ч. 55 мин.	525 мс.	525 мс.

Предупреждение

Требует задать директиву `zone` в блоке `upstream` в контексте `http`.

На этой вкладке в сводном виде отображается статистика мониторинга апстримов контекста `http`, формируемая на основе раздела `API /status/http/upstreams/`. В режиме отладки также отображается процент загрузки памяти.

- Кнопка **Показать список апстримов** переключает краткий список апстримов с указанием числа проблемных апстримов и пиров.
- Переключатель **Только проблемные** переключает режим вывода статистики по проблемным апстримам.
- Раскрывающийся список с правой стороны таблицы каждого апстрима позволяет отфильтровать серверы в определенном состоянии (**Активные**, **Проблемные**, **На проверке**, **Выключенные**, **Занятые**, **Режим ожидания**, **Восстанавливается**, **Разгружаемые**).

Для каждого апстрима, помимо имени и загрузки зоны разделяемой памяти, представлены следующие данные:

Сервер	Имена, общее время простоя и веса серверов апстрима
💡 Совет Щелкните стрелку рядом с пунктом Сервер, чтобы отсортировать серверы по их состоянию или порядку в конфигурации.	
Запросы	Общее количество и скорость обработки запросов
Ответы	Количество ответов с разбиением по кодам статуса
Соединения	Количество активных соединений и их максимальный предел, если он задан
Трафик	Скорость исходящего и входящего трафика в байтах в секунду, а также общие объемы исходящего и входящего трафика в байтах
Проверки сервера	Количество неуспешных обращений к серверу и число раз, когда сервер считался недоступным (объект <code>health</code> в API)
Проверки работоспособности	Общее количество проверок сервера, количество неуспешных проверок, а также время последней проверки
Время ответа	Время от начала запроса до отправки первого байта ответа; общее время от начала запроса до завершения отправки всего ответа (объект <code>health</code> в API)

9.3.6 Вкладка "TCP/UDP-зоны"

⚠ Предупреждение

Требует задать следующие директивы:

- `status_zone` в контексте `server` или `stream`;
- `limit_conn` в контексте `server` или `stream`;
- `limit_conn_zone` в контексте `stream`.

Пример:

```
stream {
    # ...
    limit_conn_zone $connection zone=limit-conn-stream:10m;

    server {
        # ...
        limit_conn limit-conn-stream 1;
        status_zone foo;
    }
}
```

9.3.6.1 Раздел "TCP/UDP-зоны"

TCP/UDP-зоны

Зона	Соединения			Сессии				Трафик				SSL		
	Текущих	Всего	Соед./сек.	2xx	4xx	5xx	Всего	Отпр./сек.	Получ./сек.	Отправлено	Получено	Рукопожатий	Неудачных рукопожатий	Повторных использований
sing_chorus	3	403	0	372	0	28	400	0	0	4.41 Гб	6.51 Мб	375	0	180

Здесь в сводном виде отображается статистика мониторинга зон разделяемой памяти контекста **server** в **stream**, формируемая на основе раздела API `/status/stream/server_zones/`. Для каждой зоны представлены следующие данные:

Зона	Имя зоны
Соединения	Текущее и общее количество соединений, а также число соединений в секунду
Сессии	Количество сессий с разбиением по кодам статуса, а также их общее число
Трафик	Скорость исходящего и входящего трафика в байтах в секунду, а также общие объемы исходящего и входящего трафика в байтах
SSL	Суммарные показатели количества: успешных SSL-рукопожатий; неуспешных SSL-рукопожатий; повторных использований SSL-сессий

9.3.6.2 Раздел "Зоны ограничения соединений (Limit Conn)"

Зоны ограничения соединений (Limit Conn)

Зона	Передано	Отклонено	Сброшено	Пропущено
limit-conn-stream	403	0	0	0

Здесь приведена статистика зон **limit_conn** в контексте **stream**, формируемая на основе раздела API `/status/stream/limit_conns/`. Для каждой зоны представлены следующие данные:

Зона	Имя зоны
Щелкните значок рядом с пунктом Зона, чтобы открыть или закрыть график со следующими показателями.	
Передано	Общее количество проксированных соединений
Отклонено	Общее количество сброшенных соединений
Сброшено	Общее количество соединений, сброшенных из-за переполнения хранилища зоны
Пропущено	Общее количество соединений, переданных с нулевым или превосходящим 255 байт ключом

9.3.7 Вкладка "TCP/UDP-апстримы"

✗ HTTP-зоны
✗ HTTP-апстримы
✓ TCP/UDP-зоны
✓ TCP/UDP-апстримы
⚠ Кэши
Общие зоны
✓ DNS-резолверы

TCP/UDP-апстримы
Показать список апстримов
Только проблемные

upstream-arioso
Загрузка памяти: 0 %
Показать все

Сервер	Соединения					Трафик				Проверки сервера		Проверки работоспособности		Время ответа					
	Имя	Простой	Вес	Всего	Соед./сек.	Активных	Ограниченных	Отпр./сек.	Получ./сек.	Отправлено	Получено	Ошибок	Недоступно	Проверок	Ошибок	Последняя	Соединение	Первый байт	Ответ
	10.19.127.1:80	11 мин.	1	36	0	2	∞	0	6.74 КиБ	1.07 МиБ	742 МиБ	0	0	5848	62	1 ч, 4 мин.	4 мс.	1.95 сек.	23 мин.
	10.19.127.2:80	11 мин.	1	34	0	1	∞	44.0 Б	20.8 КиБ	1.08 МиБ	748 МиБ	0	0	5858	60	1 ч, 4 мин.	4 мс.	1.92 сек.	23 мин.

⚠ Предупреждение

Требует задать директиву `zone` в блоке `upstream` в контексте `stream`.

На этой вкладке в сводном виде отображается статистика мониторинга апстримов контекста `stream`, формируемая на основе раздела API `/status/stream/upstreams/`. В режиме отладки также отображается процент загрузки памяти.

- Кнопка **Показать список апстримов** переключает отображение краткого списка апстримов с указанием числа проблемных апстримов и пиров.
- Переключатель **Только проблемные** включает и отключает режим вывода статистики по проблемным апстримам.
- Раскрывающийся список с правой стороны таблицы каждого апстрима позволяет отфильтровать серверы в определенном состоянии (Активные, Проблемные, На проверке, Выключенные, Занятые, Режим ожидания, Восстанавливается, Разгружаемые).

Для каждого апстрима представлены следующие данные:

Сервер	Имена, общее время простоя и веса серверов апстрима
Совет Щелкните стрелку рядом с пунктом Сервер, чтобы отсортировать серверы по их состоянию или порядку в конфигурации.	
Соединения	Количество активных соединений и их максимальный предел, если он задан
Трафик	Скорость исходящего и входящего трафика в байтах в секунду, а также общие объемы исходящего и входящего трафика в байтах
Проверки сервера	Количество неуспешных обращений к серверу и число раз, когда сервер считался недоступным (объект health в API)
Проверки работоспособности	Общее количество проверок сервера, количество неуспешных проверок, а также время последней проверки
Время ответа	Время, затраченное на установку соединения с бэкендом; время от начала запроса до получения первого байта ответа; общее время, прошедшее от начала запроса до получения последнего байта ответа (объект health в API)

9.3.8 Вкладка "Кэши"

ANGIE

⊗ HTTP-зоны

⊗ HTTP-апстримы

✓ TCP/UDP-зоны

✓ TCP/UDP-апстримы

🕒 Кэши

Общие зоны

⊗ DNS-резолверы

⚙

Кэши

Зона	Состояние	Загрузка памяти	Макс. объем	Использовано	Использовано диска	Трафик			Коэффициент попадания
						Предоставлено	Записано	Мимо	
cache-argentina	☀	2 %	64.0 МиБ	32.6 МиБ	51 %	1.18 ГиБ	38.8 ГиБ	38.8 ГиБ	2 %
▲ cache-brazil	☀	2 %	-	-	-	1.07 ГиБ	37.6 ГиБ	37.6 ГиБ	2 %
Путь Состояние Макс. объем Использовано Использовано диска									
/var/cache/angie/proxy_cache/brazil-hot ☀ 64.0 МиБ 5.72 МиБ 9 %									
/var/cache/angie/proxy_cache/brazil-short ☀ 64.0 МиБ 11.7 МиБ 18 %									
/var/cache/angie/proxy_cache/brazil-long ☀ 64.0 МиБ 12.0 МиБ 19 %									
cache-china	☀	1 %	64.0 МиБ	10.0 МиБ	16 %	250 МиБ	18.0 ГиБ	18.0 ГиБ	1 %

Предупреждение

Требуется задать директиву `proxy_cache_path` в контексте `http`.

На этой вкладке в сводном виде отображается статистика мониторинга зон `proxy_cache` контекста `http`, формируемая на основе раздела API `/status/http/caches/`. Для каждой зоны представлены следующие данные:

Зона	Имя зоны
 Совет Щелкните значок рядом с пунктом Зона, чтобы открыть или закрыть списки <i>шардов</i> для всех зон, где они есть.	
Состояние	Состояние кэша: холодный (метаданные загружаются в память) или горячий (метаданные загружены)
Загрузка памяти	Коэффициент использования памяти
Макс. объем	Максимальный объем кэша на диске в байтах; знак бесконечности, если объем кэша не ограничен
Использовано	Текущий объем кэша на диске в байтах
Использовано диска	Коэффициент использования диска
Трафик	Объем трафика (в байтах), переданного из кэша, записанного в кэш и возвращенного в обход кэша
Коэффициент попадания	Коэффициент попадания в кэш (отношение переданного из кэша трафика к общему объему)

Для зоны с включенным *шардингом* значения Макс. объем, Использовано и Использовано диска отслеживаются отдельно для каждого шарда; в строке самой зоны выводятся прочерки, а зона обозначена как раскрывающийся список, в котором перечислены отдельные шарды:

Путь	Путь шарда на диске
Состояние	Состояние шарда: холодный (метаданные загружаются в память) или горячий (метаданные загружены)
Макс. объем	Максимальный объем шарда на диске в байтах; знак бесконечности, если объем шарда не ограничен
Использовано	Текущий объем шарда на диске в байтах
Использовано диска	Коэффициент использования диска

9.3.9 Вкладка "Общие зоны"

  HTTP-зоны  HTTP-апстримы  TCP/UDP-зоны  TCP/UDP-апстримы  Кэши  Общие зоны  DNS-резолверы 			
Общие зоны			
Зона	Всего страниц памяти	Использовано страниц памяти	Загрузка памяти
cache-argentina	2544	2	1 %
cache-brazil	2544	2	1 %
cache-china	2544	2	1 %
cache-egypt	2544	2	1 %
cache-ethiopia	2544	2	1 %
cache-india	2544	2	1 %
cache-iran	2544	2	1 %
cache-russia	2544	2	1 %
cache-saudi-arabia	2544	2	1 %
cache-south-africa	2544	2	1 %

На этой вкладке в сводном виде отображается статистика мониторинга **всех** зон разделяемой памяти для всех контекстов. Для каждой зоны представлены следующие данные:

Зона	Имя зоны
Совет Щелкните стрелку рядом с пунктом Зона, чтобы отсортировать зоны по размеру или порядку в конфигурации.	
Всего страниц памяти	Общее количество страниц памяти
Использовано страниц памяти	Количество используемых страниц памяти
Загрузка памяти	Коэффициент использования памяти для зоны

9.3.10 Вкладка "DNS-резолверы"

DNS-резолверы

Зона	Запросы				Ответы						
	A, AAAA	SRV	PTR		Успешные	Ошибок формата	Сервер не завершил запрос	Ошибок имени	Запрос не поддерживается	Запрос отклонен	Неизвестных ошибок
resolver	72021	0	0		54017	0	0	18004	0	0	0

Предупреждение

Требуется задать директиву `resolver` в контексте `http`.

На этой вкладке в сводном виде отображается статистика запросов в зонах разделяемой памяти DNS, формируемая на основе раздела `API /status/resolvers/`. Для каждой зоны представлены следующие данные:

Зона	Имя зоны
Совет Щелкните стрелку рядом с пунктом Зона, чтобы отсортировать зоны по состоянию или порядку в конфигурации.	
Запросы	Количество запросов типа A и AAAA, SRV, PTR
Ответы	Количество ответов с разделением по соответствующим кодам (Успешные, Ошибок формата, Сервер не завершил запрос, Ошибок имени, Запрос не поддерживается, Запрос отклонен и прочих)

9.3.11 Виджет "Настройки"

Настройки

Обновлять каждые сек.

Пороговое значение ошибок для 4xx %

Вычислять коэффициент попадания в кэш за последние сек.

Пороговое значение ошибок DNS %

Язык русский

v1.8.0

Позволяет настроить общие параметры консоли:

- Частоту обновления данных. Значение по умолчанию — 1 сек.
- Пороговое соотношение статусов 4xx. По достижении порога в соответствующих разделах, посвященных ответам сервера, появляются "желтые" предупреждения. Значение по умолчанию — 7%.
- Временное окно для вычисления соотношения успешных попаданий в кэш. Значение по умолчанию — 300 сек.
- Порог учета ошибок для резолвера. По достижении указанного порога резолвер станет "красным". Значение по умолчанию — 3%.
- Язык интерфейса консоли. Доступные варианты: английский и русский. По умолчанию язык консоли выбирается на основе локали, установленной в браузере.

9.3.12 Панель управления консолью

На всех вкладках в середине левой части страницы есть выдвигающаяся панель с двумя кнопками . Верхняя приостанавливает и вновь запускает обновление данных из API, а нижняя позволяет обновить данные вручную, когда обновление приостановлено.

9.4 Экспорт метрик

Экспорт метрик из Angie ADC во внешний Prometheus выполняется с помощью механизма *federation*. Для экспорта доступны:

- системные метрики Node Exporter;
- метрики Angie ADC.

9.4.1 Экспорт метрик Node Exporter

Чтобы настроить сбор метрик от Node Exporter, необходимо добавить в конфигурационный файл внешнего Prometheus (блок `scrape_configs`) следующую конфигурацию:

```
scrape_configs:
- job_name: adc_node
  honor_labels: false
  metrics_path: /federate
  params:
    match[]:
      - '{job="node"}'
  static_configs:
    - targets:
      - <адрес_ADC-инстанса1>:8080
      - <адрес_ADC-инстанса2>:8080
```

После обновления конфигурации необходимо перезапустить сервис Prometheus.

Для визуализации собранных данных можно использовать готовый дашборд Grafana: <https://grafana.com/grafana/dashboards/1860-node-exporter-full/>.

9.4.2 Экспорт метрик балансировщика нагрузки

Чтобы настроить сбор метрик от балансировщика нагрузки, необходимо добавить в конфигурационный файл Prometheus (блок `scrape_configs`) следующую конфигурацию:

```
scrape_configs:
- job_name: adc_angie
```

```
honor_labels: false
metrics_path: /federate
params:
  match[]:
    - '{job="local-lb"}'
static_configs:
  - targets:
    - <адрес_ADC-инстанса1>:8080
    - <адрес_ADC-инстанса2>:8080
```

После обновления конфигурации необходимо перезапустить сервис Prometheus.

Для визуализации собранных данных можно использовать готовый дашборд Grafana: <https://grafana.com/grafana/dashboards/20719-angie-dashboard/>.

9.4.3 Экспорт системных метрик Angie ADC

```
scrape_configs:
  - job_name: hardware
    honor_labels: false
    metrics_path: /federate
    params:
      match[]:
        - '{job="node"}'
        - '{job="smartctl"}'
        - '{job="ipmi"}'
    static_configs:
      - targets:
        - <адрес_ADC-инстанса1>:8080
        - <адрес_ADC-инстанса2>:8080
```

Подробнее о системных метриках см. *Мониторинг системы*.

9.5 SNMP

Angie ADC может встраиваться в инфраструктуру мониторинга корпоративного уровня (например Zabbix, Nagios). Поддерживается отправка стандартных метрик операционной системы и собственных метрик Angie ADC (см. ниже) по протоколам SNMPv2c и SNMPv3 (UDP-порт 161). Протоколы SNMPv2c и SNMPv3 могут работать одновременно. При использовании протокола SNMPv3 необходимо предварительно добавить *пользователя SNMP* в веб-интерфейсе Angie ADC, чтобы предоставить безопасный доступ для системы мониторинга к SNMP-метрикам. Настройки пользователей распространяются на все узлы *кластера*.

Протокол	SNMPv2c	SNMPv3
Передача данных	в открытом виде	в зашифрованном виде
Аутентификация трафика	не требуется	требуется

Примечание

Для интеграции с системами мониторинга предоставляется MIB-файл, который можно скачать в веб-интерфейсе Angie ADC (Мониторинг → Система → Скачать SNMP MIB-файл в верхней правой части экрана).

9.5.1 Пользователи SNMP

Примечание

Перед настройкой пользователей SNMP необходимо создать кластер Angie ADC. Подробнее см. в статье [Создание кластера](#).

При использовании протокола SNMPv3 необходимо создать пользователей SNMP (т.е. те системы мониторинга, которые будут иметь доступ к данным).

angie-adc1 > Пользователи SNMP

Пользователи SNMP + Добавить пользователя SNMP

Имя пользователя	Протокол аутентификации	Протокол шифрования	Дата создания	
first-user	SHA-256	AES-256	30.07.2026 18:46:21	 
second-user	SHA-256	AES-128	30.07.2026 18:46:38	 
third-user	SHA-256	AES-256	30.07.2026 18:46:51	 

Примечание

Изменение списка пользователей SNMPv3 вызывает кратковременный перезапуск службы snmpd (1-2 секунды) на узлах кластера.

Добавление пользователя SNMP:

1. В веб-интерфейсе Angie ADC перейдите в раздел **Мониторинг** → **Пользователи SNMP**.
2. Нажмите **Добавить пользователя SNMP**.
3. В открывшемся окне заполните поля формы для добавления нового пользователя:
 - **Имя пользователя** — уникальный идентификатор пользователя SNMPv3, например `zabbix-user`.
 - **Секрет аутентификации** — зашифрованный секрет аутентификации (AES-256-GCM, Base64).
 - **Протокол шифрования** — поддерживаются AES128 и AES256.
 - **Секрет шифрования** — зашифрованный секрет шифрования (AES-256-GCM, Base64).
4. Нажмите **Добавить**.

Просмотр пользователей SNMP:

1. В веб-интерфейсе Angie ADC перейдите в раздел **Мониторинг** → **Пользователи SNMP**.
2. В открывшемся окне отображаются добавленные пользователи SNMP.

Для каждого пользователя отображается следующая информация:

- **Имя пользователя** — уникальный идентификатор пользователя SNMPv3.
- **Протокол аутентификации** — всегда SHA256.
- **Протокол шифрования** — поддерживаются AES128 и AES256.
- **Дата создания** — Дата и время создания записи.

Редактирование и удаление пользователей SNMP:

1. В веб-интерфейсе Angie ADC перейдите в раздел **Мониторинг** → **Пользователи SNMP**.
2. В открывшемся окне отображаются добавленные пользователи SNMP.

3. Нажмите значок редактирования / удаления напротив нужного пользователя.

9.5.2 Метрики операционной системы

Поддерживаются все системные метрики Net-SNMP.

Пример запроса объема доступной физической памяти:

```
snmpget -v2c -c public 127.0.0.1 1.3.6.1.4.1.2021.4.6.0
```

Пример ответа:

```
UCD-SNMP-MIB::memAvailReal.0 = INTEGER: 743028 kB
```

Ниже приведены примеры самых распространенных метрик, которые можно отслеживать:

Метрика	OID	Тип	Описание
diskIOLoad.1	.1.3.6.1.4.1.2021.13.1	Inte	Процент времени, когда диск был занят выполнением операций ввода-вывода.
diskIOReads.1	.1.3.6.1.4.1.2021.13.1	Cou	Количество операций чтения с диска.
diskIOWrites.1	.1.3.6.1.4.1.2021.13.1	Cou	Количество операций записи на диск.
dskErrorFlag.1	.1.3.6.1.4.1.2021.9.1.1	Inte	Ошибка диска: становится 1, если место на диске заканчивается.
dskPercent.1	.1.3.6.1.4.1.2021.9.1.1	Inte	Процент занятого места на основном разделе.
ifInDiscards.X	.1.3.6.1.2.1.2.2.1.13.X	Cou	Количество входящих пакетов, отброшенных из-за переполнения буферов.
ifInErrors.X	.1.3.6.1.2.1.2.2.1.14.X	Cou	Ошибки сети. Указывает на входящие пакеты с ошибками или проблемы на физическом уровне.
ifInOctets.X	.1.3.6.1.2.1.2.2.1.10.X	Cou	Входящий трафик. Объем полученных данных на сетевом интерфейсе.
ifOutOctets.X	.1.3.6.1.2.1.2.2.1.16.X	Cou	Исходящий трафик. Объем переданных данных через сетевой интерфейс.
hrSystemProcesses.0	.1.3.6.1.2.1.25.1.6.0	Gau	Общее количество запущенных процессов. Резкий рост может указывать на зависшие скрипты.
hrSystemUptime.0	.1.3.6.1.2.1.25.1.1.0	Tick	Время непрерывной работы сервера. Используется для обнаружения перезагрузки сервера.
laLoad.1	.1.3.6.1.4.1.2021.10.1	Stri	Средняя нагрузка (load average) на CPU за одну минуту.
memAvailReal.0	.1.3.6.1.4.1.2021.4.6.0	Inte	Объем доступной физической памяти.
memCached.0	.1.3.6.1.4.1.2021.4.15	Inte	Память, используемая файловым кэшем для ускорения операций чтения.
memTotalFreeSwap.0	.1.3.6.1.4.1.2021.4.4.0	Inte	Свободный Swap. При снижении до 0 возможен запуск OOM Killer.
ssCpuIdle.0	.1.3.6.1.4.1.2021.11.1	Inte	Процент свободного ресурса. Близкое к 0 значение указывает на перегрузку CPU.
ssCpuRawWait.0	.1.3.6.1.4.1.2021.11.5	Cou	Время ожидания ввода-вывода. Рост значения указывает на проблемы с дисковой подсистемой или БД.

9.5.3 Собственные метрики Angie ADC

Angie ADC позволяет получать по SNMP собственные метрики из Prometheus. У собственных метрик будет базовый OID .1.3.6.1.4.1.2021.50.

Пример запроса метрики базовой доступности сервиса up:

```
snmpget -v2c -c public 127.0.0.1 1.3.6.1.4.1.2021.50.2.1
```

Ответ:

```
UCD-SNMP-MIB::ucdavis.50.2.1 = Gauge32: 1
```

Ответ 1 означает, что сервис доступен.

Пример запроса всего дерева собственных метрик:

```
snmpwalk -v2c -c public 127.0.0.1 1.3.6.1.4.1.2021.50
```

Если Prometheus не доступен или метрик нет, в ответе может быть **NONE** или ошибка.

Ниже приведены метрики Prometheus, поддерживаемые по SNMP на данный момент. Список поддерживаемых метрик будет пополняться.

Метрика Prometheus	из	OID	Тип	Описание
angie_connections_		.1.3.6.1.4.1.2021.50.1	Cou	Принятые соединения — мониторинг общего потока входящих подключений.
angie_connections_		.1.3.6.1.4.1.2021.50.1	Gau	Активные сессии — мониторинг количества пользователей на сайте в данную секунду.
angie_connections_		.1.3.6.1.4.1.2021.50.1	Cou	Сброшенные соединения — рост этого показателя указывает на достижение лимитов сервера.
angie_http_request:		.1.3.6.1.4.1.2021.50.1	Cou	Всего HTTP-запросов — позволяет рассчитать RPS (количество запросов в секунду).
go_goroutines		.1.3.6.1.4.1.2021.50.2	Gau	Потоки — мониторинг здоровья Go-приложения (защита от утечек памяти).
go_memstats_alloc		.1.3.6.1.4.1.2021.50.2	Gau	Память процесса — позволяет отслеживать, сколько оперативной памяти выделено и используется Go-процессом сейчас.
scrape_duration_se		.1.3.6.1.4.1.2021.50.2	Gau	Время сбора данных в секундах — позволяет оценить производительность.
scrape_samples_scr		.1.3.6.1.4.1.2021.50.2	Gau	Количество полученных строк метрик. Резкое падение показателя может означать, что часть метрик потеряна.
up		.1.3.6.1.4.1.2021.50.2	Gau	Доступность — базовая health-метрика, 1 — сервис доступен, 0 — недоступен.

глава 10

Администрирование и безопасность

В этом разделе описываются общие настройки системы Angie ADC через веб-интерфейс, а также управление через CLI.

В этом разделе:

- *Настройка системы*
- *Управление сертификатами*
- *Журналы событий и экспорт*
- *Аудит действий пользователей*
- *Диагностика сети*
- *Резервное копирование и восстановление системы*
- *Парольная политика*
- *Настройка учетной записи*
- *Справочник команд (CLI)*

10.1 Настройка системы

Здесь описываются общие настройки системы Angie ADC.

В этом разделе:

- *Настройка конфигурации*
- *Настройка HTTPS*
- *Управление пользователями*
- *Аутентификация*
- *Авторизация*
- *Настройка NTP*

10.1.1 Аутентификация

10.1.1.1 Типы аутентификации

Angie ADC поддерживает два типа аутентификации пользователей:

- *Локальная аутентификация.*

При использовании локальной аутентификации данные о пользователях хранятся в системе. Вы можете просмотреть их в веб-интерфейсе (**Система** → **Пользователи**). Для входа в веб-интерфейс пользователю необходимо выбрать локальный тип аутентификации и ввести свои логин и пароль.

- *Внешняя аутентификация.*

При использовании внешней аутентификации данные о пользователях хранятся на внешнем сервере аутентификации. У системы нет доступа к данным пользователя. Для включения внешней аутентификации необходимо добавить в веб-интерфейс предварительно настроенный сервер аутентификации. Поддерживаются RADIUS, TACACS и LDAP. После этого при входе в веб-интерфейс будет доступен выбор аутентификации через заданный сервер.

Предупреждение

Необходимо также включить и настроить *авторизацию* для внешних пользователей. Если авторизация не настроена, пользователи не смогут получить доступ к системе.

10.1.1.2 Включение внешней аутентификации

1. Откройте веб-интерфейс Angie ADC. В панели навигации перейдите в раздел **Система** → **Аутентификация и авторизация**.

Откроется страница с настройками аутентификации и авторизации для внешних пользователей.

2. Включите переключатель **Включить внешнюю аутентификацию** и выберите провайдера аутентификации.

Примечание

Одновременно поддерживается использование только одного провайдера внешней аутентификации. При добавлении нового провайдера аутентификации настройки предыдущего провайдера сбрасываются.

3. Заполните нужные поля.

Для RADIUS-аутентификации укажите следующие параметры сервера:

- **Сервер:** IP-адрес или доменное имя RADIUS-сервера.
- **Порт:** порт подключения к RADIUS-серверу (по умолчанию 1812).
- **Протокол:** протокол передачи данных: UDP или TCP (по умолчанию: UDP).
- **NAS-идентификатор:** идентификатор NAS (Network Access Server), передаваемый серверу RADIUS.
- **Метод аутентификации:** поддерживается только PAP.
- **Секрет:** общий секрет для шифрования паролей.

Для TACACS-аутентификации укажите следующие параметры сервера:

- **Адрес:** IP-адрес или доменное имя TACACS-сервера.
- **Порт:** порт подключения к TACACS-серверу (по умолчанию 49).

- **Секрет:** общий секрет для шифрования паролей.

Для LDAP-аутентификации укажите параметры LDAP-сервера, сервисной учетной записи и параметры поиска пользователей:

- **Сервер:** IP-адрес или доменное имя LDAP-сервера.
- **Порт:** порт подключения к LDAP-серверу (по умолчанию 389).
- **Режим защиты соединения** (по умолчанию Без TLS, также доступны варианты LDAPS и StartTLS).
- **DN сервисной учетной записи:** логин сервисной учетной записи, от имени которой система подключается к LDAP.
- **Пароль сервисной учетной записи:** пароль сервисной учетной записи для подключения к LDAP-серверу.
- **Базовый DN пользователя:** задает ветку на LDAP-сервере, в которой система будет искать пользователя, например `ou=users,dc=example,dc=com`.
- **Идентификатор логина:** поле в LDAP, которое содержит логин, например `uid`, `sAMAccountName` или `userPrincipalName`.
- **Фильтр поиска:** LDAP-фильтр для поиска пользователя, например `(&(objectClass=user)(sAMAccountName=%s))`.

Примечание

Фильтр LDAP определяет, в каком формате пользователь должен вводить логин при входе в систему:

- если используется фильтр `(&(objectClass=user)(sAMAccountName=%s))`, то логин будет без доменного имени, например `ivanov`;
- если используется фильтр `(&(objectClass=user)(userPrincipalName=%s))`, то логин будет полным: `ivanov@domain.com`.

4. Нажмите **Сохранить**.

Внешняя аутентификация с использованием настроенного провайдера будет включена. При входе в веб-интерфейс пользователю будет необходимо выбрать нужный тип аутентификации и ввести свои логин и пароль. Если пользователь с таким логином и паролем будет найден на заданном сервере, то аутентификация пройдет успешно.

10.1.1.3 Изменение параметров внешней аутентификации

1. Откройте веб-интерфейс Angie ADC. В панели навигации перейдите в раздел **Система** → **Аутентификация и авторизация**.

Откроется страница с настройками аутентификации и авторизации для внешних пользователей.

2. Внесите необходимые изменения и нажмите **Сохранить**.

Параметры внешней аутентификации будут обновлены.

10.1.1.4 Отключение внешней аутентификации

1. Откройте веб-интерфейс Angie ADC. В панели навигации перейдите в раздел **Система** → **Аутентификация и авторизация**.

Откроется страница с настройками аутентификации и авторизации для внешних пользователей.

2. Переведите переключатель **Включить внешнюю аутентификацию** в положение "выключено".

Внешняя аутентификация будет отключена.

10.1.2 Авторизация

Angie ADC поддерживает ролевое разграничение доступа к функциям системы. Пользователям может быть назначена одна из следующих ролей:

- **Администратор:** обладает полными правами на все действия в системе;
- **Супервизор:** имеет ограниченные права, может только просматривать информацию в системе, например, проводить аудит;
- **Оператор:** имеет ограниченные права, может редактировать файлы конфигурации балансирующего, к которому ему предоставлен доступ, и просматривать статистику зон, к которому ему предоставлен доступ. Доступы предоставляет администратор. Остальные настройки Angie ADC оператору не доступны и не отображаются в веб-интерфейсе.

Поддерживается два типа авторизации пользователей:

- *Локальная авторизация.*

Роли для локальных пользователей назначаются при их создании и хранятся локально в системе. По умолчанию в Angie ADC предустановлен локальный системный пользователь **admin** с правами администратора.

- *Внешняя авторизация.*

Роли для внешних пользователей определяются через внешнего провайдера. Поддерживаются провайдеры *RADIUS*, *TACACS+* и *LDAP*.

Примечание

RADIUS-авторизация доступна только при аутентификации через RADIUS.

Система не хранит данные о ролях внешних пользователей. Принятые решения по внешней авторизации записываются в журналы событий системы. Записи можно экспортировать на внешний syslog-сервер для дальнейшего аудита.

Предупреждение

Необходимо также включить и настроить *аутентификацию* для внешних пользователей. Если аутентификация не настроена, пользователи не смогут войти в систему.

Совместимость внешних провайдеров аутентификации и авторизации:

Провайдер аутентификации	Совместимые провайдеры авторизации
RADIUS	RADIUS, LDAP, TACACS+
LDAP	LDAP, TACACS+
TACACS	LDAP, TACACS+

Примечание

Одновременно поддерживается использование только одного провайдера. При добавлении нового провайдера настройки предыдущего провайдера сбрасываются.

В этой статье:

- *LDAP-авторизация*
- *TACACS-авторизация*
- *Выключение внешней авторизации*

10.1.2.1 LDAP-авторизация

Роли внешних пользователей определяются на основе принадлежности к LDAP-группам.

Чтобы настроить LDAP-авторизацию для внешних пользователей, необходимо:

- включить внешнюю аутентификацию;
- включить внешнюю авторизацию и подключить LDAP-сервер;
- задать настройки сервисной учетной записи для подключения Angie ADC к LDAP-серверу;
- задать базовый DN и фильтры для поиска пользователей и групп на LDAP-сервере;
- настроить соответствие LDAP-групп и ролей пользователей в Angie ADC.

i Примечание

На стороне LDAP-сервера также должны быть настроены пользователи, группы и сервисная учетная запись для Angie ADC.

Настройка LDAP-авторизации

1. Откройте веб-интерфейс Angie ADC. В панели навигации перейдите в раздел Система → Аутентификация и авторизация.
Откроется страница с настройками аутентификации и авторизации для внешних пользователей.
2. В блоке Внешняя авторизация переведите переключатель Включить внешнюю авторизацию в положение "включено" и выберите провайдера авторизации LDAP.
3. Задайте параметры подключения к серверу:
 - Сервер: IP-адрес или доменное имя LDAP-сервера.
 - Порт: порт подключения к LDAP-серверу (по умолчанию 389).
4. Укажите параметры защиты соединения: Без TLS (по умолчанию) или TLS (LDAPS)/startTLS, если необходимо использовать безопасное соединение при подключении к серверу.

i Примечание

При выборе режима защиты соединения LDAPS или StartTLS становится доступен флажок Проверять TLS-сертификат. По умолчанию флажок установлен.

5. Укажите DN учетной записи, с которой будет выполняться подключение к LDAP-серверу (DN сервисной учетной записи (bind)).
Можно использовать учетную запись администратора LDAP-сервера, например cn=admin, dc=example, dc=com.
6. Укажите пароль сервисной учетной записи (Пароль пользователя bind).
7. В блоке Поиск пользователя укажите следующие параметры:
 - Базовый DN пользователей, например ou=users, dc=example, dc=com;
 - Фильтр поиска пользователей, например (uid=%s).

8. В блоке Поиск групп укажите следующие параметры:

- Базовый DN групп, например `ou=groups,dc=example,dc=com`;
- Фильтр поиска групп, например `(member=%s)`.

9. В блоке Маппинг ролей LDAP нажмите **Добавить правило** и задайте соответствие LDAP-групп пользователей ролям в Angie ADC. Например, если на LDAP-сервере для администраторов создана группа `dn: cn=<group_name>,ou=groups,dc=example,dc=com`, то укажите значение `cn=<group_name>,ou=groups,dc=example,dc=com` в поле DN группы и выберите роль Администратор (admin).

10. Нажмите **Сохранить**.

Внешняя авторизация будет настроена и включена. При входе внешнего пользователя в систему на LDAP-сервере будет выполняться поиск групп, в которых он состоит. Если найденные группы совпадают с группами, указанными в маппинге, то пользователю будет назначена соответствующая роль. Если пользователь не найден ни в одной из групп или LDAP-группы недоступны или некорректны, доступ к системе будет запрещен.

10.1.2.2 TACACS-авторизация

Роли для внешних пользователей определяются на основе соответствия атрибутам этих пользователей в TACACS+.

Чтобы настроить TACACS-авторизацию для внешних пользователей, необходимо:

- включить внешнюю аутентификацию;
- включить внешнюю авторизацию и подключить TACACS-сервер;
- настроить соответствие атрибутов TACACS+ ролям пользователей в Angie ADC.

На стороне TACACS-сервера также должны быть настроены пользователи и их атрибуты.

Пример настройки:

```
group = admin {
    service = shell {
        set priv-lvl = 15
    }
}

user = admin {
    password = clear admin
    member = admin
}
```

Настройка TACACS-авторизации

1. Откройте веб-интерфейс Angie ADC. В панели навигации перейдите в раздел Система → Аутентификация и авторизация.

Откроется страница с настройками аутентификации и авторизации для внешних пользователей.

2. В блоке Внешняя авторизация переведите переключатель Включить внешнюю авторизацию в положение "включено" и выберите провайдера авторизации TACACS.

3. Задайте параметры подключения к серверу:

- Адрес: IP-адрес или доменное имя TACACS-сервера.
- Порт: порт подключения к TACACS-серверу (по умолчанию 49).
- Секрет: общий секрет для шифрования паролей.

4. Укажите в порядке приоритета атрибуты ролей пользователя в TACACS+, например `priv-lvl`, затем `role`. По этим атрибутам Angie ADC будет определять роль для пользователя.
5. Укажите, какой сервис и протокол будут использоваться в запросе на авторизацию (AUTHOR). Параметры зависят от выбранного сервера. Например, для TACACS+ рекомендуется указать `service=shell` и `protocol=ip`.
6. Задайте соответствие значений атрибутов TACACS+ ролям в Angie ADC, например `priv-lvl=15` → Администратор или `role=adc-admin` → Администратор.
7. Нажмите Сохранить.

Внешняя авторизация будет настроена и включена. При входе внешнего пользователя в систему на TACACS-сервер будет отправлен запрос на авторизацию. Если атрибуты и значения в ответе TACACS совпадают с указанными в маппинге, то пользователю будет назначена соответствующая роль в Angie ADC. Если полученные данные не совпадают с указанными в маппинге или TACACS-сервер недоступен, доступ к системе будет запрещен.

10.1.2.3 RADIUS-авторизация

Роли для внешних пользователей определяются на основе соответствия атрибутам этих пользователей в RADIUS.

Чтобы настроить RADIUS-авторизацию для внешних пользователей, необходимо:

- включить внешнюю аутентификацию через RADIUS;
- включить внешнюю авторизацию через RADIUS;
- настроить соответствие атрибутов RADIUS ролям пользователей в Angie ADC.

Допускается использование следующих атрибутов ролей:

- `Filter-Id` — роль пользователя.
- `Class` — группа, к которой относится пользователь.
- `Tunnel-Private-Group-ID` — VLAN ID (используется для VLAN-авторизации).
- `VSA` — Vendor-Specific в формате `vsa:<vendorID>:<vendorType>` (например `vsa:9:1`).

На стороне RADIUS-сервера также должны быть настроены пользователи и их атрибуты.

Пример:

```
user1 Cleartext-Password := "P@ssword"
  Filter-Id := "adc-admin",
  Service-Type := Administrative-User,
  Cisco-AVPair := "shell:priv-lvl=15",
  Reply-Message := "Welcome, Admin",
  Class := "AdminGroup",
  Tunnel-Type := VLAN,
  Tunnel-Medium-Type := IEEE-802,
  Tunnel-Private-Group-ID := "100"

user2 Cleartext-Password := "P@ssword"
  Filter-Id := "adc-readonly",
  Service-Type := NAS-Prompt-User,
  Cisco-AVPair := "shell:priv-lvl=7",
  Reply-Message := "Welcome, Supervisor",
  Class := "SupervisorGroup",
  Tunnel-Type := VLAN,
  Tunnel-Medium-Type := IEEE-802,
  Tunnel-Private-Group-ID := "200"
```

Настройка RADIUS-авторизации

1. Откройте веб-интерфейс Angie ADC. В панели навигации перейдите в раздел Система → Аутентификация и авторизация.
Откроется страница с настройками аутентификации и авторизации для внешних пользователей.
2. В блоке Внешняя авторизация переведите переключатель Включить внешнюю авторизацию в положение "включено" и выберите провайдера авторизации RADIUS.
3. Укажите атрибут роли пользователя в RADIUS, например Filter-Id, vsa:9:1 или другой используемый атрибут. По этому атрибуту Angie ADC будет определять роль для пользователя.
4. Задайте соответствие значений атрибутов RADIUS ролям в Angie ADC, например для Filter-Id укажите adc-admin → Администратор, а при использовании vsa:9:1 укажите shell:priv-lvl=15 → Администратор.
5. Нажмите Сохранить.

Внешняя авторизация будет настроена и включена. При входе внешнего пользователя в систему через RADIUS будут получены атрибуты его роли. Если атрибуты и значения совпадают с указанными в машинке, то пользователю будет назначена соответствующая роль в Angie ADC. Если данные не совпадают с указанными в машинке или RADIUS-сервер недоступен, доступ к системе будет запрещен.

10.1.2.4 Выключение внешней авторизации

1. Откройте веб-интерфейс Angie ADC. В панели навигации перейдите в раздел Система → Аутентификация и авторизация.
Откроется страница с настройками аутентификации и авторизации для внешних пользователей.
2. В блоке Внешняя авторизация переведите переключатель Включить внешнюю авторизацию в положение "выключено".

Внешняя авторизация будет отключена. При входе в систему для внешних пользователей будет отображаться сообщение, что доступ запрещен.

10.1.3 Настройка HTTPS

Angie ADC поддерживает загрузку сертификата управления (management) для защищенного подключения к веб-интерфейсу Angie ADC (HTTPS). По умолчанию используются порты 8080/8443, их можно *поменять*. Поддерживается загрузка и использование только одного сертификата управления (management). При загрузке нового сертификата предыдущий автоматически удаляется.

Примечание

Для работы с сертификатами через веб-интерфейс необходимо предварительно создать кластер Angie ADC. Подробнее см. в статье [Создание кластера](#).

10.1.3.1 Загрузка сертификата управления и включение защищенного соединения

1. В веб-интерфейсе Angie ADC перейдите в раздел Система → Конфигурация.
2. В блоке Управление HTTPS нажмите Загрузить сертификат и включить HTTPS.
3. В открывшемся окне введите данные сертификата и ключа в формате PEM.
4. Нажмите Загрузить.

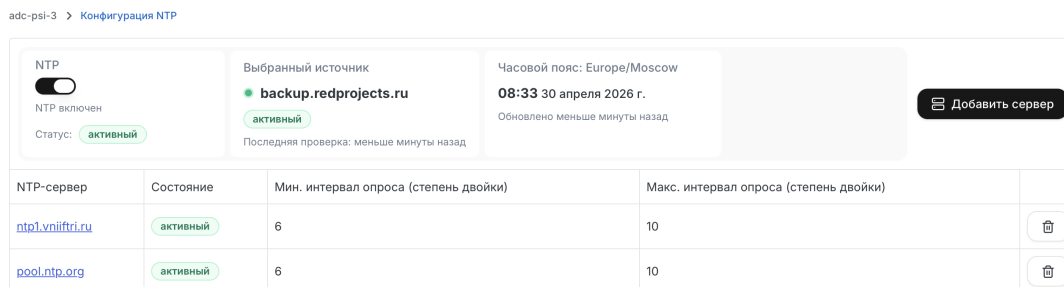
Сертификат будет загружен. В блоке Управление HTTPS будет отображаться статус HTTPS (выключен) и данные о сертификате: дата загрузки и срок действия.

- Нажмите **Включить HTTPS**, чтобы включить использование защищенного соединения. После этого веб-интерфейс автоматически откроется по HTTPS.

Сертификат управления и ключ можно также загружать *через CLI*.

10.1.4 Настройка NTP

NTP (Network Time Protocol) позволяет синхронизировать системное время на всех компонентах системы Angie ADC. Вы можете добавить один или несколько NTP-серверов, которые будут использоваться как источники времени.



В интерфейсе Angie ADC отображается статус службы NTP и состояние серверов (источников времени).

Статусы службы NTP:

- активный** — служба NTP включена.
- выключен** — служба NTP выключена.
- есть проблемы** — возникли проблемы с синхронизацией времени.
- не отвечает** — служба недоступна.
- данные устарели** — не удалось получить свежие данные.
- применение настроек...** — идет применение изменений.

У NTP-серверов могут быть следующие статусы:

- активный** — сервер доступен.
- есть проблемы** — возникли проблемы с синхронизацией времени.
- не отвечает** — сервер недоступен.
- неизвестно** — статус сервера не определен.
- применение настроек...** — идет применение изменений.

10.1.4.1 Настройка конфигурации NTP

- В веб-интерфейсе Angie ADC перейдите **Система** → **NTP**.
Откроется окно **Конфигурация NTP**.
- Включите переключатель **NTP**, чтобы запустить службу синхронизации NTP.
- Чтобы добавить NTP-сервер, нажмите **Добавить сервер** и заполните необходимые поля:
 - Адрес NTP-сервера, например `ntp1.vniiftri.ru`.
 - Минимальный и максимальный интервалы для частоты опроса сервера. Частота опроса может меняться динамически: при нестабильном времени сервер опрашивается чаще, при стабильной синхронизации — реже.

В качестве значения указывается степень двойки, интервал опроса NTP-сервера рассчитывается как 2 в степени n секунд.

Например:

- $6 \rightarrow 2^6 = 64$ секунды;
- $10 \rightarrow 2^{10} = 1024$ секунды (~ 17 минут).

4. Нажмите **Добавить**.
5. Чтобы изменить параметры сервера, нажмите на имя сервера в таблице. Откроется окно с параметрами сервера. Внесите необходимые правки и сохраните изменения.
6. Чтобы удалить сервер, нажмите значок удаления в таблице напротив этого сервера.

10.1.4.2 CLI

Можно выполнять следующие операции:

- Включение NTP (`set system ntp enable`);
- Выключение NTP (`unset system ntp enable`);
- Добавление NTP-сервера (`set system ntp server <name>`);
- Проверка статуса NTP (`ntp`);
- Удаление NTP-сервера (`unset system ntp server <name>`).

Подробнее см. [ntp](#) и [configuration](#).

10.1.5 Настройка конфигурации

В этой статье:

- [Hostname](#)
- [Сетевые интерфейсы](#)

10.1.5.1 Hostname

Имя устройства Angie ADC задается при разворачивании системы. В дальнейшем имя устройства отображается в верхней части экрана веб-интерфейса.

Изменение имени устройства

1. В веб-интерфейсе Angie ADC перейдите **Система** → **Конфигурация**.
2. В поле **Hostname (имя устройства)** введите новое имя устройства и нажмите **Сохранить**.

CLI

Можно выполнять следующие операции:

- Получение имени устройства.
- Изменение имени устройства.

Доступные команды CLI см. в описании команды [set \(CLI\)](#).

Мастер настройки

Можно выполнять следующие операции:

- Просмотр имени устройства.
- Изменение имени устройства.

Подробнее см. [Настройка в мастере](#).

10.1.5.2 Сетевые интерфейсы

Вы можете просмотреть и изменить параметры *настроенных сетевых интерфейсов* в веб-интерфейсе.

Просмотр настроенных сетевых интерфейсов

1. В веб-интерфейсе Angie ADC перейдите Система → Конфигурация.

В открывшемся окне в блоке **Сетевые интерфейсы** отображается список настроенных сетевых интерфейсов.

Для каждого интерфейса отображаются следующие параметры:

- Имя сетевого интерфейса.
- MAC-адрес (формат: "XX:XX:XX:XX:XX:XX").
- Тип конфигурации: **static**, DHCP.
- Массив IPv4-адресов в формате CIDR.
- Массив IPv6-адресов в формате CIDR.
- IP-адрес шлюза по умолчанию (глобальный для всех интерфейсов), если установлен.
- Массив IP-адресов DNS-серверов.
- Массив доменов поиска DNS.

Изменение параметров сетевого интерфейса

Предупреждение

Не рекомендуется изменять настройки интерфейсов узлов, объединенных в пару высокой доступности, иначе пара может потерять связность и работоспособность.

1. В веб-интерфейсе Angie ADC перейдите Система → Конфигурация.
2. В блоке **Сетевые интерфейсы** нажмите на сетевой интерфейс, параметры которого вы хотите изменить.

Откроется окно с параметрами выбранного сетевого интерфейса.

Примечание

Если для интерфейса был указан способ получения настроек по DHCP, то в окне редактирования будет отображаться подсказка с DHCP-параметрами интерфейса.

3. Внесите необходимые изменения и нажмите **Сохранить**.

CLI

Можно выполнять следующие операции:

- Просмотр списка сетевых интерфейсов.
- Изменение параметров сетевых интерфейсов.

Доступные команды CLI см. в описании команды *set (CLI)*.

Мастер настройки

Можно выполнять следующие операции:

- Просмотр и настройка сетевых интерфейсов, бондов и VLAN-интерфейсов.

Подробнее см. *Настройка в мастере*.

10.1.6 Управление пользователями

В системе Angie ADC могут работать локальные пользователи системы и внешние пользователи, аутентифицированные через внешний сервер аутентификации.

Пользователям можно назначить одну из следующих ролей:

- **Администратор:** обладает полными правами на все действия в системе;
- **Супервизор:** имеет ограниченные права, может только просматривать информацию в системе, например, проводить аудит;
- **Оператор:** имеет ограниченные права, может редактировать файлы конфигурации балансировщика, к которым ему предоставлен доступ, и просматривать статистику зон, к которым ему предоставлен доступ. Доступы предоставляет администратор. Остальные настройки Angie ADC оператору не доступны и не отображаются в веб-интерфейсе.

Примечание

При двух неудачных попытках входа в течение одного часа любой пользователь будет временно заблокирован.

10.1.6.1 Локальные пользователи

По умолчанию предустановлен локальный пользователь **admin** с правами администратора. Пользователь **admin** нужен для создания пары высокой доступности.

Примечание

Рекомендуется сменить пароль для пользователя **admin** после первого входа.

Локальные пользователи системы отображаются в веб-интерфейсе в разделе Система → Пользователи. Для каждого пользователя представлена следующая информация:

Логин	Логин пользователя в системе
Имя	Собственное имя пользователя
Фамилия	Фамилия пользователя
UUID	Уникальный идентификатор пользователя
Статус	Статус пользователя в системе: ● Активный — пользователь имеет доступ к системе. ● Неактивный — у пользователя временно отключен доступ к системе, например, на время отпуска. ● Удален — пользователь удален, но может отображаться в списке, если включен переключатель Показывать удаленных пользователей.
Роль	Роль пользователя в системе: ● Администратор — обладает полными правами на все действия в системе. ● Супервизор — имеет ограниченные права, может только просматривать информацию в системе, например, проводить аудит; ● Оператор — имеет ограниченные права, может редактировать файлы конфигурации балансировщика, к которым ему предоставлен доступ, и просматривать статистику зон, к которым ему предоставлен доступ. Доступы предоставляет администратор. Остальные настройки Angie ADC оператору не доступны и не отображаются в веб-интерфейсе.
Комментарий	Примечание

10.1.6.2 Внешние пользователи

Angie ADC поддерживает аутентификацию и авторизацию пользователей через внешний сервер. При использовании внешней аутентификации и авторизации данные о пользователях хранятся на внешнем сервере. У системы нет доступа к данным пользователя.

На данный момент поддерживаются провайдеры *RADIUS*, *LDAP* и *TACACS*. Подробнее см. *Аутентификация* и *Авторизация*.

10.1.6.3 Добавление локального пользователя

Чтобы добавить нового локального пользователя:

1. Откройте веб-интерфейс Angie ADC. В панели навигации перейдите в раздел **Система** → **Пользователи**.

Откроется список пользователей.

2. В открывшемся окне нажмите кнопку **+ Добавить пользователя**.
3. Укажите логин, пароль, имя и фамилию нового пользователя, а также его роль и нажмите **Добавить**.

Требования:

- Логин — допускаются латинские буквы, цифры, а также специальные символы: `_ - . @ ! # $ % ^ & * ( ) = +`.
- Пароль — допускаются латинские буквы, цифры, а также специальные символы: `_ - . @ ! # $ % ^ & * ( ) = +`. Пароль должен соответствовать текущей *парольной политике*. Пароль можно ввести вручную или сгенерировать. Пароль не может совпадать с логином без учета регистра.

Примечание

Кнопка копирования пароля будет работать, если используется защищенное подключение к веб-интерфейсу (*HTTPS*).

- Имя — допускаются только буквы латиницы и кириллицы, а также дефис и пробел.
- Фамилия — допускаются только буквы латиницы и кириллицы, а также дефис и пробел.
- Роль:
 - Администратор — обладает полными правами на все действия в системе;
 - Супервизор — имеет ограниченные права, может только просматривать информацию в системе, например, проводить аудит;
 - Оператор — имеет ограниченные права, может редактировать файлы конфигурации балансировщика, к которым ему предоставлен доступ, и просматривать статистику зон, к которым ему предоставлен доступ. Доступы предоставляет администратор. Остальные настройки Angie ADC оператору не доступны и не отображаются в веб-интерфейсе.

Новый пользователь отобразится в таблице.

4. Передайте пользователю логин и пароль безопасным способом.

10.1.6.4 Изменение учетных данных и статуса локального пользователя

Чтобы изменить данные локального пользователя:

1. Откройте веб-интерфейс Angie ADC. В панели навигации перейдите в раздел Система → Пользователи.
- Откроется список пользователей.
2. Нажмите значок редактирования рядом с пользователем, данные или статус которого вы хотите изменить.
3. В открывшемся окне укажите новые данные и нажмите Сохранить.

Информация о пользователе будет обновлена.

10.1.6.5 Сброс пароля локального пользователя

Администратор может вручную сбросить пароль пользователя и задать для него новый пароль. После сброса пароля пользователю необходимо заново войти в систему.

1. Перейдите в раздел Система → Пользователи.
2. Нажмите значок "ключ" (Сбросить пароль) напротив нужного пользователя.
3. В открывшемся окне введите новый пароль для пользователя и повторите его. Можно ввести пароль вручную или сгенерировать его с учетом текущей парольной политики.

Примечание

Кнопка копирования пароля будет работать, если используется защищенное подключение к веб-интерфейсу (*HTTPS*).

4. Нажмите Сбросить пароль.
- Старый пароль будет сброшен, новый пароль будет сохранен.
5. Передайте пользователю новый пароль безопасным способом.

10.1.6.6 Сброс настроек двухфакторной аутентификации локального пользователя

Администратор может вручную сбросить настройки двухфакторной аутентификации для локального пользователя, например, если пользователь больше не имеет доступа к подключенному приложению для аутентификации. После сброса настроек двухфакторной аутентификации пользователю необходимо заново войти в систему. При входе в систему пользователь должен настроить двухфакторную аутентификацию заново, если этого требует *парольная политика*.

1. Перейдите в раздел Система → Пользователи.
2. Нажмите значок "экран с крестиком" (Сбросить TOTP) напротив нужного пользователя.

Настройки двухфакторной аутентификации будут сброшены.

10.1.6.7 Удаление локального пользователя

Чтобы удалить локального пользователя:

1. Откройте веб-интерфейс Angie ADC. В панели навигации перейдите в раздел Система → Пользователи. Откроется список пользователей.
2. Нажмите значок удаления рядом с пользователем, запись о котором вы хотите удалить.

Пользователь будет удален из списка, но будет отображаться в списке пользователей, если включен переключатель Показывать удаленных пользователей. В дальнейшем вы можете просмотреть список удаленных пользователей при аудите системы.

Примечание

Если вы захотите вернуть удаленного пользователя в систему, необходимо изменить его статус на Активный.

10.2 Управление сертификатами

Angie ADC поддерживает загрузку сертификатов данных через веб-интерфейс (Система → Сертификаты). Сертификаты данных используются для обработки TLS-соединений балансируемого трафика (порты 80/443); в систему можно загрузить несколько сертификатов такого типа.

adc-psi-2 > Сертификаты

Сертификаты

 Загрузить сертификат

Имя	Действителен	Загружен	Пути	
сертификат	30.06.2026 — 30.06.2027	01.07.2026	/etc/angie-lb/crt/active/сертификат.crt /etc/angie-lb/crt/active/сертификат.key	
o	30.06.2026 — 30.06.2027	01.07.2026	/etc/angie-lb/crt/active/o.crt /etc/angie-lb/crt/active/o.key	

Примечание

Для работы с сертификатами через веб-интерфейс необходимо предварительно создать кластер Angie ADC. Подробнее см. в статье *Создание кластера*.

10.2.1 Загрузка сертификата данных

1. В веб-интерфейсе Angie ADC перейдите в раздел Система → Сертификаты.
2. Нажмите Загрузить сертификат и заполните следующие поля:

- **Имя сертификата** — произвольное имя сертификата, которое будет отображаться в системе.
- **Сертификат** — содержимое файла сертификата.
- **Приватный ключ** — содержимое файла приватного ключа.

3. Нажмите **Сохранить**.

Сертификат будет загружен. На странице **Сертификаты** будут отображаться данные о сертификате: имя, срок действия, дата загрузки и пути к сертификату и ключу.

Примечание

После загрузки сертификата необходимо вручную настроить его использование в *конфигурации балансировщика нагрузки*.

Например:

```
server {
    listen      80;
    listen      [::]:80;
    listen      443 ssl;
    listen      [::]:443 ssl;
    server_name localhost;

    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers  AES128-SHA:AES256-SHA:RC4-SHA:DES-CBC3-SHA:RC4-MD5;
    ssl_certificate /etc/angie-lb/crt/active/example.crt;
    ssl_certificate_key /etc/angie-lb/crt/active/example.key;
    ssl_session_cache shared:SSL:10m;
    ssl_session_timeout 10m;

    if ($scheme = http) {
        return 301 https://$host:443$request_uri;
    }

    ## location
}
```

Сертификаты данных можно также загружать и просматривать *через CLI*.

10.3 Журналы событий и экспорт

Angie ADC записывает все события системы и действия пользователей в веб-интерфейсе в журналы событий. Вы можете просмотреть журналы событий через CLI и настроить отправку событий на удаленный syslog-сервер для дальнейшего анализа и аудита.

10.3.1 Просмотр журналов событий

Журналы событий Angie ADC можно посмотреть через *CLI*.

Синтаксис: `logs <entity> [{follow|transfer}]`.

Поддерживается *фильтрация*.

Значения для <entity>:

aaa-manager	События модуля, отвечающего за аутентификацию и авторизацию внешних пользователей Angie ADC
adc-gslb	События модуля GSLB (глобальная балансировка)
adc-server	События модуля, обеспечивающего выполнение всего предоставляемого функционала Angie ADC (входная точка)
adc-system	События модуля, отвечающего за выполнение системных команд Angie ADC
adc-tracker	События модуля, отвечающего за RHI-функционал
auth-dataplane	События модуля, отвечающего за <i>аутентификацию клиентов</i> в балансировщике нагрузки
certificate-manager	События сервиса управления сертификатами
ctrl-access	События из access_log модуля, отвечающего за кластерное взаимодействие
ctrl-error	События из error_log модуля, отвечающего за кластерное взаимодействие
gslb-component	События модуля конфигурации GSLB
kern	События ядра
lb-access	События из access_log балансировщика нагрузки
lb-error	События из error_log балансировщика нагрузки
lb-ls	Позволяет получить список журналов балансировщика нагрузки (файлы должны иметь расширение .log) Пример: <pre> \$ logs lb-ls access.log error.log healthmonitoring.log \$ </pre>
mgmt-access	События из access_log модуля, отвечающего за управление Angie ADC
mgmt-error	События из error_log модуля, отвечающего за управление Angie ADC
session-store	События модуля, отвечающего за хранение sticky-сессий в паре высокой доступности
snmp-manager	События модуля SNMP
storage	События из storage_log модуля, отвечающего за управление Angie ADC
supervisor	События модуля, отвечающего за управление кластеризацией Angie ADC
upgrade	События обновления Angie ADC
view <filename>	Позволяет просмотреть произвольный журнал событий, например, добавленный вручную в настройках конфигурации балансировщика. Файл должен находиться в папке /var/log/angie-lb/ и иметь расширение .log. Пример: <pre> \$ logs view error.log </pre>

Дополнительные параметры:

follow	Вывод записей в режиме реального времени.
transfer	Копирование записей в директорию transfer.

10.3.2 Экспорт логов

Angie ADC позволяет экспортировать события в работе системы на удаленные серверы по syslog-протоколу. Для работы функциональности необходимо настроить syslog-серверы, на которые будут отправляться события, и указать уровни критичности событий, определяющие, какие события будут отправляться на удаленный сервер.

Поддерживаются следующие уровни критичности событий:

- **EMERGENCY** — только события, приводящие к аварийной ситуации и полному отказу системы.
- **ALERT** — критические ошибки безопасности или выход из строя важных сервисов.
- **CRITICAL** — серьезные проблемы, которые требуют немедленного исправления.
- **ERROR** — ошибки, влияющие на работу системы, но не приводящие к ее остановке.
- **WARNING** — предупреждения о возможных проблемах.
- **NOTICE** — уведомления о событиях.
- **INFO** — информационные сообщения (сообщения аудита дополнительно имеют помету **audit**).
- **DEBUG** — события отладки.

10.3.2.1 Создание нового syslog-сервера

1. В веб-интерфейсе Angie ADC перейдите **Система → Логирование**.
Откроется окно **Логирование**. В этом окне отображаются настроенные syslog-серверы.
2. Чтобы добавить новый syslog-сервер, нажмите **Добавить сервер**. В открывшемся окне **Новый сервер** заполните следующие поля:
 - Имя сервера (до 255 символов).
 - Доменное имя или IP-адрес (до 255 символов).
 - Порт (по умолчанию используется 514).
 - Протокол: TCP или UDP (по умолчанию: UDP).
 - Уровни критичности: выберите уровни критичности событий, определяющие, какие события будут отправляться на сервер (доступные уровни см. выше).
3. Нажмите **Добавить сервер**.
Новый сервер отобразится в списке настроенных syslog-серверов.
4. Чтобы протестировать соединение с новым сервером, нажмите на имя этого сервера в таблице.
Откроется окно с параметрами выбранного сервера.
5. Нажмите **Проверить соединение**. В случае установки соединения отобразится сообщение **Подключение прошло успешно**.

10.3.2.2 Просмотр настроенных syslog-серверов

1. В веб-интерфейсе Angie ADC перейдите **Система → Логирование**.
Откроется окно **Логирование**. В этом окне отображаются настроенные syslog-серверы.
2. Нажмите на конкретный сервер, чтобы просмотреть его параметры.

10.3.2.3 Изменение параметров syslog-сервера

1. Перейдите в веб-интерфейс **Система → Логирование → <имя_сервера>**.
Откроется окно с параметрами выбранного сервера.
2. Чтобы включить или выключить передачу событий на выбранный syslog-сервер, переведите переключатель **Включить** в соответствующее положение.

3. Чтобы изменить параметры сервера, внесите необходимые правки и нажмите **Сохранить**.
4. Чтобы протестировать соединение с сервером, нажмите **Проверить соединение**.
5. Чтобы удалить сервер, нажмите значок удаления напротив этого сервера.

10.3.2.4 CLI

Можно выполнять следующие операции:

- Создание syslog-сервера.
- Получение списка syslog-серверов.
- Изменение параметров syslog-сервера.
- Удаление syslog-сервера.
- Проверка подключения к syslog-серверу.

Доступные команды CLI см. в описании команды *syslog*.

10.4 Аудит действий пользователей

Действия пользователей, выполняемые в веб-интерфейсе, записываются в журнал событий Angie ADC и имеют помету "msg":"audit".

Примечание

На данный момент логируются только действия, выполняемые через API. В дальнейших релизах будет реализовано логирование действий пользователей, выполняемых через CLI.

Просмотреть журнал событий можно *через CLI*. Для этого необходимо ввести команды `logs → adc-server`.

При *отправке на syslog-сервер* события аудита помечаются с помощью пользовательской категории `facility=local4` (в соответствии с RFC 3164). Вы можете настроить фильтрацию по `facility=local4` на своем syslog-сервере для обработки только аудит-логов.

В сообщениях аудита выводятся следующие поля:

- `time` — время фиксации события.
- `level` — уровень критичности события: всегда `INFO` (информационное событие).
- `msg` — тип события: всегда `audit` (аудит действий пользователя).
- `app` — устройство, на котором произошло событие.
- `node_id` — имя устройства, на котором произошло событие.
- `actor_id` — идентификатор пользователя, инициировавшего событие.
- `actor_name` — имя пользователя, инициировавшего событие.
- `actor_auth_source` — тип аутентификации пользователя, например локальная или RADIUS.
- `resource` — категория объекта, к которому относится операция.
- `operation` — выполняемая операция, например `read`, `update`, `delete`.
- `mutating` — изменяет ли выполняемая операция систему (`true`, `false`).
- `resource_id` — идентификатор конкретного экземпляра объекта, если применимо.
- `actor_host` — IP-адрес клиента.
- `client_channel` — канал взаимодействия, например веб-интерфейс, CLI или межсервисный вызов.

- `duration_ms` — время выполнения операции в миллисекундах.
- `result` — результат выполнения (OK, `InvalidArgument`).
- `stage` — стадия выполнения операции: `pre` — намерение выполнить действие, `post` — результат выполнения.
- `request_id` — идентификатор запроса, связывающий события одной операции.
- `trace_id` — идентификатор цепочки выполнения запроса.
- `span_id` — идентификатор конкретной операции внутри цепочки выполнения запроса.

10.5 Диагностика сети

Вы можете проводить базовую диагностику сети через веб-интерфейс (Диагностика → Сеть).

angie-adc1 > Сеть

Диагностика сети

Ctrl+P | Предыдущая команда | Enter | Выполнить команду

traceroute 1.1.1.1

Отменить

Сохранить | Очистить

```
$ traceroute 1.1.1.1
traceroute to 1.1.1.1 (1.1.1.1), 30 hops max, 60 byte packets
```

847ms Выполняется...

```
$ ping -c 4 192.168.105.2
```

56.4s Отменено

```
PING 192.168.105.2 (192.168.105.2) 56(84) bytes of data:
64 bytes from 192.168.105.2: icmp_seq=1 ttl=255 time=2.87 ms
64 bytes from 192.168.105.2: icmp_seq=2 ttl=255 time=3.63 ms
64 bytes from 192.168.105.2: icmp_seq=3 ttl=255 time=3.39 ms
```

Поддерживаются команды `ping`, `curl` и `traceroute`. На вкладке отображается окно для ввода команды и окно вывода, в котором в режиме реального времени отображается процесс выполнения команды. При необходимости можно выгрузить отчет в HTML-формате (кнопка **Сохранить**).

Команды `ping` и `traceroute` поддерживают все стандартные аргументы (см. документацию `ping`, `traceroute` на man7.org).

Команда `curl` имеет ограничения:

- Разрешены все поддерживаемые протоколы, кроме `file` (см. документацию `curl` на man7.org).
- Разрешены только следующие аргументы:
 - `--v` (`--verbose`)
 - `--s` (`--silent`)
 - `--S` (`--show-error`)
 - `--i` (`--include`)
 - `--I` (`--head`)
 - `--L` (`--location`)
 - `--k` (`--insecure`)
 - `--m` (`--max-time`)
 - `--x` (`--proxy`)
 - `--u` (`--user`)

- --H (--header)
- --A (--user-agent)
- --e (--referer)
- --d (--data)
- --b (--cookie)
- --G (--get)
- --X (--request)
- --w (--write-out)

Примеры:

```
curl https://example.com      # проверка доступности веб-сервиса по HTTP/HTTPS
ping 192.0.2.10              # проверка сетевой доступности узла и времени отклика
traceroute 192.0.2.10        # определение маршрута до удаленного узла
```

Примечание

Также доступна расширенная диагностика через CLI (контекст *diagnostics*, поддерживаемые команды: `arping`, `diag`, `nslookup`, `ping`, `pingc`, `ss`, `ssldump`, `tcpdump`, `traceroute`).

10.6 Резервное копирование и восстановление системы

Angie ADC поддерживает резервное копирование и восстановление системы на новом узле, а также восстановление текущего узла из резервной копии.

Angie ADC > [Резервное копирование и восстановление узла](#)

Резервное копирование и восстановление узла

Есть резервная копия

Состояние на: 14.08.2026, 13:54



adc-psi-3

10.21.20.36 · VA angie-va-base-20260810 / ADC v1.1

Создание резервной копии

Восстановление из копии

 Создать новую копию

 Скачать

 Применить резервную копию

Дата создания	Размер
14.08.2026 13:54	1.1 МБ
Версия VA / Angie ADC	
angie-va-base-20260810 / v1.1	
Узел-источник	
adc-psi-3	

При переносе резервной копии на новый узел он станет полной копией исходного узла:

- сохраняются IP-адреса и сетевая конфигурация исходного узла;

- будут перенесены все данные и настройки;
- если узел был в *паре высокой доступности*, то пара восстановится;
- если узел был в *кластере*, то соседи снова будут видеть этот узел как исходный.

Важно

Если вы проводите восстановление из резервной копии на новом узле, то исходный узел необходимо отключить, иначе возникнет конфликт IP-адресов.

10.6.1 Статусы узла

Статусы узла отображают текущее состояние узла или этапы резервного копирования и восстановления.

В блоке **Создание резервной копии** у узла могут быть следующие статусы:

- Копия еще не создана;
- Создание копии...;
- Есть резервная копия;
- Не удалось создать резервную копию.

В блоке **Восстановление из копии** у узла могут быть следующие статусы:

- Файл восстановления еще не загружен;
- Готов к восстановлению;
- Проверка архива;
- Не удалось проверить архив;
- Запущено восстановление из резервной копии;
- Восстановление завершено;
- Восстановление не удалось;
- Восстановлен из копии.

10.6.2 Создание резервной копии

1. В веб-интерфейсе Angie ADC перейдите в раздел Система → Резервное копирование и восстановление.

2. Выберите **Создание резервной копии** и нажмите **Создать копию**.

Будет запущен процесс создания резервной копии текущего узла.

После завершения процесса отобразится информация о созданной копии:

- дата создания;
- размер;
- версия VA и версия Angie ADC в копии;
- узел-источник копии (hostname).

10.6.3 Обновление резервной копии

i Примечание

Поддерживается хранение только одной резервной копии узла, при загрузке новой копии предыдущая удаляется.

1. В веб-интерфейсе Angie ADC перейдите в раздел Система → Резервное копирование и восстановление.
2. Выберите Создание резервной копии и нажмите Создать новую копию.

Будет запущен процесс создания новой резервной копии текущего узла. Старая копия будет удалена.

После завершения процесса отобразится информация о новой копии:

- дата создания;
- размер;
- версия VA и версия Angie ADC в копии;
- узел-источник копии (hostname).

10.6.4 Скачивание резервной копии

Вы можете скачать резервную копию текущего узла, чтобы затем применить ее на новом узле.

1. В веб-интерфейсе Angie ADC перейдите в раздел Система → Резервное копирование и восстановление.
2. Выберите Создание резервной копии.

Отобразится информация о текущей резервной копии.

3. Если необходимо, обновите копию, нажав Создать новую копию.
4. Нажмите Скачать.

Будет загружен файл .tar, содержащий текущую резервную копию узла.

10.6.5 Восстановление текущего узла из резервной копии

1. В веб-интерфейсе Angie ADC перейдите в раздел Система → Резервное копирование и восстановление.
2. Выберите Создание резервной копии и нажмите Применить резервную копию.
3. В открывшемся окне подтвердите операцию, нажав Начать восстановление.

После завершения восстановления система перезагрузится. Веб-интерфейс будет доступен по прежнему адресу.

i Примечание

Если восстановление завершилось с ошибкой, то по кнопке Скачать лог вы можете загрузить файл `restore.log` и получить диагностическую информацию. Если доступ к веб-интерфейсу был потерян, лог также можно скачать *через CLI*.

10.6.6 Восстановление из резервной копии на новом узле

1. Разверните Angie ADC на новом узле и назначьте временный IP-адрес для веб-интерфейса. Другие настройки не требуются.

Важно

При восстановлении из резервной копии рекомендуется, чтобы версия VA и версия Angie ADC в копии и на новом узле совпадали.

2. В веб-интерфейсе Angie ADC перейдите в раздел Система → Резервное копирование и восстановление.
3. Выберите Восстановление из копии и загрузите файл .tar, содержащий резервную копию узла, который вы хотите восстановить.
Система проверит файл и отобразит сводную таблицу с основными параметрами текущего узла и загруженной копии.
4. Если были выявлены критические несовпадения и необходимо заменить копию на более подходящую, нажмите **Заменить файл** и загрузите новый архив.

Важно

Не рекомендуется начинать восстановление, если версии VA или Angie ADC не совпадают.

5. Нажмите **Начать восстановление**.

После завершения восстановления система перезагрузится. Веб-интерфейс будет доступен по адресу, который использовался в копии. В статусе узла будет отображаться **Восстановлен из копии**.

Примечание

Если восстановление завершилось с ошибкой, то по кнопке **Скачать лог** вы можете загрузить файл `restore.log` и получить диагностическую информацию. Если доступ к веб-интерфейсу был потерян, лог также можно скачать *через CLI*.

Совет

- Чтобы просмотреть данные копии, которая использовалась для восстановления, перейдите в блок **Восстановление из копии**.
- Чтобы заново запустить восстановление из копии, в блоке **Восстановление из копии** нажмите **Загрузить файл** и загрузите архив еще раз.

10.7 Справочник команд (CLI)

Рекомендуемый интерфейс для всех операций кроме настройки VRRP — CLI на порту 2222 или 22.

Примечание

Настройка VRRP поддерживается только через CLI на порту 2022 (подробнее см. *VRRP-маршрутизация*).

Возможности: автодополнение команд по **TAB**; поддержка истории команд; подсказки по синтаксису команд по нажатию **?**; поддержка цветовой схемы; поддержка сокращенных команд.

Запуск CLI на порту 2222

При запуске Angie ADC интерфейс командной строки Angie ADC будет доступен по SSH-протоколу на порту 2222. Доступ осуществляется с помощью SSH-клиента.

Чтобы запустить интерфейс командной строки (CLI), выполните следующие действия:

1. Подключитесь к SSH-серверу:

```
ssh -p 2222 user@localhost
```

2. После подключения сервер запросит пароль для авторизации. Введите пароль:

```
user@localhost's password:
#
```

Если авторизация прошла успешно, откроется терминал Angie ADC. Справка доступна по ?.

3. Для настройки протоколов BGP, OSPF и BFD дополнительно перейдите в настройки vtysh:

```
vttysh
```

Команда переключит вас в конфигурационный режим, где можно вносить изменения.

Запуск CLI на порту 2022

При запуске Angie ADC интерфейс командной строки Angie ADC будет доступен по SSH-протоколу на порту 2022. Доступ осуществляется с помощью SSH-клиента.

Чтобы запустить интерфейс командной строки (CLI), выполните следующие действия:

1. Подключитесь к SSH-серверу:

```
ssh -p 2022 user@localhost
```

2. После подключения сервер запросит пароль для авторизации. Введите пароль:

```
user@localhost's password:
#
```

Если авторизация прошла успешно, откроется терминал Angie ADC. Справка доступна по ?.

3. Для настройки протоколов BGP, OSPF и BFD дополнительно перейдите в настройки vtysh:

```
vttysh
```

Команда переключит вас в конфигурационный режим, где можно вносить изменения.

10.7.1 Поддерживаемые команды

Ниже приведен список команд в алфавитном порядке для новой версии CLI (на порту 2222).

Основные команды:

1. <i>certificates</i>	Управление сертификатами traffic в Angie ADC
<i>list</i>	Вывод списка сертификатов
<i>info</i>	Просмотр информации о сертификате
<i>upload</i>	Загрузка сертификата и ключа
<i>remove</i>	Удаление сертификата и ключа

продолжается на следующей странице

Таблица 1 – продолжение с предыдущей страницы

2. <i>configuration</i>	Переход в режим настройки Angie ADC
<i>commit</i>	Применение конфигурации-кандидата вместо текущей конфигурации.
<i>end</i>	Выход из режима настройки <i>configuration</i> .
<i>rollback</i>	Отмена всех изменений в конфигурации-кандидате.
<i>set</i>	Изменение текущей конфигурации: добавление строки (создает конфигурацию-кандидат): • системные настройки: <i>hostname</i> / <i>interface</i> / <i>ntp</i> / <i>sysctl</i> / <i>timezone</i>; • настройки маршрутизации: <i>ip route</i> / <i>rule</i>.
<i>show</i>	Просмотр текущей конфигурации и конфигурации-кандидата.
<i>unset</i>	Изменение текущей конфигурации: удаление строки (создает конфигурацию-кандидат).
3. <i>diagnostics</i>	Сетевая диагностика
<i>arping</i>	Отправка ARP-запросов сетевым хостам.
<i>diag</i>	Общая диагностика системы.
<i>nslookup</i>	Выполнение DNS-запросов.
<i>ping</i>	Выполнение команды <i>ping</i> .
<i>pingc</i>	Выполнение команды <i>ping</i> в стиле Cisco.
<i>ss</i>	Просмотр информации о сетевых соединениях, портах и статистике сетевых сокетов.
<i>ssldump</i>	Анализ SSL/TLS-трафика.
<i>tcpdump</i>	Анализ сетевого трафика.
<i>traceroute</i>	Трассировка маршрута следования пакетов.
4. <i>firewall</i>	Управление правилами брандмауэра
<i>list</i>	Вывод списка правил брандмауэра Angie ADC
<i>no open</i>	Закрытие порта в брандмауэре Angie ADC
<i>open</i>	Открытие порта в брандмауэре Angie ADC
<i>save</i>	Сохранение правил брандмауэра Angie ADC
5. <i>ip</i>	Стандартное управление сетевыми интерфейсами, маршрутами, адресами и правилами IP-трафика в Linux
6. <i>logs</i>	Просмотр журналов
7. <i>modules</i>	Просмотр информации о динамических модулях
<i>info</i>	Получение информации о динамическом модуле Angie ADC
<i>list</i>	Получение списка доступных динамических модулей Angie ADC
8. <i>ps</i>	Отображение информации о процессах, запущенных в системе
9. <i>settings</i>	Просмотр и изменение системных настроек Angie ADC (NTP, отладка, настройка времени и т. д.)
<i>angie debug</i>	Включение режима отладки для локального балансировщика нагрузки
<i>no angie debug</i>	Отключение режима отладки
<i>reload</i>	Перезагрузка элементов Angie ADC
<i>sysctl</i>	Просмотр параметров ядра Linux в реальном времени
<i>syslog</i>	Управление syslog/Logstash-серверами логирования.

продолжается на следующей странице

Таблица 1 – продолжение с предыдущей страницы

10. <i>system</i>	Перезагрузка и обновление Angie ADC
<i>ntp</i>	Просмотр статуса NTP
<i>reboot</i>	Перезагрузка системы Angie ADC
<i>stat</i>	Просмотр статистики системы Angie ADC
<i>time</i>	Просмотр и установка системного времени
<i>timezone</i>	Просмотр часового пояса системы
<i>upgrade</i>	Обновление Angie ADC
<i>version</i>	Просмотр версии Angie ADC
11. <i>transfer</i>	Работа с директорией transfer
<i>edit</i>	Редактирование файла в текущей директории.
<i>ls</i>	Получение списка файлов в директории transfer
<i>rm</i>	Удаление файла в директории transfer
<i>view</i>	Просмотр содержимого файла в директории transfer
12. <i>vttysh</i>	Переход к настройкам маршрутизации по протоколам BGP и OSPF
13. <i>web</i>	Настройка веб-интерфейса Angie ADC
<i>upload-cert</i>	Загрузка сертификата управления и ключа
<i>tls</i>	Включение / выключение HTTPS

Служебные команды:

exit или Ctrl+Z	Выход из контекста
quit или Ctrl+Q	Завершение сессии
show	Вывод текущего состояния
?	Список доступных команд

10.7.2 Синтаксис

```
command option0 {option1|option2|...|option N} [option3] [{option4|option5|...|option M}] [option6 <user-input1>] [<user-input2>] <user-input3>
```

Параметры:

command	команда
option0	обязательный параметр
{option1 option2 ... option N}	выбор из списка обязательных параметров
[option3]	необязательный параметр
[{option4 option5 ... option M}]	выбор из списка необязательных параметров
[option6 <user-input1>]	необязательный параметр с пользовательским аргументом
[<user-input2>]	необязательный пользовательский аргумент
<user-input3>	обязательный пользовательский аргумент

10.7.3 certificates

Контекст **certificates** позволяет управлять сертификатами, используемыми Angie ADC.

Поддерживаемые команды:

<code>certificates</code>	Вход в контекст
<code>list</code>	Вывод списка сертификатов <code>traffic</code> в Angie ADC
<code>info</code>	Просмотр информации о сертификате
<code>upload</code>	Загрузка сертификата и ключа
<code>remove</code>	Удаление сертификата и ключа

10.7.3.1 list

Вывод списка сертификатов `traffic` Angie ADC.

Синтаксис: `list`.

Пример:

```
(certificates)$$ list
Name      Validity                Uploaded    Paths
-----
cert1     18.02.2025 - 18.02.2027 06.08.2026 /etc/angie-lb/crt/active/cert1.crt
                                     /etc/angie-lb/crt/active/cert1.key
```

10.7.3.2 info

Просмотр информации о сертификате.

Синтаксис: `info <name>`.

Параметры:

- `<name>` — имя сертификата.

Пример:

```
(certificates)$$ info cert1
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 1 (0x1)
    Signature Algorithm: sha1WithRSAEncryption
    Issuer: C = RU, ST = Moscow, O = Angie, OU = EVO, CN = Root CA
    Validity
      Not Before: Feb 18 12:23:08 2025 GMT
      Not After : Feb 18 12:23:08 2027 GMT
    Subject: CN = Demo Certificate, ST = Moscow, C = RU, O = Angie, OU = EVO
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      Public-Key: (4096 bit)
      Modulus:
...
(certificates)$$
```

10.7.3.3 upload

Загрузка сертификатов и ключей Angie ADC.

Примечание

Для работы с сертификатами необходимо предварительно создать кластер Angie ADC. Подробнее см. в статье [Создание кластера](#).

Синтаксис: `upload <name>`.

Параметры:

- `<name>` — имя, назначаемое сертификату.

Для загрузки сертификата или ключа необходимо вставить текст сертификата или ключа в строку, затем вставить пустую строку или написать `END` в новой строке.

Пример:

```
(certificates)$ upload cert1
Certificate (PEM):
Insert PEM data line by line at the end of the input, add an empty line or enter END
-----BEGIN CERTIFICATE-----
MIIF9TCCA92gAwIBAgIBATANBgkqhkiG9w0BAQUFADBOMQswCQYDVQQGEwJSVTEP
...
Z7SIEeCr84fIlxFMpRxNGZhJQ67xWLwQmjVcphi037Wpx+6dmQRtp8Q=
-----END CERTIFICATE-----

Private key (PEM):
Insert PEM data line by line at the end of the input, add an empty line or enter END
-----BEGIN PRIVATE KEY-----
MIIJQgIBADANBgkqhkiG9w0BAQEFAASCCSwwggkoAgEAAoICAQDA9J1JqGANy5uf
...
AgjJ9gmjldD6t1dNCwLVxc8thGVWSw==
-----END PRIVATE KEY-----

Certificate 'cert1' uploaded successfully
(certificates)$
```

При загрузке ключа и сертификата проверяется их корректность. В случае некорректного значения сертификат или ключ не будет сохранен.

10.7.3.4 remove

Удаление сертификатов и ключей Angie ADC.

Синтаксис: `remove <name>`.

Параметры:

- `<name>` — имя сертификата.

Пример:

```
(certificates)$ remove cert1
Certificate 'cert1' deleted successfully
```

10.7.4 configuration

Режим настройки Angie ADC. В этом режиме можно изменять текущую конфигурацию Angie ADC.

Чтобы внести изменения:

1. Задайте необходимые параметры с помощью команд `set` (добавляет строку) и `unset` (удаляет строку).
2. Сравните текущую конфигурацию и конфигурацию-кандидат с помощью команды `show`.
3. Примените изменения с помощью команды `commit`.

Примечание

Изменения будут применены только после выполнения команды `commit`.

Если необходимо, вы можете откатить всю конфигурацию-кандидат с помощью команды `rollback` до ее сохранения.

Пример:

```
$$ configuration
## set system interface enp0s2 type dhcp
## show
-> Current configuration
set system interface enp0s2 type static
set system interface enp0s2 ip address 10.21.20.31/22
set system interface enp0s2 ip gateway 10.21.20.1
set system interface enp0s2 ip dns 8.8.8.8
set system interface enp0s2 ip dns 1.1.1.1

-> Candidate configuration
set system interface enp0s2 type dhcp
## commit
```

Поддерживаемые команды:

<code>configuration</code>	Вход в режим настройки
<code>commit</code>	Применение конфигурации-кандидата вместо текущей конфигурации.
<code>end</code>	Выход из режима настройки <code>configuration</code> .
<code>rollback</code>	Отменяет все изменения в конфигурации-кандидате.
<code>set</code>	Изменение текущей конфигурации: добавление строки (создает конфигурацию-кандидат).
<code>show</code>	Просмотр текущей конфигурации и конфигурации-кандидата.
<code>unset</code>	Изменение текущей конфигурации: удаление строки (создает конфигурацию-кандидат).

10.7.4.1 `commit`

Применение конфигурации-кандидата в качестве активной конфигурации.

10.7.4.2 `end`

Выход из режима `configuration`.

10.7.4.3 `rollback`

Отменяет все изменения в конфигурации-кандидате.

10.7.4.4 `set`

Переход к изменению текущей конфигурации Angie ADC.

Примечание

Будет создана конфигурация-кандидат. Изменения будут применены только после выполнения команды `commit`.

Синтаксис: `set system <system_input> | ip <ip_input>`, где:

- `system <system_input>` — изменение параметров конфигурации (hostname / interface / ntp / sysctl / timezone);
- `ip <ip_input>` — настройка маршрутов и правил IP-трафика (route / rule).

Возможные варианты `<system_input>`:

- `set system hostname <name>` — установка имени хоста для Angie ADC.
- `set system interface <name> {ip <keyword> <value> | type {static | dhcp}}` — настройка интерфейса.
 - Возможные значения для `<keyword>`:
 - * `address` — установка IP-адреса для интерфейса.
 - * `gateway` — установка шлюза для интерфейса.
 - * `dns` — установка DNS-адреса для интерфейса.
 - * `searches` — установка доменов поиска для интерфейса (до 10 доменов).
 - `type static` — используется по умолчанию. Этот параметр можно не указывать явно.
 - `type dhcp` — удаляет все IP-параметры интерфейса и включает режим автоматического получения настроек с DHCP-сервера.
- `set system ntp enable | ntp server <name> [minpoll <value>] [maxpoll <value>]`

Параметры:

- `enable` — включение службы NTP-синхронизации.
- `server <name>` — добавление нового NTP-сервера.
- `minpoll <value>` — минимальный интервал опроса как степень двойки (от 4 до 17, по умолчанию: 6), необязательный параметр.
- `maxpoll <value>` — максимальный интервал опроса как степень двойки (от 4 до 17, по умолчанию: 10), необязательный параметр.
- `set system sysctl <variable> <value>` — настройка параметров ядра Linux (`variable` — настраиваемый параметр, `value` — его значение).
- `set system timezone <value>`

Часовой пояс определяет смещение локального времени относительно UTC. Изменение часового пояса не изменяет системное время (Unix timestamp), а только влияет на его отображение. Часовой пояс должен быть указан в формате IANA Time Zone Database (например, Europe/Moscow) или в виде аббревиатуры (например, UTC, EST, PST).

Примеры:

```
$$ configuration
## set system hostname adc-home
## set system interface enp0s2 ip address 10.21.20.39/22
## set system interface enp0s2 ip gateway 10.21.20.1
## set system interface enp0s2 ip dns 8.8.8.8
## set system interface enp0s2 ip searches angie.software.ru
## set system sysctl net.ipv4.ip_forward 1
## set system sysctl net.core.somaxconn 1024
## set system ntp server pool.ntp.org minpoll 4 maxpoll 17
## set system timezone Europe/Moscow
```

Возможные варианты `<ip_input>`:

- `rule from <address> table <value>` — установка правила.

- `route <address> {nexthop <address> | interface <name>} [table <value>] [type local]` — установка маршрута. Минимально необходимо задать адрес и `nexthop` или `interface`, остальные параметры (таблица маршрутизации и тип маршрута) опциональны.

Пример:

```
$$ configuration
## set ip rule from 0.0.0.0/0 table 100
## set ip route 0.0.0.0/0 nexthop 10.90.90.90
```

10.7.4.5 show

Просмотр текущей конфигурации и конфигурации-кандидата.

10.7.4.6 unset

Переход к изменению текущей конфигурации Angie ADC. Удаляет параметр из конфигурации-кандидата.

Примечание

Будет создана конфигурация-кандидат. Изменения будут применены только после выполнения команды `commit`.

Пример:

```
$$ configuration
## unset ip rule from 0.0.0.0/0 table 100
## show
-> Current configuration
set system interface enp0s2 type static
set system interface enp0s2 ip address 10.21.20.31/22
set ip rule from 0.0.0.0/0 table 100

-> Candidate configuration
set system interface enp0s2 type static
set system interface enp0s2 ip address 10.21.20.31/22
## commit
```

10.7.5 diagnostics

Контекст `diagnostics` позволяет проводить диагностику сети и анализировать сетевой трафик.

Поддерживаемые команды:

<code>diagnostics</code>	Вход в контекст
<code>arping</code>	Отправка ARP-запросов сетевым хостам.
<code>diag</code>	Общая диагностика системы.
<code>nslookup</code>	Выполнение DNS-запросов.
<code>ping</code>	Выполнение команды <code>ping</code> .
<code>pingc</code>	Выполнение команды <code>ping</code> в стиле Cisco.
<code>ss</code>	Просмотр информации о сетевых соединениях, портах и статистике сетевых сокетов.
<code>ssldump</code>	Анализ SSL/TLS-трафика.
<code>tcpdump</code>	Анализ сетевого трафика.
<code>traceroute</code>	Трассировка маршрута следования пакетов.

10.7.5.1 arping

Отправка ARP-запросов сетевым хостам.

Синтаксис: `arping [-c <count>] [-w <deadline>] [-t <interval>] [-if <interface>] [-src <source>] [-d] [-u] [-a] [-f] [-b] [-q] <target>`

Параметры:

- `-c <count>` — количество ARP-запросов для отправки перед завершением команды;
- `-w <deadline>` — максимальное время ожидания ответа в секундах;
- `-t <interval>` — интервал между отправкой запросов в секундах;
- `-if <interface>` — сетевой интерфейс, через который отправлять запросы;
- `-src <source>` — исходный IP-адрес для использования в запросах;
- `-d` — режим обнаружения дубликатов адресов;
- `-u` — режим незапрошенного ARP: отправка ARP-пакетов без ожидания ответа (для обновления ARP-кэша);
- `-a` — режим ответа ARP: команда будет отвечать на входящие ARP-запросы;
- `-f` — завершить выполнение при получении первого ответа;
- `-b` — широковещательный режим: отправка запросов на широковещательный адрес;
- `-q` — тихий режим: минимальный вывод информации;
- `<target>` — IP-адрес или имя хоста, к которому отправляются ARP-запросы (обязательный параметр).

Примеры:

- `(diagnostics)$ arping 192.168.1.1` — проверка доступности хоста.
- `(diagnostics)$ arping -c 5 -w 10 192.168.1.100` — отправка до 5 ARP-запросов и ожидание ответа не более 10 секунд.
- `(diagnostics)$ arping -d 192.168.1.50` — проверка на дублирование адреса.
- `(diagnostics)$ arping -u -if eth0 192.168.1.1` — незапрошенный ARP для обновления кэша.

10.7.5.2 diag

Выполнение общей диагностики. Собираются системные метрики, конфигурация системы и системный журнал за последние 24 часа. Информация сохраняется в файл `diag_<timestamp>.log`, например в папке `/var/transfer`.

Синтаксис: `diag`

10.7.5.3 nslookup

Выполнение DNS-запросов.

Разовый запрос

Синтаксис: `nslookup <host>`

Примеры:

- `(diagnostics)$ nslookup example.com` — прямой DNS-запрос.
- `(diagnostics)$ nslookup 203.0.113.10` — обратный DNS-запрос (PTR).

Интерактивный режим

В интерактивном режиме можно выполнять множественные DNS-запросы.

Переход в интерактивный режим: `(diagnostics)$$ nslookup`

Поддерживаемые команды интерактивного режима:

- `host [<server>]` — поиск информации о хосте с использованием текущего сервера по умолчанию или указанного сервера.
- `server <domain>` — изменение сервера по умолчанию. Для поиска информации о домене используется текущий сервер.
- `lserver <domain>` — изменение сервера по умолчанию. Для поиска информации о домене используется начальный сервер (тот, с которым запущен nslookup).
- `set keyword[=value]` — изменение параметров конфигурации, где **keyword** — имя параметра, **value** — значение (если поддерживается).

Возможные варианты:

- `set all` — вывод всех текущих настроек.
- `set class=value` — изменение класса запроса.

Доступные значения:

- * `IN` — класс Internet (по умолчанию);
- * `CH` — класс Chaos;
- * `HS` — класс Hesiod;
- * `ANY` — подстановочный знак.
- `set [no]debug` — включение или выключение отображения полного пакета ответа. По умолчанию `nodebug`.
- `set [no]d2` — включение или выключение режима отладки. По умолчанию `nod2`.
- `set domain=<name>` — установка списка поиска доменов по умолчанию.
- `set [no]search` — если запрос содержит хотя бы одну точку, но не заканчивается точкой, добавляет имена доменов из списка поиска доменов к запросу до получения ответа. По умолчанию `search`.
- `set port=<value>` — изменение порта TCP/UDP-сервера имен (по умолчанию 53).
- `set querytype=<value>` — изменение типа запроса информации. По умолчанию `A`, затем `AAAA`.
- `set type=<value>` — изменение типа запроса информации. По умолчанию `A`, затем `AAAA`.
- `set [no]recurse` — включение или выключение рекурсивных запросов. По умолчанию `recurse`.
- `set ndots=<number>` — установка количества точек (разделителей меток) в домене, при котором поиск отключается. Абсолютные имена всегда останавливают поиск.
- `set retry=<number>` — установка количества повторных попыток.
- `set timeout=<number>` — изменение начального интервала ожидания ответа (в секундах).
- `set [no]vc` — указывает, должен ли использоваться виртуальный канал при отправке запросов к серверу. По умолчанию `novc`.
- `set [no]fail` — использовать ли следующий сервер имен, при ошибке на первом сервере. По умолчанию `nofail`.

10.7.5.4 ping

Выполнение команды `ping`.

Синтаксис: `ping [-c <count>] [-n] <hostname>`

Параметры:

- `-c <count>` — количество пакетов (по умолчанию бесконечно);
- `-n` — не резолвить имя хоста;
- `<hostname>` — имя хоста или IP-адрес.

10.7.5.5 pingc

Выполнение команды `ping` в стиле Cisco.

Синтаксис: `pingc {ip|ip6} <hostname> [count <n>]`

Параметры:

- `ip` — выполнение пинга через `ipv4`;
- `ip6` — выполнение пинга через `ipv6`;
- `<hostname>` — имя хоста или IP-адрес;
- `count <n>` — количество пакетов (по умолчанию 4).

10.7.5.6 ss

Позволяет получать детальную информацию о TCP/UDP-соединениях, открытых портах, процессах, использующих сеть, и статистике сетевых интерфейсов.

Синтаксис: `ss [-t] [-u] [-l] [-p] [-a] [-n] [-r] [-s] [-e] [-i] [-4] [-6] [-o] [-m] [-h] [<expression>]`

Параметры

- `-t` — отображение только TCP-соединений.
- `-u` — отображение только UDP-соединений.
- `-l` — отображение только прослушивающих сокетов.
- `-p` — отображение процесса (PID и имя программы), использующего сокет.
- `-a` — отображение всех сокетов: как установленных соединений, так и прослушивающих.
- `-n` — выводить только IP-адреса и номера портов.
- `-r` — разрешать IP-адреса и порты в имена (обратное `-n`).
- `-s` — отображение краткой статистики по протоколам (количество соединений, пакетов и т.д.).
- `-e` — расширенный вывод статистики с дополнительной информацией о соединениях.
- `-i` — отображение информации о сетевых интерфейсах.
- `-4` — отображение только IPv4-соединений.
- `-6` — отображение только IPv6-соединений.
- `-o` — отображение таймеров сокетов в выводе.
- `-m` — отображение использования памяти сокетом.
- `-h` — не отображать заголовки таблицы.
- `<expression>` — фильтрация результатов по `state` и по заданным критериям, оформляется кавычками `" "`.

Примеры

Примеры:

- `(diagnostics)$$ ss -tulpn` — все TCP/UDP-соединения с процессами и портами.
- `(diagnostics)$$ ss -tlnp` — TCP-соединения в состоянии LISTEN.
- `(diagnostics)$$ ss "state listening"` — только прослушивающие соединения.
- `(diagnostics)$$ ss "dport = 80"` — соединения на порт 80.
- `(diagnostics)$$ ss "sport >= 1024"` — соединения с портом источника ≥ 1024 .
- `(diagnostics)$$ ss "dst 192.168.1.1"` — соединения к хосту 192.168.1.1.
- `(diagnostics)$$ ss "src 10.0.0.0/8"` — соединения из сети 10.0.0.0/8.
- `(diagnostics)$$ ss "state established dport = 443"` — соединения, установленные на порт 443.
- `(diagnostics)$$ ss "fwmark = 0x01/0x03"` — соединения с определенной меткой fwmark.
- `(diagnostics)$$ ss -s` — статистика по протоколам.

Фильтрация по state

- `state all` — все состояния.
- `state connected` — все соединенные состояния.
- `state synchronized` — синхронизированные состояния.
- `state bucket` — состояния ожидания.
- `state big` — все возможные состояния.
- `state established` — установленное соединение.
- `state syn-sent` — отправлен SYN.
- `state syn-recv` — получен SYN.
- `state fin-wait-1` — ожидание первого FIN.
- `state fin-wait-2` — ожидание второго FIN.
- `state time-wait` — ожидание закрытия.
- `state closed` — закрытое соединение.
- `state close-wait` — ожидание закрытия.
- `state last-ack` — последний ACK.
- `state listening` — прослушивание.
- `state closing` — закрытие соединения.

Фильтрация по критериям

Фильтрация сокетов на основе определенных критериев. Выражение состоит из серии предикатов, объединенных логическими операторами.

Логические операторы (в порядке возрастания приоритета):

<code>or, , </code>	логическое ИЛИ
<code>and, &, &&</code>	логическое И
<code>not, !</code>	логическое НЕ

Если между последовательными предикатами нет оператора, предполагается неявный оператор **and**. Подвыражения можно группировать с помощью скобок "(" и ")".

Предикаты:

dst HOST src HOST dst=HOST src=HOST	Проверка соответствия адреса назначения или источника хосту HOST. HOST может быть IP-адресом, именем хоста или сетью в формате CIDR.
dport OP PORT sport OP PORT	Сравнение порта назначения или источника с PORT. OP (оператор) может быть следующим: <, <=, =, ==, !=, >=, >. Например: dport = 80, sport >= 1024, dport < 8080 .
dev DEVICE dev=DEVICE dev!=DEVICE	Совпадение по устройству (интерфейсу), которое использует соединение. DEVICE может быть именем устройства или индексом интерфейса.
fwmark MASK fwmark=MASK fwmark!=MASK	Совпадение по значению fwmark для соединения. MASK может быть конкретным значением метки или значением с маской битов. Например: fwmark=0x01, fwmark=0x01/0x03 (проверка двух младших битов).
cgroup PATH cgroup=PATH cgroup!=PATH	Совпадение, если соединение является частью cgroup по указанному пути.
cautobound	Совпадение, если порт или путь исходного адреса был автоматически выделен (а не явно указан).

Операторы сравнения (все эквивалентны):

=, ==, eq	равно
!=, ne, neq	не равно
>, gt	больше
<, lt	меньше
>=, ge, geq	больше или равно
<=, le, leq	меньше или равно
!, not	отрицание
 , , or	логическое ИЛИ
&, &&, and	логическое И

Если оператор не указан, предполагается оператор **=**.

10.7.5.7 ssldump

Анализ SSL/TLS-трафика. Показывает содержимое SSL/TLS-пакетов, включая информацию о рукопожатии (handshake), сертификатах, используемых шифрах и зашифрованные данные. Результаты сохраняются в файлы **ssl_capture_<timestamp>.txt** и **ssl_capture_<timestamp>.pcap**.

Синтаксис: **ssldump [-i <interface>] [-n] [-e] [-h] [<expression>]**

Параметры:

- **-i <interface>** — сетевой интерфейс для захвата SSL/TLS-трафика.
- **-n** — показывать только IP-адреса (не выполнять разрешение доменных имен).
- **-e** — показывать время в читаемом формате вместо Unix timestamp.
- **-h** — показывать полные SSL-заголовки пакетов для детального анализа.
- **<expression>** — фильтр захвата, оформляется кавычками "". Синтаксис выражений фильтров см. документацию pcap-filter.

Примеры:

- **(diagnostics)\$\$ ssldump -i eth0** — захват всего SSL-трафика на eth0.

- (diagnostics)\$\$ ssldump -i eth0 "host example.com" — SSL-трафик для/от example.com.
- (diagnostics)\$\$ ssldump -i eth0 "port 443" — SSL-трафик на порт 443.
- (diagnostics)\$\$ ssldump -h -i eth0 "tcp and port 443" — полные заголовки SSL-пакетов.

10.7.5.8 tcpdump

Анализ сетевого трафика в реальном времени.

Синтаксис: `tcpdump [-i <interface>] [-c <count>] [-w <file>] [-v] [-vv] [-vvv] [-n] [-nn] [-x] [-e] [<expression>]`

Параметры:

- `-i <interface>` — указание сетевого интерфейса для захвата трафика (по умолчанию используется первый доступный интерфейс). Если интерфейс не указан, то команда будет использовать все доступные интерфейсы.
- `-c <count>` — количество пакетов для захвата перед автоматическим завершением команды.
- `-w <file>` — запись захваченных пакетов в файл в формате pcap.
- `-v` — подробный вывод: отображение дополнительной информации о пакетах.
- `-vv` — очень подробный вывод: расширенная информация о пакетах.
- `-vvv` — максимально подробный вывод: максимальный уровень детализации.
- `-n` — не выполнять разрешение доменных имен (DNS lookup): отображать только IP-адреса.
- `-nn` — не выполнять разрешение доменных имен и портов: отображать только IP-адреса и номера портов.
- `-x` — вывод пакетов в шестнадцатеричном формате для детального анализа содержимого.
- `-e` — отображение информации о канальном уровне (Ethernet-заголовки): MAC-адреса, тип фрейма и т.д.
- `<expression>` — фильтр захвата для ограничения типа перехватываемых пакетов, оформляется кавычками "". Синтаксис выражений фильтров см. документацию pcap-filter.

Примеры:

- (diagnostics)\$\$ tcpdump -i eth0 — захват всего трафика на eth0.
- (diagnostics)\$\$ tcpdump -i eth0 "host 192.168.1.1" — трафик только для/от хоста 192.168.1.1.
- (diagnostics)\$\$ tcpdump -i eth0 port 80 — трафик на порт 80 (HTTP).
- (diagnostics)\$\$ tcpdump -i eth0 "tcp and port 443" — TCP-трафик на порт 443 (HTTPS).
- (diagnostics)\$\$ tcpdump -w capture.pcap -c 100 — записать 100 пакетов в файл.

10.7.5.9 traceroute

Трассировка маршрута следования пакетов до указанного сетевого узла.

Синтаксис: `traceroute [-i] [-t] [-u] [-4] [-6] [-n] [-d] [-m <max_hops>] [-q <nqueries>] [-p <port>] [-w <waittime>] [-if <interface>] [-s <source>] <target>`

Параметры:

- `-i` — ICMP-режим для трассировки (режим по умолчанию).
- `-t` — TCP-режим: отправка TCP-пакетов вместо ICMP.
- `-u` — UDP-режим: отправка UDP-пакетов.

- **-4** — принудительное использование IPv4.
- **-6** — принудительное использование IPv6.
- **-n** — не выполнять разрешение доменных имен: показывать только IP-адреса.
- **-d** — включение отладочной информации в вывод.
- **-m <max_hops>** — максимальное количество промежуточных узлов (hops) для проверки (по умолчанию 30).
- **-q <nqueries>** — количество запросов, отправляемых на каждый промежуточный узел (по умолчанию 3).
- **-p <port>** — указание порта для использования в TCP/UDP-режимах.
- **-w <waittime>** — время ожидания ответа от каждого узла в секундах (по умолчанию 3).
- **-if <interface>** — указание сетевого интерфейса для отправки пакетов.
- **-s <source>** — указание исходного IP-адреса для отправки пакетов.
- **<target>** — целевой IP-адрес или доменное имя узла, до которого выполняется трассировка (обязательный параметр).

Примеры:

- (diagnostics)\$**\$** **tracert** -n -w 2 192.168.1.1 — трассировка без DNS, время ожидания 2 секунды.
- (diagnostics)\$**\$** **tracert** -t -p 80 example.com — TCP-трассировка на порт 80.

10.7.6 firewall

Контекст **firewall** позволяет управлять правилами брандмауэра Angie ADC.

Поддерживаемые команды:

firewall	Вход в контекст
<i>list</i>	Вывод списка правил брандмауэра Angie ADC
<i>no open</i>	Закрытие порта в брандмауэре Angie ADC
<i>open</i>	Открытие порта в брандмауэре Angie ADC
<i>save</i>	Сохранение правил брандмауэра Angie ADC

10.7.6.1 list

Вывод списка правил брандмауэра Angie ADC.

Синтаксис: **list** {ip|ip6}.

Параметры:

- **ip** — вывод списка правил для IPv4;
- **ip6** — вывод списка правил для IPv6.

Пример вывода:

```
(firewall)$$ list ip
tcp 9999 enp0s2
(firewall)$$
```

10.7.6.2 no open

Закрытие порта в брандмауэре Angie ADC.

Синтаксис: `no open {ip|ip6} {tcp|udp} <port> <interface>`.

Параметры:

- `ip` — порт для IPv4;
- `ip6` — порт для IPv6;
- `tcp` — порт для протокола TCP;
- `udp` — порт для протокола UDP;
- `<port>` — номер порта;
- `<interface>` — интерфейс.

Пример:

```
(firewall)$$ no open ip tcp 9999 enp0s2
Port 9999/tcp successfully closed on interface enp0s2
(firewall)$$ save
rules saved successfully
```

Примечание

Изменения будут применены сразу, но сохранены только после выполнения команды `save`.

10.7.6.3 open

Открытие порта в брандмауэре Angie ADC.

Синтаксис: `open {ip|ip6} {tcp|udp} <port> <interface>`.

Параметры:

- `ip` — порт для IPv4;
- `ip6` — порт для IPv6;
- `tcp` — порт для протокола TCP;
- `udp` — порт для протокола UDP;
- `<port>` — номер порта;
- `<interface>` — интерфейс.

Пример:

```
(firewall)$$ open ip tcp 9999 enp0s2
Port 9999/tcp successfully opened on interface enp0s2
(firewall)$$ save
rules saved successfully
```

Примечание

Изменения будут применены сразу, но сохранены только после выполнения команды `save`.

Примечание

При настройке конфигурации не занимайте следующие порты (используются внутренними сервисами Angie ADC):

TCP: 22, 199, 2022, 2222, 2601, 2602, 2603, 2604, 2605, 2606, 2615, 2616, 2617, 2619, 2623, 3050, 3051, 3053, 3054, 3060, 3111, 3122, 3123, 3301, 3302, 3380, 3391, 5044, 5355, 5432, 8010, 8080, 8443, 9090, 9100, 9120, 9600, 9633, 9898, 9900, 60080

- Порты 8080 и 8443 можно использовать, если у вас выделен отдельный *интерфейс управления*.

UDP: 53, 161, 323, 546, 3784, 3785, 4784, 5355, 7784

10.7.6.4 save

Сохранение правил брандмауэра Angie ADC.

Синтаксис: **save**.

10.7.7 ip

Стандартное управление сетевыми интерфейсами, маршрутами, адресами и правилами IP-трафика в Linux.

Предупреждение

При перезагрузке Angie ADC настройки, выполненные через **ip**, не сохраняются. Для настройки системы рекомендуется использовать режим *configuration*.

10.7.8 logs

Позволяет просматривать журналы событий Angie ADC. Используйте эту команду для получения диагностической информации при обращении в службу технической поддержки.

Синтаксис: **logs <entity> [{follow|transfer}]**.

Поддерживается *фильтрация*.

Значения для <entity>:

aaa-manager	События модуля, отвечающего за аутентификацию и авторизацию внешних пользователей Angie ADC
adc-gslb	События модуля GSLB (глобальная балансировка)
adc-server	События модуля, обеспечивающего выполнение всего предоставляемого функционала Angie ADC (входная точка)
adc-system	События модуля, отвечающего за выполнение системных команд Angie ADC
adc-tracker	События модуля, отвечающего за RHI-функционал
auth-dataplane	События модуля, отвечающего за <i>аутентификацию клиентов</i> в балансировщике нагрузки
certificate-manager	События сервиса управления сертификатами
ctrl-access	События из access_log модуля, отвечающего за кластерное взаимодействие
ctrl-error	События из error_log модуля, отвечающего за кластерное взаимодействие
gslb-component	События модуля конфигурации GSLB
kern	События ядра
lb-access	События из access_log балансировщика нагрузки
lb-error	События из error_log балансировщика нагрузки
lb-ls	Позволяет получить список журналов балансировщика нагрузки (файлы должны иметь расширение .log) Пример: <pre>\$\$ logs lb-ls access.log error.log healthmonitoring.log \$\$</pre>
mgmt-access	События из access_log модуля, отвечающего за управление Angie ADC
mgmt-error	События из error_log модуля, отвечающего за управление Angie ADC
session-store	События модуля, отвечающего за хранение sticky-сессий в паре высокой доступности
snmp-manager	События модуля SNMP
storage	События из storage_log модуля, отвечающего за управление Angie ADC
supervisor	События модуля, отвечающего за управление кластеризацией Angie ADC
upgrade	События обновления Angie ADC
view <filename>	Позволяет просмотреть произвольный журнал событий, например, добавленный вручную в настройках конфигурации балансировщика. Файл должен находиться в папке /var/log/angie-lb/ и иметь расширение .log. Пример: <pre>\$\$ logs view error.log</pre>

Дополнительные параметры:

follow	Вывод записей в режиме реального времени.
transfer	Копирование записей в директорию transfer.

10.7.8.1 Фильтрация вывода

Вывод записей поддерживает фильтрацию через pipe. Допускается использование нескольких фильтров в виде:

```
command | filter1 | filter2 | ... | filterN
```

Поддерживаются следующие фильтры:

- **grep** — фильтрация по подстроке;
- **exclude** — исключение строк, содержащих указанную подстроку;
- **include** — вывод только строк, содержащих указанную подстроку;
- **head** — вывод первых N строк;
- **tail** — вывод последних N строк.

grep

Синтаксис: **grep -i -o -v <pattern>**

Параметры:

- **-i** — игнорировать регистр;
- **-o** — показать только совпавшую часть вместо всей строки;
- **-v** — инвертировать результат;
- **<pattern>** — подстрока для поиска.

Пример:

```
$$ logs lb-access | grep GET
127.0.0.1 - - [29/Aug/2025:14:34:03 +0300] "GET /metrics HTTP/1.1" 200 1800 "-"
↪ "Prometheus/" "-"
$$ logs lb-access | grep -v GET
2025/08/29 10:19:42 [notice] 976#976: signal 29 (SIGIO) received
$$ logs lb-access | grep -i get
127.0.0.1 - - [29/Aug/2025:14:34:03 +0300] "GET /metrics HTTP/1.1" 200 1800 "-"
↪ "Prometheus/" "-"
$$ logs lb-access | grep -o GET
GET
```

include

Синоним **grep <pattern>**.

Синтаксис: **include <pattern>**.

Параметры:

- **<pattern>** — подстрока для поиска.

Пример:

```
$$ logs lb-access | include GET
127.0.0.1 - - [29/Aug/2025:14:34:03 +0300] "GET /metrics HTTP/1.1" 200 1800 "-"
↪ "Prometheus/" "-"
```

exclude

Синоним `grep -v <pattern>`.

Синтаксис: `exclude <pattern>`.

Параметры:

- `<pattern>` — подстрока для поиска.

Пример:

```
$$ logs lb-error | exclude exit
2025/08/29 10:19:42 [notice] 976#976: signal 29 (SIGIO) received
```

head

Вывод первых N строк.

Синтаксис: `head <n>`.

Параметры:

- `<n>` — количество строк.

Пример:

```
$$ logs lb-access | head 2
127.0.0.1 - - [28/Aug/2025:14:53:48 +0300] "GET /metrics HTTP/1.1" 200 1798 "-"
↪ "Prometheus/" "-"
127.0.0.1 - - [28/Aug/2025:14:53:58 +0300] "GET /metrics HTTP/1.1" 200 1798 "-"
↪ "Prometheus/" "-"
```

tail

Вывод последних N строк.

Синтаксис: `tail <n>`.

Параметры:

- `<n>` — количество строк.

Пример:

```
$$ logs lb-access | tail 2
127.0.0.1 - - [28/Aug/2025:14:53:48 +0300] "GET /metrics HTTP/1.1" 200 1798 "-"
↪ "Prometheus/" "-"
127.0.0.1 - - [28/Aug/2025:14:53:58 +0300] "GET /metrics HTTP/1.1" 200 1798 "-"
↪ "Prometheus/" "-"
```

10.7.9 modules

Контекст `modules` позволяет просматривать информацию о динамических модулях Angie ADC.

<i>info</i>	Получение информации о динамическом модуле Angie ADC
<i>list</i>	Получение списка доступных динамических модулей Angie ADC
<code>modules</code>	Вход в контекст

10.7.9.1 info

Получение информации о динамическом модуле Angie ADC.

Синтаксис: `info <module>`.

Параметры:

- `<module>` — имя модуля.

10.7.9.2 list

Получение списка доступных динамических модулей Angie ADC.

Синтаксис: `list`.

10.7.10 ps

Отображение информации о запущенных процессах в системе. Показывает список активных процессов с различной степенью детализации в зависимости от используемых параметров.

Синтаксис: `ps [-f] [-p <pid>] [-o <format>] [-c <command>] [-s <sortspec>] [-t] [-w] [-h]`.

Параметры:

- `-f` — полный формат вывода: отображение расширенной информации о процессах.
- `-p <pid>` — отображение информации только о процессе с указанным PID (Process ID).
- `-o <format>` — пользовательский формат вывода. Формат задается как список полей, разделенных запятыми (например: `pid,cmd,mem`). Доступные поля описаны ниже.
- `-c <command>` — поиск и отображение процессов по имени команды (фильтрация по имени процесса).
- `-s <sortspec>` — сортировка вывода по указанным полям (например, по использованию CPU, памяти).
- `-t` — отображение процессов в древовидной структуре: показывает родительно-дочерние связи между процессами.
- `-w` — широкий вывод: позволяет отображать полные команды без обрезания.
- `-h` — не показывать заголовки столбцов в выводе.

Примеры:

- `$$ ps` — список процессов текущего пользователя.
- `$$ ps -p 1234` — информация о процессе с PID 1234.
- `$$ ps -c angie` — поиск процессов `angie`.
- `$$ ps -o pid,ppid,cmd,mem` — пользовательский формат вывода.

Форматы для параметра `-o`:

Ключ	Полное имя	Описание
c	cmd	Простое имя исполняемого файла
C	pcpu	Использование CPU (в процентах)
f	flags	Флаги как в длинном формате (поле F)
g	pgrp	Идентификатор группы процессов
G	tpgid	Идентификатор группы процессов управляющего терминала
j	cutime	Совокупное пользовательское время
J	cstime	Совокупное системное время
k	utime	Пользовательское время
m	min_ft	Количество малых ошибок страниц (minor page faults)
M	maj_ft	Количество больших ошибок страниц (major page faults)
n	cmin_ft	Совокупные малые ошибки страниц
N	cmaj_ft	Совокупные большие ошибки страниц
o	session	Идентификатор сессии
p	pid	Идентификатор процесса
P	ppid	Идентификатор родительского процесса
r	rss	Размер резидентного набора (resident set size) в килобайтах
R	resident	Резидентные страницы
s	size	Размер памяти в килобайтах
S	share	Количество разделяемых страниц
t	tty	Номер устройства управляющего терминала
T	start_time	Время запуска процесса
U	uid	Числовой идентификатор пользователя
u	user	Имя пользователя
v	vsize	Общий размер виртуальной памяти в килобайтах
y	priority	Приоритет планировщика ядра

Примеры:

- `$$ ps -o pid,cmd` — отобразить только PID и команду.
- `$$ ps -o user,pid,pcpu,rmem,cmd` — пользователь, PID, CPU, память, команда.
- `$$ ps -o pid,ppid,cmd,start_time` — PID, PPID, команда, время запуска.

10.7.11 settings

Контекст **settings** позволяет просматривать и изменять настройки Angie ADC.

Поддерживаемые команды:

<i>angie debug</i>	Включение режима отладки для балансировщика нагрузки
<i>no angie debug</i>	Отключение режима отладки
<i>reload</i>	Перезагрузка элементов Angie ADC
settings	Вход в контекст
<i>sysctl</i>	Просмотр параметров ядра Linux в реальном времени
<i>syslog</i>	Управление syslog-серверами

10.7.11.1 angie debug

Включение *режима отладки* для локального балансировщика нагрузки.

Синтаксис: `angie debug`.

10.7.11.2 no angie debug

Отключение режима отладки **angie**.

Синтаксис: **no angie debug**.

10.7.11.3 reload

Перезагрузка элементов Angie ADC.

Синтаксис: **reload {angie|prometheus}**.

Параметры:

- **angie** — перезагрузка **angie-mgmt**.
- **prometheus** — перезагрузка **prometheus**.

10.7.11.4 sysctl

Просмотр конкретного параметра ядра Linux или всех параметров в реальном времени.

Синтаксис: **sysctl show {<variable> | all}**.

10.7.11.5 syslog

Настройка и управление удаленными серверами для экспорта событий Angie ADC по syslog-протоколу.

Синтаксис: **syslog {add <input> | del <uuid> | disable <uuid> | enable <uuid> | list | test | update}**.

Параметры:

- **add** — добавление нового syslog-сервера.

Синтаксис: **syslog add <name> <host> <port> <tcp | udp> [<levels>]**

Параметры:

- **name** — имя сервера.
- **host** — IP-адрес или доменное имя сервера.
- **port** — номер порта (по умолчанию 514).
- **tcp | udp** — используемый транспортный протокол (по умолчанию **udp**).
- **levels** — уровни критичности событий от 0 до 7 (если не указано, будут включены все уровни):
 - * 0 — Emergency: только события, приводящие к аварийной ситуации и полному отказу системы.
 - * 1 — Alert: критические ошибки безопасности или выход из строя важных сервисов.
 - * 2 — Critical: серьезные проблемы, которые требуют немедленного исправления.
 - * 3 — Error: ошибки, влияющие на работу системы, но не приводящие к ее остановке.
 - * 4 — Warning: предупреждения о возможных проблемах.
 - * 5 — Notice: уведомления о событиях.
 - * 6 — Informational: информационные сообщения.
 - * 7 — Debug: включение событий отладки.

Примеры:

- **syslog add rsyslog1 10.0.2.2 514 udp**
- **syslog add rsyslog1 10.0.2.2 514 udp 1 2 3**

- **del** — удаление syslog-сервера по UUID.

Синтаксис: `syslog del <uuid>`

- **disable** — выключение syslog-сервера по UUID.

Синтаксис: `syslog disable <uuid>`

- **enable** — включение syslog-сервера по UUID.

Синтаксис: `syslog enable <uuid>`

- **list** — просмотр списка настроенных syslog-серверов.

Синтаксис: `syslog list`

- **test** — проверка доступности syslog-сервера.

Синтаксис: `syslog test <host> <port> <tcp|udp>`

Параметры:

- **host** — IP-адрес или доменное имя сервера.
- **port** — номер порта.
- **tcp | udp** — используемый транспортный протокол.

- **update** — изменение параметров syslog-сервера. Можно менять только одно поле за раз.

Синтаксис: `syslog update <uuid> {name <name> | host <host> | port <port> | protocol <tcp|udp> | levels: [0..7]}`

Параметры:

- **name** — имя syslog-сервера.
- **host** — IP-адрес или доменное имя syslog-сервера.
- **port** — номер порта.
- **tcp | udp** — транспортный протокол.
- **levels** — уровень критичности событий от 0 до 7 (если не указано, будут включены все уровни):
 - * 0 — Emergency: только события, приводящие к аварийной ситуации и полному отказу системы.
 - * 1 — Alert: критические ошибки безопасности или выход из строя важных сервисов.
 - * 2 — Critical: серьезные проблемы, которые требуют немедленного исправления.
 - * 3 — Error: ошибки, влияющие на работу системы, но не приводящие к ее остановке.
 - * 4 — Warning: предупреждения о возможных проблемах.
 - * 5 — Notice: уведомления о событиях.
 - * 6 — Informational: информационные сообщения.
 - * 7 — Debug: включение событий отладки.

Примеры:

- * `syslog update <uuid> levels 1 3` — выставляет только указанные уровни критичности событий.
- * `syslog update <uuid> levels` — выставляет все уровни критичности событий.

10.7.12 system

Контекст **system** для управления системой Angie ADC.

Поддерживаемые команды:

<i>ntp</i>	Просмотр статуса NTP
<i>reboot</i>	Перезагрузка системы Angie ADC
<i>stat</i>	Просмотр статистики системы Angie ADC
system	Вход в контекст
<i>time</i>	Просмотр и установка системного времени
<i>timezone</i>	Просмотр часового пояса системы
<i>upgrade</i>	Обновление Angie ADC
<i>version</i>	Просмотр версии Angie ADC

10.7.12.1 ntp

Просмотр статуса NTP.

Синтаксис: **ntp**

Примеры:

```
$$ system
(system)$$ ntp
enabled: False
source: None
synchronised: False
time_utc: 2026-01-21 10:40:42
time_local: 2026-01-21 13:40:42
timezone: Europe/Moscow
```

Примечание

Настройка NTP проводится через меню конфигурации (*configuration*).

10.7.12.2 reboot

Перезагрузка системы Angie ADC.

Синтаксис: **reboot** [**force**]

Параметры:

- **force** — принудительная перезагрузка.

10.7.12.3 stat

Просмотр статистики системы Angie ADC.

Синтаксис: **stat** [**count** <n>] [**delay** <m>]

Параметры:

- **count** <n> — количество обновлений статистики;
- **delay** <m> — задержка между обновлениями статистики в секундах.

10.7.12.4 time

Просмотр и установка системного времени.

Синтаксис: `time {set <YYYY-MM-DD HH:MM:SS>}`.

Подкоманды:

- `set <YYYY-MM-DD HH:MM:SS>` — установка системного времени в указанном формате.

Примеры:

- `(system)$$ time` — показать текущее время.
- `(system)$$ time set 2025-01-15 14:30:00` — установить время на 15 января 2025, 14:30:00.

10.7.12.5 timezone

Просмотр часового пояса системы. Часовой пояс определяет смещение локального времени относительно UTC.

Синтаксис: `timezone`.

Примеры:

- `(system)$$ timezone` — показать текущий часовой пояс.

Примечание

Настройка `timezone` проводится через меню конфигурации (*configuration*).

10.7.12.6 upgrade

Обновление Angie ADC.

Синтаксис: `upgrade <filename>`

Параметры:

- `<filename>` — имя файла.

10.7.12.7 version

Просмотр версии Angie ADC.

Синтаксис: `version [detail]`

Параметры:

- `detail` — подробный вывод версии.

10.7.13 transfer

Контекст **transfer** для работы с директорией **transfer**.

Поддерживаемые команды:

<i>edit</i>	Редактирование файла в текущей директории.
<i>ls</i>	Получение списка файлов в директории transfer
<i>rm</i>	Удаление файла в директории transfer
transfer	Вход в контекст
<i>view</i>	Просмотр содержимого файла в директории transfer

10.7.13.1 edit

Позволяет редактировать файлы в текущей директории.

Синтаксис: `edit [<filename>]`.

Параметры:

- `<filename>` — имя файла, который необходимо отредактировать (обязательный параметр).

10.7.13.2 ls

Получение списка файлов в директории `transfer`.

Синтаксис: `ls`.

10.7.13.3 rm

Удаление файла в директории `transfer`.

Синтаксис: `rm <filename>`.

Параметры:

- `<filename>` — имя файла.

10.7.13.4 view

Просмотр содержимого файла в директории `transfer`.

Синтаксис: `view <filename> [page]`.

Параметры:

- `<filename>` — имя файла;
- `page` — вывод записей в режиме постраничного просмотра.

10.7.14 vtysh

Переход к настройкам маршрутизации по протоколам BGP и OSPF.

Синтаксис: `vttysh`.

Подробнее о настройке BGP и OSPF см. *BGP-маршрутизация* и *OSPF-маршрутизация*.

10.7.15 web

Контекст `web` для настройки веб-интерфейса Angie ADC.

Поддерживаемые команды:

<code>web</code>	Вход в контекст
<code>upload</code>	Загрузка сертификата управления и ключа
<code>tls</code>	Включение / выключение HTTPS

10.7.15.1 upload

Загрузка сертификата и ключа для управления Angie ADC.

Примечание

Для работы с сертификатами необходимо предварительно создать кластер Angie ADC. Подробнее см. в статье *Создание кластера*.

Синтаксис: `upload`.

Для загрузки сертификата или ключа необходимо вставить текст сертификата или ключа в строку, затем вставить пустую строку или написать `END` в новой строке.

При загрузке ключа и сертификата проверяется их корректность. В случае некорректного значения сертификат или ключ не будет сохранен.

10.7.15.2 `tls`

Включение / выключение HTTPS и просмотр информации о статусе и используемом сертификате.

Синтаксис: `tls {enable | disable | info | status}`.

Параметры:

- `enable` — включение HTTPS;
- `disable` — выключение HTTPS;
- `info` — вывод информации о загруженном сертификате управления;
- `status` — вывод статуса HTTPS (включен или выключен).

10.8 Веб-интерфейс Angie ADC

Веб-интерфейс Angie ADC представляет собой экран, в левой части которого отображается панель навигации, а в правой части – выбранный раздел.

В разделах ниже описания элементов интерфейса даны в порядке сверху вниз.

- Экран "Вход"
- Элементы настройки веб-интерфейса
- Безопасность учетной записи
- Раздел "Система"
- Вкладка "Конфигурация"
- Вкладка "Пользователи"
- Вкладка "Аутентификация и авторизация"
- Вкладка "Парольная политика"
- Вкладка "Сертификаты"
- Вкладка "Пользователи SNMP"
- Вкладка "Резервное копирование и восстановление"
- Вкладка "NTP"
- Вкладка "Логирование"
- Вкладка "О решении"
- Раздел "Мониторинг"
- Вкладка "Балансировщик"
- Вкладка "Система"
- Раздел "Управление трафиком"
- Вкладка "Балансировщик нагрузки"
- Окно "Управление доступами"
- Окно "История"

- Окно "Настройки резервного копирования для балансировщика нагрузки"
- Вкладка "Глобальная балансировка"
- Вкладка "Аутентификация клиентов"
- Вкладка "Конфигурация RHI"
- Раздел "Высокая доступность и кластеризация"
- Вкладка "Высокая доступность"
- Вкладка "Кластер"
- Раздел "Диагностика"
- Вкладка "Сеть"

10.8.1 Экран "Вход"

Вход

Логин

Пароль

Тип аутентификации

Локальная

Войти

Чтобы войти в веб-интерфейс и получить доступ к функциям, необходимо выбрать тип аутентификации и ввести логин и пароль. Реквизиты для локальной аутентификации при первом входе предоставляются после покупки решения. Рекомендуется сменить пароль учетной записи после первого входа. Если настроена двухфакторная аутентификация, то необходимо также ввести одноразовый код из приложения-аутентификатора.

Примечание

При двух неудачных попытках ввода логина и пароля в течение одного часа пользователь будет временно заблокирован.

После пяти неудачных попыток ввода одноразового кода в течение пяти минут пользователь будет заблокирован на пять минут.

Элементы интерфейса:

Логин	Поле для ввода логина
Пароль	Поле для ввода пароля
Тип аутентификации	Раскрывающийся список для выбора типа аутентификации. Доступные варианты: • Локальная; • LDAP (отображается, если в веб-интерфейсе заданы соответствующие настройки); • RADIUS (отображается, если в веб-интерфейсе заданы соответствующие настройки); • TACACS (отображается, если в веб-интерфейсе заданы соответствующие настройки). При использовании внешней аутентификации необходимо предварительно подключить сервер аутентификации и настроить авторизацию внешних пользователей.
Войти	Кнопка входа
Одноразовый код	Поле для ввода одноразового кода из приложения-аутентификатора, если настроена двухфакторная аутентификация
Подтвердить	Кнопка подтверждения одноразового кода и входа
Назад	Кнопка перехода к выбору логина и типа аутентификации

10.8.2 Элементы настройки веб-интерфейса

В верхней правой части экрана расположены элементы настройки веб-интерфейса и учетной записи.

Значок "солнце-луна"	Переключатель темного и светлого режимов интерфейса
Инициал пользователя	Открывает контекстное меню пользователя
Безопасность учетной записи	Открывает настройки безопасности текущей учетной записи
Выйти	Выход из текущей учетной записи

10.8.3 Безопасность учетной записи

Настройки текущей учетной записи.

angie-adc1 > [Безопасность учётной записи](#)

Безопасность учётной записи

Изменить пароль

После смены пароля ранее выданные сессии будут отозваны.

Требования к новому паролю

От 8 до 64 символов
Пароль не должен совпадать с логином

Текущий пароль *

Новый пароль *

Повторите новый пароль *

Изменить пароль

Управление доступно только для локальных учётных записей. Для подтверждения операций используются текущий пароль и код из приложения аутентификации.

Включить TOTP

Поддерживаются приложения, совместимые со стандартом TOTP, например Яндекс.Ключ и Google Authenticator.

Текущий пароль *

Начать настройку

Отключить TOTP

Если политика требует двухфакторную аутентификацию, сервер не позволит её отключить.

Текущий пароль *

Одноразовый код *

Отключить TOTP

10.8.3.1 Блок "Изменить пароль"

Элементы интерфейса:

Текущий пароль	Поле ввода текущего пароля
Новый пароль	Поле ввода нового пароля. Новый пароль не должен повторять текущий пароль, не должен повторять логин без учета регистра и должен соответствовать <i>требованиям парольной политики</i> .
Повторите новый пароль	Поле для подтверждения нового пароля
Изменить пароль	Кнопка применения изменений. В случае прохождения всех проверок пароль будет изменен и текущая пользовательская сессия будет завершена.

10.8.3.2 Блок "Включить TOTP"

Включение двухфакторной аутентификации.

Элементы интерфейса:

Текущий пароль	Поле ввода текущего пароля
Начать настройку	Кнопка перехода к подключению приложения для аутентификации. Например, можно использовать приложения Яндекс.Ключ, Google Authenticator или Multifactor.
QR-код	QR-код для подключения приложения-аутентификатора на мобильном устройстве
Секрет	Секрет для подключения приложения-аутентификатора на компьютере
Подтвердить и включить	Кнопка применения изменений. Текущая пользовательская сессия будет завершена.

10.8.3.3 Блок "Отключить TOTP"

Выключение двухфакторной аутентификации.

Элементы интерфейса:

Текущий пароль	Поле ввода текущего пароля
Одноразовый код	Поле ввода одноразового кода из подключенного приложения для аутентификации
Отключить TOTP	Кнопка применения изменений. Текущая пользовательская сессия будет завершена.

10.8.4 Раздел "Система"

10.8.4.1 Вкладка "Конфигурация"

adc-psi-2 > Конфигурация

Конфигурация Angie ADC

Hostname (имя устройства)

Сбросить Сохранить

Управление HTTPS

● Выключен

Загрузить сертификат и включить HTTPS

Сетевые интерфейсы
 Кликните по строке интерфейса, чтобы редактировать.

ИМЯ	ТИП	MAC-АДРЕС	IPv4-АДРЕСА	IPv6-АДРЕСА	ШЛЮЗ	DNS-АДРЕСА	ДОМЕНЫ ПОИСКА
enp0s2	DHCP	22:22:0a:15:14:39	10.21.20.57/22	fe80::2022:aff:fe15:1439/64	10.21.20.1	—	—

Блок Hostname

Элементы интерфейса:

Hostname (имя устройства)	Имя устройства Angie ADC
Сбросить	Кнопка сброса введенных данных
Сохранить	Кнопка сохранения нового имени устройства Angie ADC

Блок "Управление HTTPS"

Управление HTTPS

● Включен

Загружен: 05.06.2026

Действует с: 01.01.2026

Истекает: 01.01.2027 (осталось 184 дней)

Загрузить новый сертификат

Выключить HTTPS

В этом блоке можно загрузить сертификат управления (management) и включить HTTPS для веб-интерфейса.

Примечание

Поддерживается загрузка и использование только одного сертификата управления (management). При загрузке нового сертификата предыдущий автоматически удаляется.

Элементы интерфейса:

Статус HTTPS	● Включен — используется защищенное соединение. ● Выключен — соединение не защищено.
Загрузить сертификат и включить HTTPS	Кнопка добавления сертификата управления.
Загрузить новый сертификат	Кнопка загрузки сертификата управления, который будет использован вместо текущего.
Включить HTTPS	Кнопка, включающая использование защищенного соединения.
Выключить HTTPS	Кнопка выключающая использование защищенного соединения.

Окно загрузки сертификата

Загрузить сертификат

Сертификат (PEM)

Вставьте содержимое PEM-сертификата

Приватный ключ (PEM)

Вставьте содержимое PEM-ключа

Отмена

Загрузить

Элементы интерфейса:

Сертификат (PEM)	Поле для ввода содержимого сертификата в формате PEM
Приватный ключ (PEM)	Поле для ввода содержимого приватного ключа в формате PEM
Отмена	Кнопка сброса данных
Загрузить	Кнопка загрузки сертификата

Блок "Сетевые интерфейсы"

Элементы интерфейса:

Имя	Имя сетевого интерфейса (например <code>eth0</code> , <code>ens3</code> , <code>ens6</code>)
Тип	Тип конфигурации интерфейса (<code>Static</code> или <code>DHCP</code>)
MAC-адрес	MAC-адрес интерфейса (формат: <code>XX:XX:XX:XX:XX:XX</code>)
IPv4-адреса	Массив IPv4-адресов в формате CIDR (например <code>["192.168.1.100/24"]</code>)
IPv6-адреса	Массив IPv6-адресов в формате CIDR (например <code>["fe80::1/64"]</code>)
Шлюз	IP-адрес шлюза по умолчанию (глобальный для всех интерфейсов) или <code>null</code> , если не установлен
DNS-адреса	Массив IP-адресов DNS-серверов (например <code>["8.8.8.8", "8.8.4.4"]</code>)
Домены поиска	Массив доменов поиска DNS (например <code>["example.com", "internal.local"]</code>)

Окно "Изменение параметров сетевого интерфейса"

Редактирование интерфейса

Имя

eth1

MAC-адрес

00:11:22:33:44:66

Тип

Static

Шлюз (IPv4-адрес)

10.10.0.1

IPv4 (IP или CIDR через запятую)

10.10.0.20/24, 10.10.0.21/24

IPv6 (IP или CIDR через запятую)

2001:db8::20/64

DNS (IPv4 через запятую)

9.9.9.9

Домены поиска (через запятую)

dmz.local, svc.local

Отмена

Сохранить

Экран предоставляет возможность изменить параметры сетевого интерфейса.

Имя	Имя сетевого интерфейса (например eth0, ens3, ens6)
MAC-адрес	MAC-адрес интерфейса (формат: XX:XX:XX:XX:XX:XX)
Тип	Тип конфигурации интерфейса (Static или DHCP)
 Примечание Если для интерфейса был указан способ получения настроек по DHCP, то в окне редактирования будет отображаться подсказка с текущими DHCP-параметрами интерфейса (Текущее состояние).	
IPv4-адреса	Массив IPv4-адресов в формате CIDR (например ["192.168.1.100/24"])
IPv6-адреса	Массив IPv6-адресов в формате CIDR (например ["fe80::1/64"])
Шлюз	IP-адрес шлюза по умолчанию (глобальный для всех интерфейсов) или null, если не установлен
DNS-адреса	Массив IP-адресов DNS-серверов (например ["8.8.8.8", "8.8.4.4"])
Домены поиска	Массив доменов поиска DNS (например ["example.com", "internal.local"])
Отмена	Кнопка сброса данных
Сохранить	Кнопка сохранения изменений

10.8.4.2 Вкладка "Пользователи"

Angie ADC > Пользователи

Пользователи

+ Добавить пользователя

Показывать удаленных пользователей ☒

Логин	Имя	Фамилия	Роль	Статус	Комментарий
admin			Администратор	✓ активный	 
Ivan_Ivanov	Иван	Иванов	Супервизор	↻ неактивный	 
 					

Таблица на этой вкладке содержит список пользователей, зарегистрированных в Angie ADC, и позволяет управлять как их составом, так и свойствами отдельных пользователей.

Элементы интерфейса:

Добавить пользователя	Открывает экран добавления нового пользователя
Показывать удаленных пользователей	Отображает удаленных пользователей в списке пользователей; по умолчанию переключатель находится в положении "Выключено"
Логин	Учетная запись пользователя в системе; щелчок этой ссылки открывает экран изменения данных пользователя
Имя	Имя пользователя
Фамилия	Фамилия пользователя
UUID	Уникальный идентификатор пользователя
Роль	Роль пользователя в системе (Администратор, Супервизор, Оператор)
Статус	Статус пользователя в системе (активный, неактивный, удален)
Комментарий	Примечание
Значок редактирования	Открывает экран изменения данных пользователя
Значок удаления	Удаляет запись о пользователе

Окно "Добавление пользователя"

Angie ADC > Пользователи > [Добавить пользователя](#)

Добавление нового пользователя

Заполните все обязательные поля

Логин *

Пароль *

Имя *

Фамилия *

Роль *

выберите роль

⌵

Комментарий

Сбросить

Добавить

Экран предоставляет возможность добавить запись о пользователе в системе.

Элементы интерфейса:

Логин	Логин пользователя в системе
Пароль	Пароль пользователя в системе (проверьте вводимое значение!)
Имя	Имя пользователя
Фамилия	Фамилия пользователя
Роль	Роль пользователя в системе (Администратор, Оператор или Супервизор)
Комментарий	Примечание
Сбросить	Кнопка сброса данных
Добавить	Кнопка добавления пользователя

Окно "Изменение данных пользователя"

Экран предоставляет возможность изменить существующую запись о пользователе в системе.

Элементы интерфейса аналогичны тем, что представлены на экране "Добавление нового пользователя", за исключением кнопки "Сохранить", изменяющей данные пользователя.

10.8.4.3 Вкладка "Аутентификация и авторизация"

Экран предоставляет возможность включить и настроить внешнюю аутентификацию, а также задать роли для внешних пользователей.

Блок "Внешняя аутентификация"

Элементы интерфейса:

Внешняя аутентификация RADIUS TACACS LDAP	Переключатель включает или выключает использование внешней аутентификации; по умолчанию находится в положении "Выключено". Позволяет выбрать тип провайдера внешней аутентификации.
 Примечание Одновременно поддерживается использование только одного провайдера. При добавлении нового провайдера настройки предыдущего провайдера сбрасываются.	

RADIUS

Angie ADC > Аутентификация и авторизация

Аутентификация и авторизация

☒ Внешняя аутентификация

RADIUS
 TACACS
 LDAP

Сервер

Порт

Протокол

Метод аутентификации

NAS-идентификатор

Секрет

Сохранить

Сервер	IP-адрес или доменное имя сервера внешней аутентификации.
Порт	Порт подключения к серверу (по умолчанию 1812).
Протокол	Протокол передачи данных (по умолчанию UDP).
Метод аутентификации	На данный момент поддерживается метод PAP.
NAS-идентификатор	Идентификатор NAS (Network Access Server), передаваемый серверу RADIUS.
Секрет	Общий секрет для шифрования паролей.
Сохранить	Кнопка сохранения изменений.

TACACS

Angie ADC > Аутентификация и авторизация

Аутентификация и авторизация

☒ **Внешняя аутентификация**
RADIUS
TACACS
LDAP

Сервер

Порт

Секрет

Сохранить

Сервер	IP-адрес или доменное имя сервера внешней аутентификации.
Порт	Порт подключения к серверу (по умолчанию 49).
Секрет	Общий секрет для шифрования паролей.
Сохранить	Кнопка сохранения изменений.

LDAP

☒ **Внешняя аутентификация**
RADIUS
TACACS
LDAP

Параметры подключения LDAP-сервера

Сервер

Порт

Режим защиты соединения

Без TLS

LDAPS

StartTLS

☐ Проверять TLS сертификат

DN сервисной учетной записи

Пароль сервисной учетной записи

Поиск пользователя

Базовый DN пользователя

Идентификатор логина

Фильтр поиска

Сохранить

Сервер	IP-адрес или доменное имя сервера внешней аутентификации.
Порт	Порт подключения к серверу (по умолчанию 389).
Режим защиты соединения	Указывает, будет ли защищено соединение. Доступные варианты: • Без TLS (по умолчанию); • LDAPS (использование TLS при подключении); • StartTLS (использование механизма StartTLS при подключении).
Проверять TLS-сертификат	Флажок доступен при выборе режима защиты соединения LDAPS или StartTLS. По умолчанию флажок установлен.
DN сервисной учетной записи	Логин сервисной учетной записи, от имени которой система подключается к LDAP.
Пароль сервисной учетной записи	Пароль сервисной учетной записи для подключения к LDAP-серверу.
Базовый DN пользователя	Задаёт ветку на LDAP-сервере, в которой система будет искать пользователя, например <code>ou=users,dc=example,dc=com</code> .
Идентификатор логина	Поле в LDAP, которое содержит логин, например <code>uid</code> , <code>sAMAccountName</code> или <code>userPrincipalName</code> .
Фильтр поиска	LDAP-фильтр для поиска пользователя, например <code>(&(objectClass=user)(sAMAccountName=%s))</code> .
Сохранить	Кнопка сохранения изменений.

Блок "Внешняя авторизация"

Элементы интерфейса:

Внешняя авторизация	Переключатель включает или выключает использование внешней авторизации; по умолчанию находится в положении "Выключено".
RADIUS TACACS LDAP	Позволяет выбрать тип провайдера внешней авторизации.

Примечание

Одновременно поддерживается использование только одного провайдера. При добавлении нового провайдера настройки предыдущего провайдера сбрасываются.

LDAP

Внешняя авторизация

RADIUS

TACACS

LDAP

Подключение LDAP-сервера

Сервер

Порт

FQDN или IPv4/IPv6

389

DN сервисной учетной записи

cn=readonly,dc=example,dc=local

Режим защиты соединения

Без TLS

LDAPS

StartTLS

Пароль пользователя bind

Введите пароль

☐ Проверять TLS сертификат

Поиск пользователей

Базовый DN пользователей

Фильтр поиска пользователей

ou=users,dc=example,dc=local

(uid={username})

Поиск групп

Базовый DN групп

Фильтр поиска групп

ou=groups,dc=example,dc=local

(member={dn})

Мэппинг ролей LDAP

Правила мэппинга

Правила мэппинга LDAP-групп и ролей не добавлены.

+

Добавить правило

Сохранить

10.8. Веб-интерфейс Angie ADC

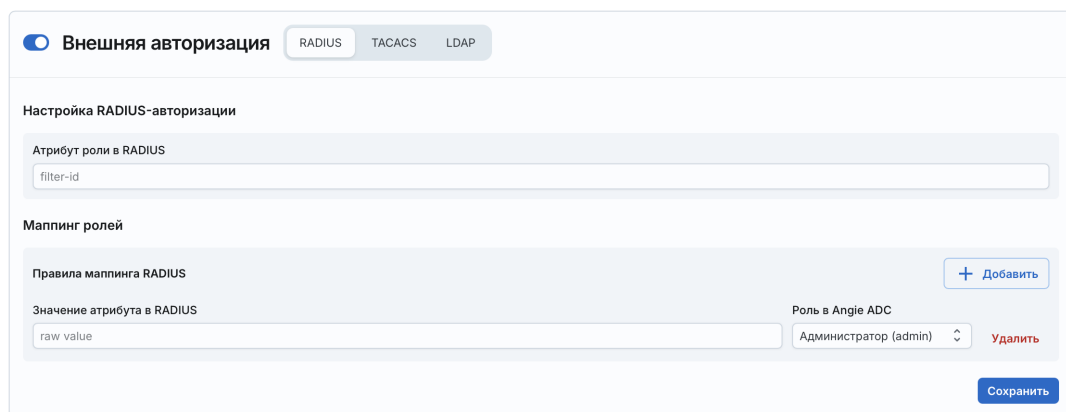
809

Блок Подключение LDAP-сервера	Задаёт настройки подключения к LDAP-серверу.
Сервер	IP-адрес или доменное имя сервера внешней авторизации.
Порт	Порт подключения к LDAP-серверу (по умолчанию 389).
Режим защиты соединения	Доступные варианты: • Без TLS (по умолчанию, незащищенное соединение); • LDAPS (использование TLS при подключении); • StartTLS (использование механизма StartTLS при подключении).
Проверять TLS-сертификат	Флажок доступен при выборе режима защиты соединения LDAPS или StartTLS. По умолчанию флажок установлен.
DN сервисной учетной записи	Логин сервисной учетной записи, от имени которой система подключается к LDAP. Можно использовать учетную запись администратора LDAP-сервера, например <code>cn=admin,dc=example,dc=com</code> .
Пароль пользователя bind	Пароль сервисной учетной записи для подключения к LDAP-серверу.
Блок Поиск пользователя	Задаёт настройки поиска пользователей и групп на LDAP-сервере.
Базовый DN пользователей	Задаёт ветку на LDAP-сервере, в которой система будет искать пользователей, например <code>ou=users,dc=example,dc=com</code> .
Фильтр поиска пользователей	LDAP-фильтр для поиска пользователя, например <code>(uid=%s)</code> .
Блок Поиск групп	Задаёт настройки поиска групп на LDAP-сервере.
Базовый DN групп	Задаёт ветку на LDAP-сервере, в которой система будет искать группы пользователей, например <code>ou=groups,dc=example,dc=com</code> .
Фильтр поиска групп	LDAP-фильтр для поиска групп, которым принадлежит пользователь, например <code>(member=%s)</code> .
Блок Маппинг ролей LDAP	Задаёт правила соответствия групп на LDAP-сервере и ролей в Angie ADC.
Добавить правило	Позволяет задать правило соответствия LDAP-группы конкретной роли в Angie ADC.
DN группы	DN группы пользователей на LDAP-сервере, например <code>cn=adc-admins,ou=groups,dc=example,dc=com</code> или <code>cn=adc-supervisors,ou=groups,dc=example,dc=com</code> .
Роль	Роль в Angie ADC, которая будет назначена пользователю, входящему в указанную LDAP-группу. Доступные роли: • Администратор: обладает полными правами на все действия в системе; • Супервизор: имеет ограниченные права, может только просматривать информацию в системе, например, проводить аудит; • Оператор: имеет ограниченные права, может редактировать файлы конфигурации балансировщика, к которым ему предоставлен доступ, и просматривать статистику зон, к которым ему предоставлен доступ. Доступы предоставляет администратор. Остальные настройки Angie ADC оператору не доступны и не отображаются в веб-интерфейсе.
Удалить	Кнопка удаления правила соответствия LDAP-группы конкретной роли в Angie ADC.
Сохранить	Кнопка сохранения настроек.

TACACS+

Блок Подключение TACACS-сервера	Задаёт настройки подключения к TACACS-серверу.
Сервер	IP-адрес или доменное имя сервера внешней аутентификации.
Порт	Порт подключения к серверу (по умолчанию 49).
Секрет	Общий секрет для шифрования паролей.
Подключаться с секретом	Указывает, будет ли использоваться секрет при подключении к серверу. По умолчанию флажок установлен.
Блок Настройка TACACS-авторизации	Задаёт настройки определения ролей пользователей на TACACS-сервере.
Атрибуты ролей в TACACS (по приоритету)	Атрибуты ролей пользователя, заданные на TACACS-сервере, например <code>priv-lvl</code> , <code>role</code> . По этим атрибутам Angie ADC будет определять роль для пользователя.
Параметры AUTHOR-запроса	Сервис и протокол, которые будут использоваться в запросе на авторизацию. Например, для TACACS+ рекомендуется указать <code>service=shell</code> и <code>protocol=ip</code> . Параметры зависят от выбранного сервера и не являются обязательными.
Блок Маппинг ролей	Задаёт правила соответствия атрибутов пользователей TACACS-сервера и ролей в Angie ADC.
Добавить правило	Позволяет задать правило соответствия атрибута пользователя TACACS-сервера конкретной роли в Angie ADC.
Удалить	Кнопка удаления правила.
Значение атрибута в TACACS	Значения атрибутов, которые будут соответствовать той или иной роли. Например <code>priv-lvl=15</code> или <code>role=adc-admin</code> .
Роль в Angie ADC	Роль в Angie ADC, которая будет назначена пользователю, имеющему указанное значение атрибута. Доступные роли: ● Администратор: обладает полными правами на все действия в системе; ● Супервизор: имеет ограниченные права, может только просматривать информацию в системе, например, проводить аудит; ● Оператор: имеет ограниченные права, может редактировать файлы конфигурации балансировщика, к которым ему предоставлен доступ, и просматривать статистику зон, к которым ему предоставлен доступ. Доступы предоставляет администратор. Остальные настройки Angie ADC оператору не доступны и не отображаются в веб-интерфейсе.
Сохранить	Кнопка сохранения изменений.

RADIUS



Блок	Настройка RADIUS-авторизации	Задаёт настройки определения ролей пользователей на RADIUS-сервере.
Атрибут роли в RADIUS		Атрибут роли в RADIUS. Допускается использование следующих атрибутов: • Filter-Id — роль пользователя. • Class — группа, к которой относится пользователь. • Service-Type — тип сервиса, по которому определяется роль пользователя (Administrative, Login и др.) • Tunnel-Private-Group-ID — VLAN ID (используется для VLAN-авторизации). • VSA — Vendor-Specific в формате <code>vsa:<vendorID>:<vendorType></code> (например <code>vsa:9:1</code>). По этому атрибуту Angie ADC будет определять роль для пользователя.
Блок	МAPPING ролей	Задаёт правила соответствия атрибутов пользователей RADIUS-сервера и ролей в Angie ADC.
Добавить правило		Позволяет задать правило соответствия атрибута пользователя RADIUS-сервера конкретной роли в Angie ADC.
Удалить		Кнопка удаления правила.
Значение атрибута в RADIUS		Значение атрибута, которое будет соответствовать той или иной роли (например <code>adc-admin</code> для Filter-Id или <code>shell:priv-lvl=15</code> для <code>vsa:9:1</code>).
Роль в Angie ADC		Роль в Angie ADC, которая будет назначена пользователю, имеющему указанное значение атрибута. Доступные роли: • Администратор: обладает полными правами на все действия в системе; • Супервизор: имеет ограниченные права, может только просматривать информацию в системе, например, проводить аудит; • Оператор: имеет ограниченные права, может редактировать файлы конфигурации балансировщика, к которым ему предоставлен доступ, и просматривать статистику зон, к которым ему предоставлен доступ. Доступы предоставляет администратор. Остальные настройки Angie ADC оператору не доступны и не отображаются в веб-интерфейсе.
Сохранить		Кнопка сохранения изменений.

10.8.4.4 Вкладка "Парольная политика"

Angie ADC поддерживает парольную политику для локальных пользователей Angie ADC. Парольная политика начинает действовать сразу при запуске Angie ADC со значениями по умолчанию.

adc-psi-3 > Парольная политика

Парольная политика

Парольная политика

Минимальная длина пароля (символов)

8

Максимальная длина пароля (символов)

64

Срок действия пароля (0 дней - бессрочный пароль)

0

Предупреждать об истечении срока действия пароля (дней)

0

Количество паролей, сохраняемых в истории

5

Допустимое количество неверных попыток входа за указанный интервал

2

Интервал подсчета неверных попыток входа (минут)

60

Длительность блокировки входа (минут)

1

Обязательно использовать следующие символы

Например: !@#\$\$%

До 63 уникальных печатных ASCII-символов без пробелов

☐ Требовать заглавную букву
☐ Требовать цифру
☐ Требовать смену при первом входе
☐ Требовать строчную букву

☐ Требовать двухфакторную аутентификацию
☐ Проверять историю паролей
☐ Разблокировка только администратором

Сохранить политику

Сброс истории паролей

☒ Один пользователь
☐ Все пользователи

Пользователь

Выберите пользователя

Сбросить историю

Angie ADC поддерживает парольную политику для локальных пользователей Angie ADC. Парольная политика начинает действовать сразу при запуске Angie ADC со значениями по умолчанию.

Элементы интерфейса:

10.8. Веб-интерфейс Angie ADC

813

Минимальная длина пароля (символов)	Допустимый диапазон от 1 до 100 символов. По умолчанию 8 символов.
Максимальная длина пароля	Максимальная длина пароля должна быть не больше 500 символов. По умолчанию 64 символа.
Срок действия пароля	Значение по умолчанию 0 — срок действия пароля не ограничен (бесечно). Срок действия пароля может быть не более 1000 дней. При указании срока действия пароля система будет напоминать о скором окончании срока за 7 дней до его истечения.
Предупреждать об истечении срока действия пароля	Значение по умолчанию 0 — предупреждение отключено; любое другое значение — количество дней, прошедшее с последней смены пароля, после которого система будет напоминать о необходимости сменить пароль. Максимальное значение 360 дней.
Количество паролей, сохраняемых в истории	Значение определяет, сколько паролей подряд не может повторяться. Значение 0 — количество хранимых паролей не ограничено. По умолчанию 5.
Интервал подсчета неверных попыток входа	Поле доступно, если установлен флажок Проверять историю паролей . Указывает, за какое время система будет подсчитывать попытки входа пользователя. По умолчанию 60 минут. Допустимый диапазон от 1 до 1000 минут.
Допустимое количество неверных попыток входа за указанный интервал	Разрешенное количество попыток входа за время, указанное в Интервал подсчета неверных попыток входа . По умолчанию 2. Допустимый диапазон от 1 до 1000 попыток.
 Примечание При сбросе пароля администратором счетчик неправильных попыток ввода пароля тоже сбрасывается.	
Длительность блокировки входа	Указывает, на какое время блокировать пользователя, превысившего количество разрешенных попыток входа. По умолчанию 1 минута. Допустимый диапазон от 1 до 1000 минут.
Обязательно использовать следующие символы	Дополнительные требования к паролю. Можно указать до 63 уникальных печатных ASCII-символов без пробелов. По умолчанию поле не задано.
Требовать заглавную букву	Дополнительные требования к паролю: наличие заглавной буквы (A-Z). По умолчанию флажок снят.
Требовать цифру	Дополнительные требования к паролю: наличие цифры (0-9). По умолчанию флажок снят.
Требовать смену при первом входе`	Дополнительные требования к паролю: смена предустановленного пароля при первом входе пользователя в веб-интерфейс. По умолчанию флажок снят.
Требовать строчную букву	Дополнительные требования к паролю: наличие строчной буквы (a-z). По умолчанию флажок снят.
Требовать двухфакторную аутентификацию	При входе в веб-интерфейс пользователям будут предложены QR-код и секрет для подключения приложения-аутентификатора и дальнейшей аутентификации в системе через одноразовый пароль из этого приложения. Например, можно использовать приложения-аутентификаторы Яндекс.Ключ, Google Authenticator или Multifactor. По умолчанию флажок снят.
 Важно После пяти неудачных попыток ввода одноразового кода в течение пяти минут пользователь будет заблокирован на пять минут.	
Проверять историю паролей	Включает хранение истории паролей. По умолчанию флажок снят.
Разблокировка только администратором	Указывает, что заблокированных пользователей может разблокировать только администратор вручную. Если флажок установлен, поле Длительность блокировки входа не учитывается. По умолчанию флажок снят.

10.8.4.5 Вкладка "Сертификаты"

Загрузка пользовательских сертификатов.

adc-psi-2 > Сертификаты

Сертификаты + Загрузить сертификат

Имя	Действителен	Загружен	Пути	
сертификат	30.06.2026 — 30.06.2027	01.07.2026	/etc/angie-lb/crt/active/сертификат.crt /etc/angie-lb/crt/active/сертификат.key	
о	30.06.2026 — 30.06.2027	01.07.2026	/etc/angie-lb/crt/active/o.crt /etc/angie-lb/crt/active/o.key	

Элементы интерфейса:

Таблица загруженных
сертификатов

- **Имя** — произвольное имя сертификата, отображаемое в системе.
- **Действителен** — срок действия сертификата.
- **Загружен** — дата загрузки сертификата.
- **Пути** — пути к сертификату и приватному ключу.
- **Значок копирования** — позволяет скопировать путь к сертификату или приватному ключу для использования в конфигурации балансировщика нагрузки (наличие значка зависит от браузера и протокола подключения).
- **Значок удаления** — удаляет сертификат.

Загрузить
сертификат

Кнопка загрузки сертификата

Окно загрузки сертификата

Загрузить сертификат

Имя сертификата

my-cert

Сертификат

Вставьте содержимое сертификата

Приватный ключ

Вставьте содержимое ключа

Отмена

Загрузить

Элементы интерфейса:

Имя сертификата	Произвольное имя сертификата, которое будет отображаться в системе.
Сертификат	Поле для ввода содержимого файла сертификата.
Приватный ключ	Поле для ввода содержимого файла приватного ключа.
Отмена	Кнопка сброса данных
Загрузить	Кнопка загрузки сертификата.

10.8.4.6 Вкладка "Пользователи SNMP"

При использовании протокола SNMPv3 необходимо предварительно добавить пользователя SNMP в веб-интерфейсе Angie ADC, чтобы предоставить безопасный доступ для системы мониторинга к SNMP-метрикам.

angie-adc1 > Пользователи SNMP

Пользователи SNMP + Добавить пользователя SNMP

Имя пользователя	Протокол аутентификации	Протокол шифрования	Дата создания	
first-user	SHA-256	AES-256	30.07.2026 18:46:21	 
second-user	SHA-256	AES-128	30.07.2026 18:46:38	 
third-user	SHA-256	AES-256	30.07.2026 18:46:51	 

Элементы интерфейса:

Добавить пользователя SNMP	Переход к добавлению нового пользователя SNMP
Имя пользователя	Уникальный идентификатор пользователя SNMPv3, например zabbix-user
Протокол аутентификации	Всегда SHA256
Протокол шифрования	AES128 или AES256
Дата создания	Дата и время создания записи
Значок редактирования	Редактирование пользователя SNMP
Значок удаления	Удаление пользователя SNMP

Окно "Добавление пользователя"

Добавить пользователя SNMP
×

Имя пользователя *

Секрет аутентификации *

Протокол шифрования *

выберите протокол

^
v

Секрет шифрования *

Отмена
Добавить

Экран предоставляет возможность добавить запись о пользователе в системе.

Элементы интерфейса:

Имя пользователя	Уникальный идентификатор пользователя SNMPv3, например zabbix-user . Имя пользователя должно начинаться с буквы и может содержать только латинские буквы, цифры, подчеркивание и дефис.
Секрет аутентификации	Зашифрованный секрет аутентификации (AES-256-GCM, Base64). Секрет аутентификации должен содержать не менее 8 символов.
Протокол шифрования	Поддерживаются AES128 или AES256.
Секрет шифрования	Зашифрованный секрет шифрования (AES-256-GCM, Base64). Секрет шифрования должен содержать не менее 8 символов.
Добавить	Кнопка сохранения нового пользователя
Отмена	Кнопка отмены изменений

10.8.4.7 Вкладка "Резервное копирование и восстановление"

Angie ADC > Резервное копирование и восстановление узла

Резервное копирование и восстановление узла

Копия еще не создана

Состояние на: 18.08.2026, 17:25

↺↻

angie-adc1

10.0.2.15 · VA angie-va-base-20260810 / ADC v1.1-SNAPSHOT-4f93c9c

Создание резервной копии

Восстановление из копии

Будет создана полная резервная копия текущего узла

Создать копию

Angie ADC > Резервное копирование и восстановление узла

Резервное копирование и восстановление узла

Файл восстановления еще не загружен

Состояние на: 18.08.2026, 17:32

↺↻

angie-adc1

10.0.2.15 · VA angie-va-base-20260810 / ADC v1.1-SNAPSHOT-4f93c9c

Создание резервной копии

Восстановление из копии

↑

Перетащите архив .tar сюда или нажмите для выбора файла

Элементы интерфейса:

Статус резервного копирования и восстановления	Здесь отображаются статус текущей копии, этапы резервного копирования и восстановления из резервной копии. В блоке Создание резервной копии могут быть следующие статусы: • Копия еще не создана. • Создание копии...; • Есть резервная копия; • Не удалось создать резервную копию. В блоке Восстановление из копии могут быть следующие статусы: • Файл восстановления еще не загружен. • Готов к восстановлению; • Проверка архива; • Не удалось проверить архив; • Запущено восстановление из резервной копии; • Восстановление завершено; • Восстановление не удалось; • Восстановлен из копии.
Блок "Текущий узел"	Имя узла (hostname) и информация об узле: • IP-адрес; • версия VA; • версия Angie ADC.
Кнопка обновления	Обновляет данные об узле и его копии
Состояние на	Дата и время последнего обновления данных об узле и копии
Создание резервной копии	Блок управления резервным копированием текущего узла.
Создать копию	Кнопка, запускающая создание полной резервной копии текущего узла (архив <code>.tar</code>)
Восстановление из копии	Блок управления восстановлением текущего узла из копии. Содержит поле для загрузки архива <code>.tar</code> для восстановления текущего узла из этой копии.

Блок "Создание резервной копии"

Angie ADC > Резервное копирование и восстановление узла

Резервное копирование и восстановление узла

Есть резервная копия

Состояние на: 14.08.2026, 13:54

↻

adc-psi-3
10.21.20.36 - VA angle-va-base-20260810 / ADC v1.1

Создание резервной копии

Восстановление из копии

↻ Создать новую копию

⬇ Скачать

✓ Применить резервную копию

Дата создания
14.08.2026 13:54

Размер
1.1 МБ

Версия VA / Angie ADC
angle-va-base-20260810 / v1.1

Узел-источник
adc-psi-3

Элементы интерфейса:

Создать новую копию	Кнопка обновления резервной копии текущего узла.
Скачать	Кнопка выгрузки архива <code>.tar</code> с текущей резервной копией.
Применить резервную копию	Кнопка восстановления текущего узла из текущей резервной копии.
Информация о текущей копии	Информация о текущей резервной копии: • Дата создания; • Размер резервной копии; • Версия VA; • Версия Angie ADC; • Узел-источник резервной копии.

Блок "Восстановление из копии"

Angie ADC > Резервное копирование и восстановление узла

Резервное копирование и восстановление узла

Готов к восстановлению
Состояние на: 18.08.2026, 17:33

angie-adc1
10.0.2.15 · VA: angie-va-base-20260810 / ADC v1.1-SNAPSHOT-4f93c9c

Создание резервной копии
Восстановление из копии

Проверен
angie-adc1 · 1.0 МБ
Заменить файл

СРАВНЕНИЕ С ТЕКУЩИМ УЗЛОМ

Параметр	Копия	Текущий узел
Hostname	angie-adc1	angie-adc1
IP-адрес	10.0.2.15	10.0.2.15
Версия VA	angie-va-base-20260810	angie-va-base-20260810
Версия Angie ADC	v1.1-SNAPSHOT-4f93c9c	v1.1-SNAPSHOT-4f93c9c

Пара высокой доступности: Не настроена Кластер: В кластере · 192.168.105.2:3391

Копия этого узла: IP-адреса совпадают. Все сервисы будут остановлены, данные и конфигурации будут взяты из копии, узел будет перезагружен. Отменить восстановление после старта нельзя.

Начать восстановление

Элементы интерфейса:

Статус проверки архива	Отображается после загрузки архива: - Статус Проверен — архив проверен и готов к применению. - Статус Не удалось проверить архив — архив не прошел проверку и не подходит для использования, например, поврежден. Также указывается имя узла-источника и размер архива.
Заменить файл	Переход к загрузке другого архива
Сравнение с текущим узлом	Позволяет сравнить параметры текущего узла и загруженной копии: - Hostname - IP-адрес - Версия VA - Версия Angie ADC
Пара высокой доступности	Указывает, участвовал ли узел из копии в паре высокой доступности. Если да, то приводится IP-адрес соседнего узла.
Кластер	Указывает, участвовал ли узел из копии в кластере. Если да, то приводится IP-адрес точки входа.
Начать восстановление	Кнопка запуска восстановления из загруженной резервной копии.

10.8.4.8 Вкладка "NTP"

adc-psi-3 > Конфигурация NTP

NTP

☒ NTP включен

Статус: **активный**

Выбранный источник

backup.redprojects.ru

активный

Последняя проверка: меньше минуты назад

Часовой пояс: Europe/Moscow

08:33 30 апреля 2026 г.

Обновлено меньше минуты назад

Добавить сервер

NTP-сервер	Состояние	Мин. интервал опроса (степень двойки)	Макс. интервал опроса (степень двойки)	
ntp1.vniiftr.ru	активный	6	10	
pool.ntp.org	активный	6	10	

На этой вкладке отображается статус службы синхронизации NTP, а также список NTP-серверов, добавленных в Angie ADC.

Элементы интерфейса:

NTP	Переключатель, включающий использование службы синхронизации NTP на текущем устройстве
Статус	Статус службы NTP. Возможные значения: активный — служба NTP включена. выключен — служба NTP выключена. есть проблемы — возникли проблемы с синхронизацией времени. не отвечает — служба недоступна. данные устарели — не удалось получить свежие данные. применение настроек... — идет применение изменений.
Выбранный источник	NTP-сервер, используемый как источник времени в данный момент
Последняя проверка	Дата и время последнего обновления данных от NTP-сервера
Часовой пояс	Текущий часовой пояс устройства
<дата и время>	Текущие дата и время устройства
Добавить сервер	Кнопка добавления нового NTP-сервера

Таблица добавленных NTP-серверов:

НTP-сервер	Адрес НTP-сервера. По нажатию на ссылку доступно редактирование сервера.
Состояние	Состояние сервера. Возможные значения: • активный — сервер доступен. • есть проблемы — возникли проблемы с синхронизацией времени. • не отвечает — сервер недоступен. • неизвестно — статус сервера не определен. • применение настроек... — идет применение изменений.
Мин. интервал опроса (степень двойки)	Заданный минимальный интервал частоты опроса сервера. Возможные значения: от 4 до 17. По умолчанию 6: 64 секунды.
Макс. интервал опроса (степень двойки)	Заданный максимальный интервал частоты опроса сервера. Возможные значения: от 4 до 17. По умолчанию 10: 1024 секунды (~17 минут).
Значок "Удалить"	Удаление НTP-сервера

Окно "Добавление NTP-сервера"

✕

Добавление NTP-сервера

*

Адрес NTP-сервера

ntp1.vniiftri.ru или 192.168.1.*

Доменное имя или IP-адрес

*

Мин. интервал опроса (степень двойки)

4

Допустимые значения от 4 до 17

*

Макс. интервал опроса (степень двойки)

17

Допустимые значения от 4 до 17

Отмена

Добавить NTP-сервер?

Добавить

Адрес NTP-сервера	Адрес NTP-сервера (доменное имя или IP-адрес, не более 255 символов)
Мин. интервал опроса (степень двойки)	Минимальный интервал для частоты опроса сервера. Возможные значения: от 4 до 17. В качестве значения указывается степень двойки, интервал опроса NTP-сервера рассчитывается как 2 в степени n секунд. По умолчанию 6: 64 секунды.
Макс. интервал опроса (степень двойки)	Максимальный интервал для частоты опроса сервера. Возможные значения: от 4 до 17. В качестве значения указывается степень двойки, интервал опроса NTP-сервера рассчитывается как 2 в степени n секунд. По умолчанию 10: 1024 секунды (~17 минут).
Добавить	Кнопка добавления сервера
Отмена	Кнопка сброса заполненных полей формы

Окно редактирования NTP-сервера

Экран предоставляет возможность изменить существующую запись в системе.

Элементы интерфейса аналогичны тем, что представлены на экране "Добавление NTP-сервера".

10.8.4.9 Вкладка "Логирование"

Angie ADC > Логирование

Всего серверов 1	Активных 1	Добавить сервер		
Сервер	Адрес	Протокол	Уровни	
Production Syslog	192.168.1.100:514	udp	ERROR WARNING NOTICE INFO	

Таблица на этой вкладке содержит список syslog-серверов, добавленных в Angie ADC, и позволяет управлять как их составом, так и свойствами отдельных серверов.

Всего серверов	Количество настроенных syslog-серверов в системе
Активных	Количество syslog-серверов, на которые отправляются логи работы системы
Добавить сервер	Добавление нового сервера
Сервер	Имя сервера, заданное при добавлении сервера в систему. По нажатию на ссылку доступно редактирование сервера.
Адрес	Адрес сервера (доменное имя или IP-адрес)
Протокол	TCP или UDP
Уровни	Уровни критичности событий, определяющие, какие события будут отправляться на сервер. Поддерживаются следующие уровни критичности событий: • Emergency — только события, приводящие к аварийной ситуации и полному отказу системы. • Alert — критические ошибки безопасности или выход из строя важных сервисов. • Critical — серьезные проблемы, которые требуют немедленного исправления. • Error — ошибки, влияющие на работу системы, но не приводящие к ее остановке. • Warning — предупреждения о возможных проблемах. • Notice — уведомления о событиях. • Informational — информационные сообщения. • Debug — включение событий отладки.
Значок "Удалить"	Удаление syslog-сервера

Окно "Новый сервер"

Новый сервер

Имя сервера

Наименование

Адрес

IP-адрес или доменное имя

Порт

513

Протокол

TCP

Уровни критичности

EMERGENCY

ALERT

+6

Отмена

Добавить сервер?

Добавить

Имя	Имя syslog-сервера (до 255 символов)
Адрес	Адрес syslog-сервера: доменное имя или IP-адрес (до 255 символов)
Порт	Порт syslog-сервера (по умолчанию 514)
Протокол	TCP или UDP (по умолчанию UDP).
Уровни критичности	Уровни критичности событий, определяющие, какие события будут отправляться на сервер: • Emergency — только события, приводящие к аварийной ситуации и полному отказу системы. • Alert — критические ошибки безопасности или выход из строя важных сервисов. • Critical — серьезные проблемы, которые требуют немедленного исправления. • Error — ошибки, влияющие на работу системы, но не приводящие к ее остановке. • Warning — предупреждения о возможных проблемах. • Notice — уведомления о событиях. • Informational — информационные сообщения. • Debug — включение событий отладки.
Добавить	Кнопка добавления нового сервера
Отмена	Кнопка сброса заполненных полей формы

Окно редактирования сервера

Экран предоставляет возможность изменить существующую запись в системе.

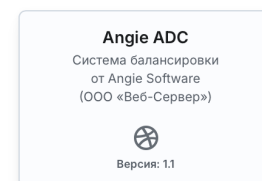
Элементы интерфейса аналогичны тем, что представлены на экране "Новый сервер", за исключением:

Включить	Переключатель, позволяющий включить или выключить отправку логов на удаленный сервер
Проверить соединение	Кнопка, позволяющая протестировать соединение с сервером
Сохранить	Кнопка сохранения данных

10.8.4.10 Вкладка "О решении"

adc-psi-3 > [Об Angie ADC](#)

Angie ADC — комплексное программное обеспечение для балансировки нагрузки и управления сетевым трафиком для создания гибкой, производительной и безопасной инфраструктуры. Поставляется как виртуальное устройство (Virtual Appliance). Регистрация в реестре ПО номер 24972 от 27.11.2024.



На этой вкладке приведено краткое описание решения Angie ADC и сведения об используемой версии.

10.8.5 Раздел "Мониторинг"

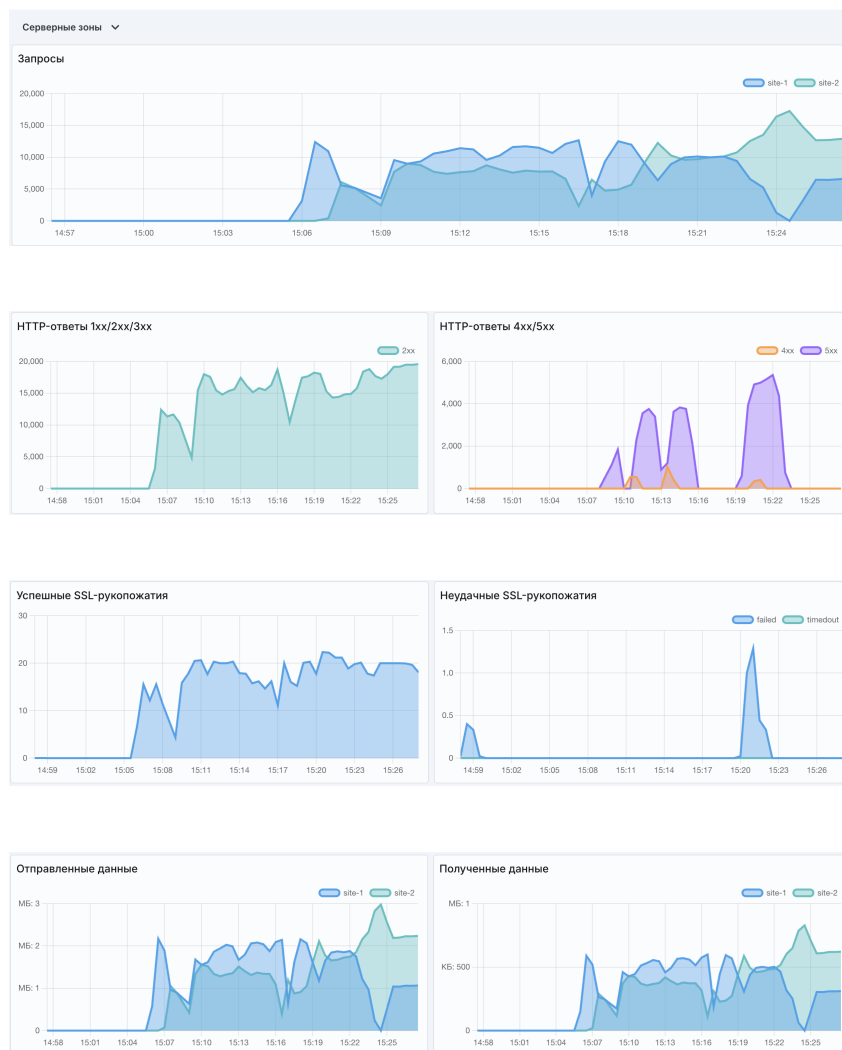
10.8.5.1 Вкладка "Балансировщик"

Экран предоставляет доступ к детализированным графикам со статистикой по балансировщику.

Элементы интерфейса:

Кнопка Конфигурация	Нажатие кнопки открывает интерактивный редактор конфигурации балансировщика, где можно просмотреть все настройки в сводном текстовом виде и при необходимости изменить их вручную.
Кнопка Перейти к интерактивному мониторингу	Нажатие кнопки открывает <i>консоль Console Light</i> для этого балансировщика.
Список За 30 минут	Раскрывающийся список для выбора временного периода при показе статистики (30 минут, 3 часа, 6 часов, 12 часов, 24 часа, 48 часов, неделя, две недели)
Серверные зоны	Раскрывающийся виджет с графиками серверных зон
Зоны апстримов	Раскрывающийся виджет с графиками зон апстримов

Виджет "Серверные зоны"



Серверные зоны

Зона	Запр.	1...	2xx	3xx	4xx	5xx	Отпр.	Пол...
site-1	0	0	108971330		13840	1620473	ГБ: 2.95	МБ: 597.44
site-2	0	0	8770116	0	0	1865952	ГБ: 2.7	МБ: 505.04

Зоны путей (Location)

Зона	Зап...	1...	2xx	3xx	4xx	5xx	Отпр.	Пол...
location_alpha	0	0	75275430		0	0	ГБ: 1.33	МБ: 361.33
location_loop	0	0	33695560		13838	34825	МБ: 551.37	МБ: 160.87
location_beta	0	0	57867690		0	0	МБ: 960.61	МБ: 273.19
location_loop2	0	0	29833470		0	0	МБ: 480.3	МБ: 143.21

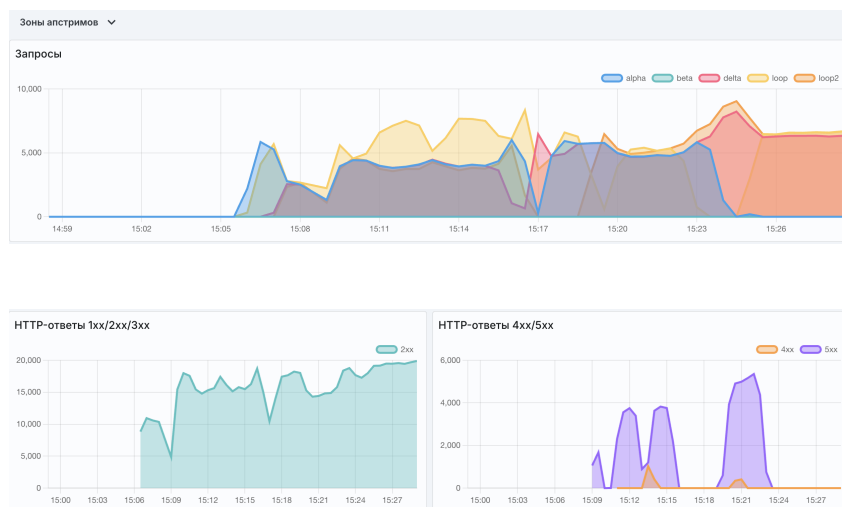
Здесь собрана статистика по серверным зонам разделяемой памяти.

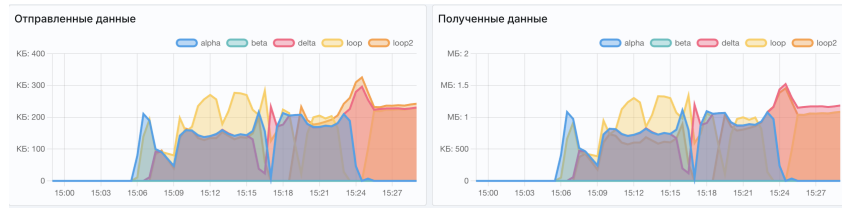
Элементы интерфейса:

Запросы	График числа запросов с разделением по отдельным серверам
HTTP-ответы 1xx/2xx/3xx	Графики числа HTTP-ответов с определенными кодами состояния с разделением по группам кодов (1xx, 2xx, 3xx)
HTTP-ответы 4xx/5xx	Графики числа HTTP-ответов с определенными кодами состояния с разделением по группам кодов (4xx, 5xx)
Отправленные данные	График объема отправленных данных с разделением по отдельным серверам
Полученные данные	График объема полученных данных с разделением по отдельным серверам
Успешные SSL-рукопожатия	График числа успешных SSL-рукопожатий с разделением по отдельным серверам
Неудачные SSL-рукопожатия	График числа неудачных SSL-рукопожатий с разделением по отдельным серверам
Серверные зоны	Таблица серверных зон со статистикой запросов, ответов и данных с разделением по отдельным зонам
Зоны путей (Location)	Таблица зон location со статистикой запросов, ответов и данных с разделением по отдельным location

Виджет "Зоны апстримов"

Здесь собрана статистика по зонам разделяемой памяти для апстримов.





Элементы интерфейса:

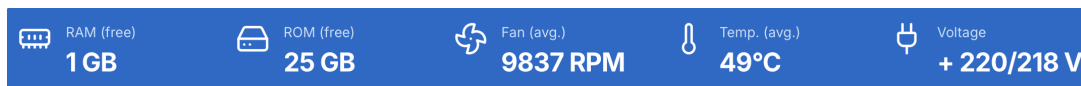
Запросы	График числа запросов с разделением по отдельным серверам
HTTP-ответы 1xx/2xx/3xx	Графики числа HTTP-ответов с определенными кодами состояния с разделением по группам кодов (1xx, 2xx, 3xx)
HTTP-ответы 4xx/5xx	Графики числа HTTP-ответов с определенными кодами состояния с разделением по группам кодов (4xx, 5xx)
Отправленные данные	График объема отправленных данных с разделением по отдельным серверам
Полученные данные	График объема полученных данных с разделением по отдельным серверам

10.8.5.2 Вкладка "Система"

Экран мониторинга, отображающий метрики системы Angie ADC и детализированные графики со статистикой за выбранный временной интервал.

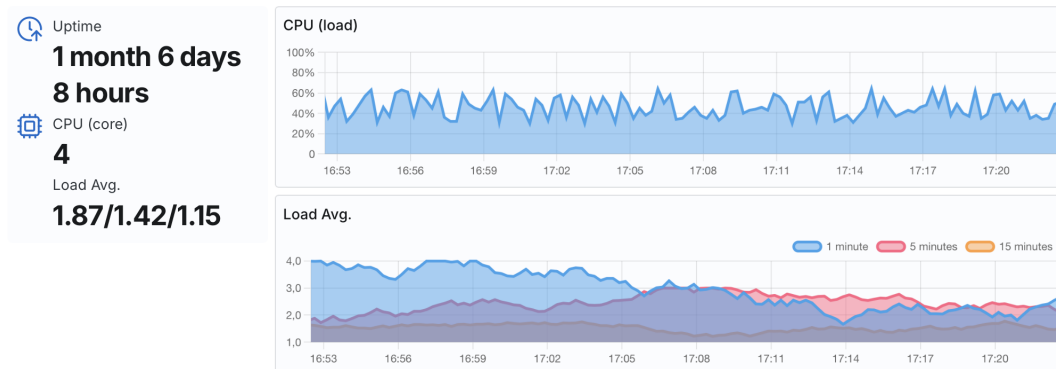
Скачать SNMP MIB-файл	Кнопка загрузки MIB-файла для интеграции с системами мониторинга
Список За 30 минут	Раскрывающийся список для выбора временного периода при показе статистики (30 минут, 3 часа, 6 часов, 12 часов, 24 часа, 48 часов, неделя, две недели)

Сводная иконографика



RAM (free)	Объем свободной оперативной памяти устройства (ГБ)
ROM (free)	Объем свободной постоянной памяти устройства (ГБ)
Fan (avg.)	Скорость вращения вентиляторов: среднее значение в оборотах в минуту (RPM) по всем вентиляторам устройства
Temp. (avg.)	Средняя температура по всем датчикам устройства (по Цельсию)
Voltage	Текущее напряжение блоков питания

Центральный процессор



Uptime	Время непрерывной работы системы с момента последнего запуска.
CPU (core)	Количество ядер процессора.
Load Avg.	Усредненная загрузка системы за 1 минуту / 5 минут / 15 минут. Интерпретируется относительно числа ядер, например, на сервере с 4 ядрами значение 2.00 означает, что система загружена на 50%.
CPU (load) (график)	График использования процессора в процентах за выбранный временной интервал.
Load Avg. (график)	График усредненной загрузки системы (1 минута / 5 минут / 15 минут) за выбранный временной интервал. Интерпретируется относительно числа ядер, например, на сервере с 4 ядрами значение 2.00 означает, что система загружена на 50%.

Обработка данных



Received	Среднее количество данных, полученных в секунду (КБ/с)
Transmitted	Среднее количество данных, переданных в секунду (КБ/с)
Collisions (avg.)	Среднее количество коллизий в сети
Input Packet Errors (avg.)	Среднее количество ошибок входящих пакетов
Output Packet Errors (avg.)	Среднее количество ошибок исходящих пакетов
Received (график)	График среднего количества данных, полученных в секунду, за выбранный временной интервал с распределением по интерфейсам
Transmitted (график)	График среднего количества данных, переданных в секунду, за выбранный временной интервал с распределением по интерфейсам

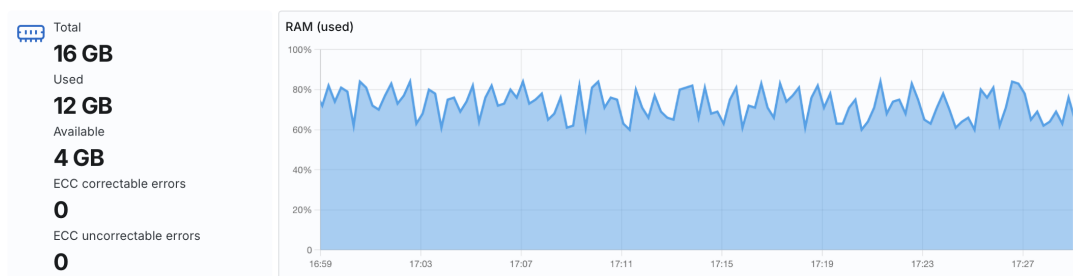
Сетевые интерфейсы

Interfaces					
Interface	Adminstate	Operstate	Collisions, %	Input Packet Errors, %	Output Packet Errors, %
eth0	✓ up	up	0	0	0

Таблица **Interfaces** отображает список подключенных интерфейсов.

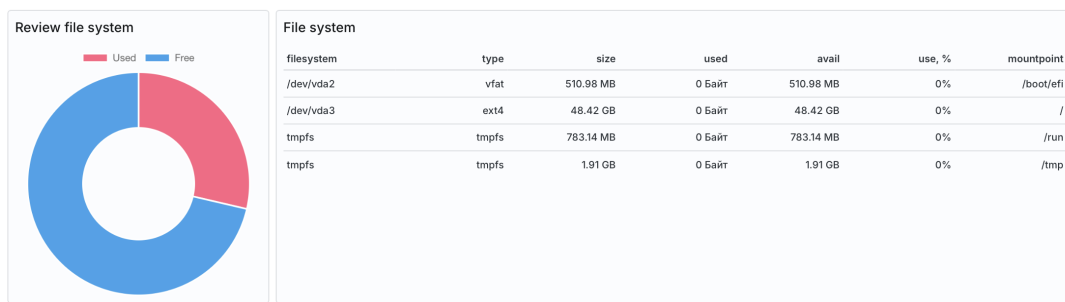
Interface	Имя сетевого интерфейса
Adminstate	Состояние сетевого интерфейса, заданное администратором (up или down)
Operstate	Фактическое состояние сетевого интерфейса на данный момент (up или down)
Collisions, %	Процент коллизий за выбранный временной интервал
Input Packet Errors, %	Процентная доля принятых пакетов, в которых содержатся ошибки, за выбранный временной интервал
Output Packet Errors, %	Процентная доля переданных пакетов, в которых содержатся ошибки, за выбранный временной интервал

ОЗУ



Total	Объем оперативной памяти, установленной на сервере (ГБ)
Available	Объем свободной оперативной памяти (ГБ)
Used	Объем используемой оперативной памяти (ГБ)
ECC correctable errors	Ошибки памяти, которые были обнаружены и автоматически исправлены (шт.)
ECC uncorrectable errors	Обнаруженные ошибки памяти, которые не удалось исправить (шт.)
RAM (used) (график)	График использования оперативной памяти в процентах за выбранный временной интервал

Файловая система



Круговая диаграмма **Review file system** отображает заполнение файловой системы в процентах (**Used** — использованное пространство, **Free** — свободное пространство).

Таблица **File system** содержит список смонтированных файловых систем и их параметры.

filesystem	Имя файловой системы
type	Тип файловой системы
size	Общий размер файловой системы
used	Объем использованного пространства
avail	Объем свободного пространства
use, %	Процент использования файловой системы
mountpoint	Точка монтирования файловой системы

Накопители (ROM)

ROM (devices)							
smartctl_status	power_on	temp	size	device	interface	model_name	serial_number
✓ OK	18 days 18 hours 40 minutes	44°C	0 B	sda	scsi	BROADCOM MR9540-8i	00b2d3e4f5a6c7d8e9f0a1b2c
✓ OK	18 days 17 hours 45 minutes	46°C	0 B	sdb	scsi	BROADCOM MR9540-8i	001a2b3c4d5e6f7a8b9c0d1e
✓ OK	18 days 6 hours 24 minutes	32°C	44713 GB	bus_0_sat+megaraid_0	sat+megaraid_0	SAMSUNG MZ7L3480HQ-00A07	S6CDNX08204513
✓ OK	17 days 23 hours 40 minutes	34°C	44713 GB	bus_0_sat+megaraid_1	sat+megaraid_1	SAMSUNG MZ7L3480HQ-00A07	S6CDNX08204527
✓ OK	18 days 15 hours 10 minutes	31°C	1.75 TB	bus_0_sat+megaraid_2	sat+megaraid_2	SAMSUNG MZ7L31T9HBLT-00A07	S5XNNA0M371248
✓ OK	18 days 10 hours	33°C	1.75 TB	bus_0_sat+megaraid_3	sat+megaraid_3	SAMSUNG MZ7L31T9HBLT-00A07	S5XNNA0M371293

Таблица **ROM (devices)** отображает состояние накопителей.

smartctl_status	Состояние накопителя по данным SMART (Self-Monitoring, Analysis and Reporting Technology)
power_on	Общее время работы с момента включения
temp	Текущая температура
size	Объем накопителя
device	Имя устройства
interface	Тип интерфейса подключения
model_name	Модель устройства
serial_number	Серийный номер

S.M.A.R.T. атрибуты диска

Таблица **Reallocated_Sector_Ct** отображает количество переназначенных (перемещенных в резервную область) секторов диска.

Reallocated_Sector_Ct				
device	raw	thresh	value	worst
sda	0	0	0	0
sdb	0	0	0	0
bus_0_sat+megaraid_0	0	0	0	0
bus_0_sat+megaraid_1	0	0	0	0
bus_0_sat+megaraid_2	0	0	0	0
bus_0_sat+megaraid_3	0	0	0	0

device	Имя устройства
raw	Необработанное значение атрибута SMART
thresh	Пороговое значение, при достижении которого состояние атрибута считается критическим
value	Нормализованное текущее значение атрибута
worst	Наихудшее нормализованное значение атрибута за выбранный временной интервал

Таблица **Current_Pending_Sector** отображает количество секторов, которые считаются нестабильными и ожидают переназначения.

Current_Pending_Sector				
device	raw	thresh	value	worst
sda	0	0	0	0
sdb	0	0	0	0
bus_0_sat+megaraid_0	0	0	0	0
bus_0_sat+megaraid_1	0	0	0	0
bus_0_sat+megaraid_2	0	0	0	0
bus_0_sat+megaraid_3	0	0	0	0

device	Имя устройства
raw	Необработанное значение атрибута SMART
thresh	Пороговое значение, при достижении которого состояние атрибута считается критическим
value	Нормализованное текущее значение атрибута
worst	Наихудшее нормализованное значение атрибута за выбранный временной интервал

Датчики

IPMI (temperature)		IPMI (fan)		IPMI (voltage)	
Sensor	Temp.	Sensor	RPM	Sensor	Voltage
TEMP_MEZZ2	44°C	Fan_PSU2	3800 RPM	P12V_MB	+12 V
TEMP_MEZZ1	44°C	Fan_PSU1	4200 RPM	PSU1_VIN	+218 V
CPU2_VCCANA	36°C	Fan5_CPU2_In	9600 RPM	PSU2_VIN	+228 V
CPU1_VCCANA	36°C	Fan7_CPU2_In	9600 RPM		
CPU1_VCCIO	35°C	Fan3_CPU1_In	9600 RPM		

IPMI (power state)		IPMI (sensor state)	
Sensor	State	Sensor	State
Server_Power_IN	1	CPU1_TMargin	0
CPU1_Power	1	CPU2_TMargin	0
CPU2_Power	1	System Event Log	0
DIMM_ABCD_Power	1	System Event	0
DIMM_EFGH_Power	1	IPMI Watchdog	0
DIMM_IJKL_Power	1	CPU2_TStatus	NaN
		CPU1_TStatus	NaN

Таблица IPMI (**temperature**) отображает список датчиков с температурами. Данные получены через интерфейс IPMI (Intelligent Platform Management Interface):

Sensor	Идентификатор датчика температуры
Temp.	Текущее значение температуры

Таблица IPMI (**fan**) отображает информацию о вентиляторах. Данные получены через интерфейс IPMI (Intelligent Platform Management Interface):

Sensor	Идентификатор датчика вентилятора
RPM	Текущая скорость вращения вентилятора в оборотах в минуту (RPM)

Таблица IPMI (**voltage**) отображает параметры напряжения. Данные получены через интерфейс IPMI (Intelligent Platform Management Interface):

Sensor	Идентификатор датчика напряжения
Voltage	Текущее значение напряжения

Таблица IPMI (**power state**) отображает состояние питания устройств. Данные получены через интерфейс IPMI (Intelligent Platform Management Interface):

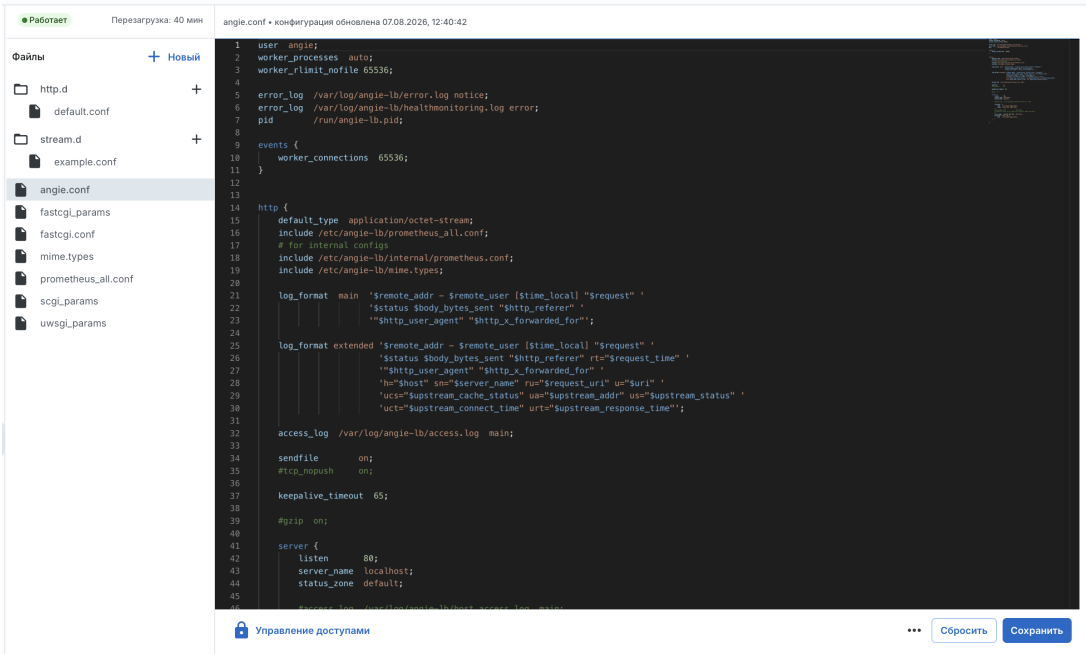
Sensor	Идентификатор датчика питания
State	Текущее состояние питания устройства (0 — выключено, 1 — включено).

Таблица IPMI (**sensor state**) отображает состояние всех остальных отслеживаемых датчиков, помимо перечисленных выше, в зависимости от типа оборудования. Данные получены через интерфейс IPMI (Intelligent Platform Management Interface):

Sensor	Идентификатор датчика
State	Текущее значение датчика (0 или 1).

10.8.6 Раздел "Управление трафиком"

10.8.6.1 Вкладка "Балансировщик нагрузки"



Элементы интерфейса:

Статус балансировщика	У балансировщика могут быть следующие статусы: • Работает — балансировщик запущен; • Не работает — балансировщик не запустился после перезагрузки конфигурации. Также отображается время, прошедшее с последней перезагрузки (Перезагрузка) и количество перезагрузок при наведении на время (Поколение).
Файлы	Дерево файлов конфигурации балансировщика
Новый	Переход к добавлению нового файла конфигурации
Редактор конфигурации	Позволяет редактировать выбранный файл конфигурации
Управление доступами	Переход к настройке прав на работу с файлами конфигурации (для администраторов)
...	Переход к контекстному меню конфигурации: • Логи: просмотр журнала ошибок балансировщика нагрузки. • История: переход к списку сохраненных файлов конфигурации балансировщика нагрузки. Позволяет выбрать произвольную версию конфигурации и применить ее. • Удалить файл: удаляет выбранный файл конфигурации балансировщика нагрузки. • Применить последнюю рабочую версию: применяет последнюю рабочую версию конфигурации балансировщика нагрузки.
Сбросить	Кнопка сброса изменений в редактируемом файле конфигурации
Сохранить	Кнопка сохранения изменений в редактируемом файле конфигурации

Окно "Управление доступами"

angle-adc1 > Балансировщик > local-lb > Конфигурация > Управление доступами

Управление доступами

Назад к конфигурации

Добавить доступ

Тип ресурса

Файл конфигурации

Пользователь

Доступ

Файл

angle.conf

логин

Просмотр

+ Добавить

Текущие доступы

Все

фильтр по ресурсу

фильтр по логину

Ресурс	Пользователь	Доступ	Кто выдал	Дата
Файл: http.d/my.conf	user	Просмотр	admin	28.07.2026

Элементы интерфейса:

Тип ресурса	Файл или Зона, к которым выдается доступ
Файл конфигурации	Раскрывающийся список файлов конфигурации. Отображается, если выбран тип ресурса Файл .
Зона	Имя зоны. Отображается, если выбран тип ресурса Зона . Зону предварительно необходимо настроить.
Пользователь	Логин пользователя, которому выдается доступ
Доступ	Тип доступа Просмотр или Редактирование . Для типа ресурса Зона всегда выдается доступ на редактирование.
Добавить	Кнопка добавления доступа
Текущие доступы	Таблица со списком выданных доступов. Для удобства поиска поддерживается фильтрация.
Назад к конфигурации	Кнопка перехода к конфигурации балансировщика

Окно "История"

angle-adc1 > Балансировщик > local-lb > Конфигурация > История

Сохраненные конфигурации

Управление резервным копированием

Дата создания	Версия Angie ADC	Статус конфигурации	Теги	Комментарий	Последнее изменение
19.08.2026 20:49:44	adc-v1.1-SNAPSHOT-4f93c9c	Рабочая	angle-lb, 33c8d699-71d1-4d65-937f-4f2a24d5fcbd		19.08.2026 20:49:44 ...
18.08.2026 16:04:00	adc-v1.1-SNAPSHOT-4f93c9c	Рабочая	angle-lb, 33c8d699-71d1-4d65-937f-4f2a24d5fcbd		18.08.2026 16:04:00 ...

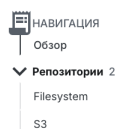
Элементы интерфейса:

Управление резервным копированием	Переход к настройкам хранения копий конфигурации (Окно "Настройки резервного копирования для балансировщика нагрузки")
Таблица со списком сохраненных конфигураций	Для каждой конфигурации указано: • время создания; • версия Angie ADC; • статус (рабочая или нерабочая конфигурация); • пользовательский комментарий; • теги (angie-lb и UUID); • дата последнего изменения.
...	Переход к контекстному меню конфигурации: • Пометить как нерабочую: помечает конфигурацию как нерабочую, такую конфигурацию нельзя применить. • Пометить как рабочую: помечает конфигурацию как рабочую, такую конфигурацию можно применить. • Применить: применяет выбранную конфигурацию.

Окно "Настройки резервного копирования для балансировщика нагрузки"

Настройки хранения копий конфигурации балансировщика нагрузки.

Настройки резервного копирования для балансировщика нагрузки



Обзор

Текущий репозиторий
filesystem

Размер копии
5.88 КБ

Последняя запись
01.07.2026 12:36:49

Управление репозиториями

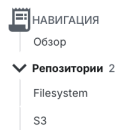
По умолчанию резервные копии конфигурации балансировщика нагрузки хранятся в локальном репозитории (filesystem). Для большей надежности рекомендуется подключить внешний репозиторий. [Настройка локального репозитория \(filesystem\)](#)
[Настройка внешнего репозитория \(S3\)](#)

Элементы интерфейса:

Текущий репозиторий	Репозиторий, в котором хранится резервная копия конфигурации балансировщика нагрузки в данный момент
Размер копии	Размер резервной копии конфигурации в репозитории
Последняя запись	Дата и время последнего изменения в репозитории
Настройка локального репозитория (filesystem)	Открывает окно подключения локального репозитория
Настройка внешнего репозитория (S3)	Открывает окно подключения внешнего репозитория

Окно "Настройка локального репозитория"

Настройка локального репозитория



Подключение локального репозитория (filesystem)



Подключение нового репозитория

Если у вас уже была создана резервная копия в локальном репозитории, то при подключении нового репозитория будет создана новая резервная копия, а старая копия будет полностью удалена.

Подключить

Элементы интерфейса:

Подключить

Кнопка подключения локального репозитория и создания в нем резервной копии конфигурации балансировщика.

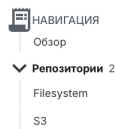


Примечание

Если у вас до этого уже была сохранена резервная копия в filesystem, она будет удалена.

Окно "Настройка внешнего репозитория"

Настройка внешнего репозитория



Подключение внешнего репозитория (S3)

Резервная копия конфигурации балансировщика нагрузки будет храниться в S3-совместимом внешнем репозитории. Если в подключаемом репозитории уже была резервная копия конфигурации, то будет использована она. Если подключается новый репозиторий, то будет создана новая резервная копия.

Если резервная копия до этого была сохранена в filesystem, то она будет удалена.

Подключение к S3-совместимому внешнему репозиторию

Заполните обязательные поля

Конечная точка (endpoint) *

Имя корзины (bucket_name)

☐ Использовать SSL

Ключ доступа (access_key) *

Секретный токен (secret_access_key) *

Сбросить

Подключить

Элементы интерфейса:

Конечная точка	Адрес сервиса, который будет использоваться для хранения резервной копии.
Имя корзины	Имя корзины (бакета), в которой будут храниться данные. По умолчанию установлено значение angie-adc .
Ключ доступа	Идентификатор для доступа к сервису.
Секретный токен	Секретный токен для доступа к сервису.
Использовать SSL	При установке флажка подключение к сервису будет осуществляться по HTTPS. По умолчанию флажок снят.
Сбросить	Кнопка сброса заполненных полей формы
Подключить	Кнопка подключения внешнего репозитория. Если вы подключаете репозиторий, в котором уже была резервная копия конфигурации, то будет использована эта копия. Если подключается новый репозиторий, будет создана резервная копия текущей конфигурации балансировщика.

Примечание

Если у вас до этого уже была сохранена резервная копия в filesystem, она будет удалена.

10.8.6.2 Вкладка "Глобальная балансировка"

adc-psi-2 > Глобальная балансировка

Настройки глобальной балансировки

☐ Глобальная балансировка

Домены: 1 Записи: 0 Проверки работоспособности: 0 Порт: 53

Домен	Динамические записи	Статические записи
example.com	0	0

Добавить домен

Выбранный домен: example.com Удалить домен

Динамические записи					
Имя	Тип	Балансировка	TTL	Бэкенды	Проверки
В этом домене пока нет динамических записей					

Добавить запись

Статические записи			
Имя	Тип	TTL	Бэкенды
В этом домене пока нет статических записей			

Добавить запись

Проверки работоспособности				
Имя	Режим	Интервал	Таймаут	Используется в
Проверок пока нет				

Добавить проверку

Сбросить Применить

На этом экране можно задать конфигурацию GSLB, просмотреть текущую версию и изменять ее. После создания конфигурации в верхней части экрана отображается статус применения конфигурации:

- **Применяется** – конфигурация находится в процессе применения.
- **Применено** – конфигурация успешно применена.
- **Повторная попытка** – не удалось применить конфигурацию, процесс запущен заново.

Элементы интерфейса:

Глобальная балансировка	Переключатель включает или выключает глобальную балансировку; по умолчанию находится в положении "Выключено".
Сводная статистика	Общее количество настроенных доменов, записей и проверок работоспособности. Также указан используемый порт (53).
Таблица доменов, для которых настроена глобальная балансировка	Для каждого домена приведено его имя и статистика по количеству динамических и статических записей. Вы можете выбрать домен и просмотреть параметры глобальной балансировки, установленные для него, а также изменить эти параметры, если необходимо.
Добавить домен	Кнопка добавления домена, для которого нужно настроить глобальную балансировку.
Добавить проверку	Кнопка добавления проверки работоспособности. При создании записей эту проверку можно будет подключить к нужным бэкенд-серверам.
Удалить домен	Кнопка удаления домена из конфигурации.
Сбросить	Кнопка сброса введенных данных.
Применить	Кнопка сохранения изменений в конфигурации глобальной балансировки.

Блок "Выбранный домен"

Выбранный домен

example.com

Удалить домен

Динамические записи

Добавить запись

Имя	Тип	Балансировка	TTL	Бэкенды	Проверки
Без имени	A	failover	30s	1	—

Параметры динамической записи

Удалить запись

Имя

name

Тип

A

Балансировка

failover

TTL

30s

Бэкенды

Добавить бекенд

#	IP	Проверка	Приоритет	
1	IP	Нет проверок	0+	Удалить

Статические записи

Добавить запись

Имя	Тип	TTL	Бэкенды
Без имени	A	300s	1

Параметры статической записи

Удалить запись

Имя

name

Тип

A

TTL

300s

Бэкенды

Добавить ответ

#	IP	
1	IP	Удалить

В этом блоке можно просмотреть параметры выбранного домена и изменить их.

Элементы интерфейса:

Выбранный домен	Имя домена. Имя домена должно быть корректным DNS-именем без завершающей точки.
Динамические записи	Список динамических записей, добавленных для домена, с параметрами. По нажатию на запись доступно ее редактирование и удаление.
Статические записи	Список статических записей, добавленных для домена, с параметрами. По нажатию на запись доступно ее редактирование и удаление.
Добавить запись	Кнопка добавления новой записи для домена.

Параметры динамической записи типа **A** и **AAAA**

Имя	Имя записи. Должно быть уникальным в рамках одного домена.
Тип	Тип записи. Возможные значения: • A — сопоставляет доменное имя с IPv4-адресом. • AAAA — сопоставляет доменное имя с IPv6-адресом.
Балансировка	Тип балансировки. Возможные значения: • round_robin — запросы распределяются по серверам последовательно. • weighted round_robin — запросы распределяются по серверам последовательно с учетом весов серверов (на серверы с большим весом запросы будут попадать чаще). • failover — запрос направляется на доступный сервер с наивысшим приоритетом.
TTL	Значение TTL (Time to Live) в секундах. Определяет, как долго DNS-запись может храниться в кэше. По умолчанию установлено значение 30.
Бэкенды	Таблица бэкенд-серверов. Необходимо задать как минимум один бэкенд. Для каждого бэкенд-сервера задаются параметры: • IP: IP-адрес сервера. Поддерживаются IPv4 и IPv6-адреса. • Проверка: проверка работоспособности, выполняемая для этого сервера. Выбирается по имени из списка заданных проверок. • Приоритет: приоритет сервера (необходим для режима failover). Чем меньше значение, тем выше приоритет сервера, которому это значение назначено. • Вес: вес сервера (необходим для режима weighted round_robin). На серверы с большим весом запросы будут попадать чаще.
Добавить бэкенд	Кнопка добавления нового бэкенд-сервера для записи.
Удалить	Кнопка удаления бэкенд-сервера из записи.

Параметры статической записи типа A и AAAA

Имя	Имя записи.
Тип	Тип записи. Возможные значения: • A — сопоставляет доменное имя с IPv4-адресом. • AAAA — сопоставляет доменное имя с IPv6-адресом.
TTL	Значение TTL (Time to Live) в секундах. Определяет, как долго DNS-запись может храниться в кэше. По умолчанию установлено значение 300.
Бэкенды	Таблица бэкенд-серверов. Для каждого бэкенд-сервера задаются его параметры: • IP: IP-адрес сервера.
Добавить бэкенд	Кнопка добавления нового бэкенд-сервера для записи.
Удалить	Кнопка удаления бэкенд-сервера из записи.

Параметры статической записи типа CNAME

Имя	Имя записи.
Тип	Тип записи CNAME — задает псевдоним для другого доменного имени. Необходимо указать ровно один хост.
TTL	Значение TTL (Time to Live) в секундах. Определяет, как долго DNS-запись может храниться в кэше. По умолчанию установлено значение 300.
Бэкенды	Таблица бэкенд-серверов. Необходимо указать ровно один бэкенд: • Хост: доменное имя, на которое будет указывать CNAME-запись.

Параметры статической записи типа MX

Имя	Имя записи.
Тип	Тип записи MX — указывает почтовый сервер для домена. Необходимо указать хотя бы один почтовый сервер.
TTL	Значение TTL (Time to Live) в секундах. Определяет, как долго DNS-запись может храниться в кэше. По умолчанию установлено значение 300.
Бэкенды	Таблица бэкенд-серверов. Для каждого бэкенд-сервера задаются его параметры: • Хост: доменное имя почтового сервера. • Приоритет: приоритет сервера. Чем меньше значение, тем выше приоритет. Необязательный параметр.
Добавить бэкенд	Кнопка добавления нового бэкенд-сервера для записи.
Удалить	Кнопка удаления бэкенд-сервера из записи.

Параметры статической записи типа TXT

Имя	Имя записи.
Тип	Тип записи TXT — содержит произвольный текст.
TTL	Значение TTL (Time to Live) в секундах. Определяет, как долго DNS-запись может храниться в кэше. По умолчанию установлено значение 300.
Бэкенды	Таблица записей бэкенд-серверов. Для каждой записи бэкенд-сервера задаются его параметры: • Текст: TXT-запись должна содержать непустой текст длиной не более 4096 символов.
Добавить бэкенд	Кнопка добавления нового бэкенд-сервера для записи.
Удалить	Кнопка удаления бэкенд-сервера из записи.

Блок "Проверки работоспособности"

В этом блоке можно просмотреть список настроенных проверок работоспособности, добавлять новые проверки и изменять их параметры.

Элементы интерфейса:

Таблица настроенных проверок работоспособности.	Для каждой проверки приведено ее имя, режим, интервал, таймаут и количество бэкендов, на которых она подключена. Вы можете выбрать проверку и просмотреть ее параметры, а также изменить их, если необходимо.
Добавить проверку	Кнопка добавления новой проверки работоспособности.

Параметры проверки работоспособности HTTP

Имя	Имя проверки работоспособности (до 255 символов).
Режим	Режим проверки HTTP — активные проверки работоспособности по протоколу HTTP.
Интервал	Интервал между проверками в секундах.
Таймаут	Таймаут ожидания в секундах. Если за это время ответ не получен, проверка считается неуспешной.
Хост	Значение заголовка Host: в HTTP-запросе (DNS-имя проверяемого сервиса), например <code>app.example.com</code> .
Порт	Порт для запроса.
Путь запроса	Путь, по которому выполняется запрос, например <code>/health</code> или <code>/status</code> .
Метод	HTTP-метод запроса проверки, например GET.
Код ответа	Код ответа сервера, используемый в качестве критерия успешного прохождения проверки, например 200.
Заголовки	Заголовок ответа сервера, используемый в качестве критерия успешного прохождения проверки.
Тело ответа	Тело ответа сервера, используемое в качестве критерия успешного прохождения проверки.

Параметры проверки работоспособности ICMP

Имя	Имя проверки работоспособности (до 255 символов).
Режим	Режим проверки ICMP — проверка доступности с помощью ICMP Echo Request (ping).
Интервал	Интервал между проверками в секундах.
Таймаут	Таймаут ожидания в секундах. Если за это время ответ не получен, проверка считается неуспешной.
Хост	DNS-имя проверяемого сервиса, например <code>app.example.com</code> .
Число пакетов	Количество ICMP-пакетов, отправляемых при выполнении проверки.

Параметры проверки работоспособности TCP

Имя	Имя проверки работоспособности (до 255 символов).
Режим	Режим проверки TCP — активные проверки работоспособности по протоколу TCP.
Интервал	Интервал между проверками в секундах.
Таймаут	Таймаут ожидания в секундах. Если за это время ответ не получен, проверка считается неуспешной.
Хост	IP-адрес проверяемого сервиса.
Порт	Порт, используемый для выполнения проверки.

Параметры для проверки работоспособности Angie

Имя	Имя проверки работоспособности (до 255 символов)
Режим	Режим проверки Angie — проверка на основе данных о состоянии бэкендов, полученных через <i>Status API Angie</i> . Подходит для случаев, когда бэкенды не имеют собственного URL для проверки работоспособности.
Интервал	Интервал между проверками в секундах.
Таймаут	Таймаут ожидания в секундах. Если за это время ответ не получен, проверка считается неуспешной.
Хост	Значение заголовка <code>Host:</code> в HTTP-запросе (DNS-имя проверяемого сервиса), например <code>app.example.com</code> .
Порт	Порт, используемый для выполнения проверки.
URI	Путь, по которому выполняется запрос.
Заголовки	Заголовок ответа сервера, используемый в качестве критерия успешного прохождения проверки.

10.8.6.3 Вкладка "Аутентификация клиентов"

angie-adc1 > Аутентификация клиентов

Аутентификация клиентов

Настройка порталов для аутентификации клиентов.

Порталы [+ Добавить](#)

Порталы ещё не настроены.
Добавьте портал, чтобы получить URL для проверки сессии и входа клиентов

Новый портал [Сбросить](#)

Имя
customer-a-ldap

Пути
URL проверки сессии : /auth/verify?provider=customer-a-ldap
URL входа : /auth/login?provider=customer-a-ldap

Провайдер
RADIUS TACACS LDAP

Параметры подключения LDAP-сервера

Сервер: FQDN или IPv4/IPv6 Порт: 389
Режим защиты соединения: Без TLS LDAPS StartTLS
☒ Проверять TLS сертификат

DN сервисной учётной записи: cn=svc-ldap,dc=example,dc=org
Пароль сервисной учётной записи: Введите пароль

Поиск пользователя
Базовый DN пользователя: dc=example,dc=org Идентификатор логина: uid
Фильтр поиска: (&(objectClass=person)(uid=%s))

[+ Добавить портал](#)

Экран позволяет настроить аутентификацию клиентов в балансировщике нагрузки.

Элементы интерфейса:

Порталы	Список настроенных порталов с указанием имени и провайдера аутентификации (LDAP, TACACS+ или RADIUS).
Добавить	Кнопка добавления нового портала аутентификации.

Окно добавления портала с провайдером аутентификации RADIUS

Новый портал [Сбросить](#)

Имя
идентификатор портала

Пути
URL для проверки сессии и входа отобразятся после заполнения имени портала.

Провайдер
RADIUS TACACS LDAP

Параметры подключения RADIUS-сервера

Сервер: FQDN или IPv4/IPv6 Порт: 1812

Протокол: UDP Метод аутентификации: PAP

NAS-идентификатор: adc-backend

Секрет: Введите новый секрет (если нужно обновить)

[+ Добавить портал](#)

Сбросить	Кнопка сброса введенных данных
Имя	Имя портала. Имя должно быть уникальным и валидным для использования в URL: нижний регистр, начинается с буквы; допускаются буквы, цифры и дефисы, не допускаются пробелы и специальные символы.
Пути	URL проверки сессии и URL входа. Отображаются после ввода имени портала. Эти пути необходимо указать в конфигурации балансировщика нагрузки.
Провайдер	Тип провайдера, LDAP, TACACS+ или RADIUS
Сервер	IP-адрес или доменное имя сервера внешней аутентификации.
Порт	Порт подключения к серверу (по умолчанию 1812).
Протокол	Протокол передачи данных (по умолчанию UDP).
Метод аутентификации	На данный момент поддерживается метод PAP.
NAS-идентификатор	Идентификатор NAS (Network Access Server), передаваемый серверу RADIUS.
Секрет	Общий секрет для шифрования паролей.
Добавить портал	Кнопка сохранения изменений.

Окно добавления портала с провайдером аутентификации TACACS+

Новый портал

Сбросить

Имя

идентификатор портала

Пути

URL для проверки сессии и входа отобразятся после заполнения имени портала.

Провайдер

RADIUS TACACS+ LDAP

Параметры подключения TACACS-сервера

Сервер

FQDN или IPv4/IPv6

Порт

49

Секрет

Введите новый секрет (если нужно обновить)

+ Добавить портал

Сбросить	Кнопка сброса введенных данных
Имя	Имя портала. Имя должно быть уникальным и валидным для использования в URL: нижний регистр, начинается с буквы; допускаются буквы, цифры и дефисы, не допускаются пробелы и специальные символы.
Пути	URL проверки сессии и URL входа. Отображаются после ввода имени портала. Эти пути необходимо указать в конфигурации балансировщика нагрузки.
Провайдер	Тип провайдера, LDAP, TACACS+ или RADIUS
Сервер	IP-адрес или доменное имя сервера внешней аутентификации.
Порт	Порт подключения к серверу (по умолчанию 49).
Секрет	Общий секрет для шифрования паролей.
Добавить портал	Кнопка сохранения изменений.

Окно добавления портала с провайдером аутентификации LDAP

Новый портал

Сбросить

Имя

идентификатор портала

Пути

URL для проверки сессии и входа отобразятся после заполнения имени портала.

Провайдер

RADIUS

TACACS

LDAP

Параметры подключения LDAP-сервера

Сервер

FQDN или IPv4/IPv6

Порт

389

DN сервисной учетной записи

cn=svc-ldap,dc=example,dc=org

Режим защиты соединения

Без TLS

LDAPS

StartTLS

Введите пароль

Проверять TLS сертификат

Поиск пользователя

Базовый DN пользователя

dc=example,dc=org

Идентификатор логина

uid

Фильтр поиска

(&(objectClass=person)(uid=%s))

+

Добавить портал

Сбросить	Кнопка сброса введенных данных
Имя	Имя портала. Имя должно быть уникальным и валидным для использования в URL: нижний регистр, начинается с буквы; допускаются буквы, цифры и дефисы, не допускаются пробелы и специальные символы.
Пути	URL проверки сессии и URL входа. Отображаются после ввода имени портала. Эти пути необходимо указать в конфигурации балансировщика нагрузки.
Провайдер	Тип провайдера, LDAP, TACACS+ или RADIUS
Сервер	IP-адрес или доменное имя сервера внешней аутентификации.
Порт	Порт подключения к серверу (по умолчанию 389).
Режим защиты соединения	Указывает, будет ли защищено соединение. Доступные варианты: - Без TLS (по умолчанию); - LDAPS (использование TLS при подключении); - StartTLS (использование механизма StartTLS при подключении).
Проверять TLS-сертификат	Флажок доступен при выборе режима защиты соединения LDAPS или StartTLS. По умолчанию флажок установлен.
DN сервисной учетной записи	Логин сервисной учетной записи, от имени которой система подключается к LDAP.
Пароль сервисной учетной записи	Пароль сервисной учетной записи для подключения к LDAP-серверу.
Базовый DN пользователя	Задаёт ветку на LDAP-сервере, в которой система будет искать пользователя, например ou=users,dc=example,dc=com.
Идентификатор логина	Поле в LDAP, которое содержит логин, например uid, sAMAccountName или userPrincipalName.
Фильтр поиска	LDAP-фильтр для поиска пользователя, например (&(objectClass=user)(sAMAccountName=%s)).
Добавить портал	Кнопка сохранения изменений.

10.8. Веб-интерфейс Angie ADC

844

10.8.6.4 Вкладка "Конфигурация RNI"

adc-psi-2 > Конфигурация RNI

Конфигурация RNI
Заполните все обязательные поля.

Интервал опроса (мс)

☐ Объявлять префиксы при восстановлении апстримов

Префиксы + Добавить

VIP-адрес (CIDR) *	Приоритет *	Имя префикс-листа *	Имя апстрима *	
	1			

Сохранить

Экран позволяет настроить конфигурацию RNI.

Элементы интерфейса:

Интервал опроса	Информационное поле, определяющее частоту опроса апстримов. Указывается в мс. Установленное значение — 1000 мс. Поле недоступно для изменения.
Объявлять префиксы при восстановлении апстримов	При установке флажка отозванные префиксы будут автоматически анонсироваться в маршрутизируемую сеть при восстановлении работоспособности апстримов. По умолчанию флажок снят.
Добавить	Кнопка добавления нового префикса в конфигурацию.
VIP-адрес (CIDR)	Виртуальный IP-адрес для апстрима. Указывается в формате X.X.X.X/маска. Значение должно быть уникальным в рамках одного префикс-листа.
Приоритет	Номер последовательности, определяющий порядок обработки правил. Маршрут с меньшим приоритетом обрабатывается первым. Значение приоритета должно быть уникальным в рамках одного префикс-листа.
Имя префикс-листа	Имя префикс-листа для маршрутизации.
Имя апстрима	Апстрим, для которого настраивается отзыв префикса.
Сохранить	Кнопка сохранения изменений в конфигурации.
Значок "Удалить"	Удаление префикса из таблицы конфигурации RNI.

10.8.7 Раздел "Высокая доступность и кластеризация"

10.8.7.1 Вкладка "Высокая доступность"

angie-adc2 > Высокая доступность

Пара высокой доступности

Состояние пары on
Последнее изменение 15.04.2026, 12:36:38

Узел	IP-адрес узла	Роль узла
Текущий	192.168.105.18	Резервный
Соседний	192.168.105.2	Основной

Синхронизировать
Расформировать

Проверка доступности узлов
Укажите интервал отправки запросов к основному узлу и время ожидания ответа.

Интервал отправки запросов (мс) *

250

Таймаут ожидания ответа (с) *

3

Сбросить
Сохранить

На вкладке "Высокая доступность" можно просмотреть свойства пары высокой доступности, изменить параметры проверки доступности узлов, запустить синхронизацию пары и удалить ее.

Блок "Пара высокой доступности"

Элементы интерфейса:

Состояние пары	Состояние пары высокой доступности. Возможны следующие варианты: • starting – пара запускается. Статус отображается сразу после создания пары в процессе ее конфигурации и синхронизации. • on – статус отображается при нормальной работе пары. Высокая доступность обеспечивается. • failed – статус отображается после запуска пары при возникновении ошибок. Синхронизация и обмен данными между узлами в паре невозможны. Высокая доступность не обеспечивается. • not_reachable – статус отображается, когда узел не получает ответ от соседнего узла по всем интерфейсам, участвующим в паре, в течение заданного таймаута (например из-за сетевой проблемы или недоступности соседнего узла). Если резервный узел фиксирует недоступность основного узла, выполняется переключение ролей: резервный узел становится основным. При этом для соседнего узла будет отображаться его последняя известная роль. Примечание Рекомендуется проверить соседний узел и, если требуется, восстановить его связность и работоспособность.
Последнее изменение	Дата и время последнего изменения в паре, например смены ролей узлов.
Узел	Идентификация узла пары, на котором открыт веб-интерфейс Angie ADC: • Текущий – узел, на котором в данный момент открыт веб-интерфейс Angie ADC со свойствами пары; • Соседний – второй узел пары.
IP-адрес узла	IP-адрес узла, используемый для взаимодействия в паре. Предупреждение Не рекомендуется изменять настройки интерфейсов узлов, объединенных в пару высокой доступности, иначе пара может потерять связность и работоспособность.
Роль узла	Роль узла пары: • Основной – находится в режиме Active (обрабатывает трафик). • Резервный – находится в режиме ожидания (Standby).
Синхронизировать	Кнопка запуска синхронизации конфигураций узлов в паре.
Расформировать	Кнопка удаления пары.

Блок "Проверка доступности узлов"

Элементы интерфейса:

Интервал отправки запросов (мс)	Интервал отправки запросов резервным узлом для проверки доступности основного узла. По умолчанию запросы отправляются каждые 250 мс. Допустимый диапазон значений – от 200 до 1000 мс.
Таймаут ожидания ответа (с)	Таймаут ожидания ответа от интерфейса основного узла. Если с момента получения последнего успешного ответа прошло больше заданного времени интерфейс будет считаться недоступным. При недоступности всех интерфейсов основного узла произойдет переключение ролей. По умолчанию – 3 секунды. Допустимый диапазон значений – от 1 до 3 секунд.
Сбросить	Кнопка сброса заполненных полей формы.
Сохранить	Кнопка сохранения и применения изменений.

Мастер создания пары высокой доступности: подключение второго узла

Подключите второй узел пары
Заполните все обязательные поля.

IP-адрес *

Логин *

Пароль *

Сбросить

Подключить

Элементы интерфейса:

IP-адрес	IP-адрес подключаемого узла пары (поддерживается только IPv4), выделенный для удаленного администрирования и мониторинга этого узла. Этот адрес не распространяется и не синхронизируется. Подключаемому узлу будет автоматически назначена роль резервного.
Логин	Логин для авторизации на подключаемом узле
Пароль	Пароль локальной учетной записи <code>admin</code> для авторизации на подключаемом узле
Сбросить	Кнопка сброса заполненных полей формы
Подключить	Создает соединение между двумя узлами и открывает окно настройки соответствия интерфейсов узлов в паре.

Примечание

Необходимо использовать логин локальной учетной записи `admin`.

Мастер создания пары высокой доступности: настройка соответствия интерфейсов

Настройка соответствия интерфейсов

Основной узел

Резервный узел

Интерфейсы основного узла

↔

Интерфейсы резервного узла

Назад

Сохранить

Интерфейсу одного узла может соответствовать только один интерфейс другого узла. Минимально достаточно одной пары интерфейсов, но мы рекомендуем задать пары для всех интерфейсов обоих узлов.

В списках отображаются интерфейсы в состоянии **up**. В списках не отображаются и не доступны для выбора следующие интерфейсы: **loopback**, в состоянии **down**, без IP-адреса.

Элементы интерфейса:

Раскрывающийся список "Интерфейсы основного узла"	Позволяет выбрать интерфейс основного узла пары для создания пары интерфейсов
Раскрывающийся список "Интерфейсы резервного узла"	Позволяет выбрать интерфейс резервного узла пары для создания пары интерфейсов
Добавить	Добавляет строку для создания новой пары интерфейсов
Сохранить	Сохраняет добавленные пары интерфейсов и завершает мастер
Назад	Возвращает на предыдущий шаг к подключению второго узла пары
Значок удаления	Удаляет добавленную пару интерфейсов

10.8.7.2 Вкладка "Кластер"

На вкладке "Кластер" можно создать кластер Angie ADC, добавлять узлы и просматривать состояние кластера.

adc-psi-2 > [Кластер](#)

Настройки кластера

Подключен
Состояние на: 01.07.2026, 12:42
Текущий узел: cluster

Статус кластера

● Активный

1 пир

1 активный

0 проблемных

Пир	Статус	Upstream	Downstream	Ошибки
clus	● активный	активный Последняя синхронизация: 663 мс назад Задержка: 638 мс	активный Последняя синхронизация: 686 мс назад Задержка: 7.1 мс	нет

Блок "Текущий узел"

Элементы интерфейса:

Текущий узел	Кластерное имя узла, на котором сейчас открыт веб-интерфейс Angie ADC. Имя узла в кластере может отличаться от <code>hostname</code> узла и отображает логическую структуру кластера.
Статус текущего узла	● Ожидает подключения – узел не входит в кластер. ● Подключен – узел подключен к кластеру. ● Подключение... – выполняется подключение узла к кластеру.
Состояние на	Дата и время получения последних данных о статусе узла.
Значок "Обновить"	Кнопка обновления данных узла.

Блок "Статус кластера"

Элементы интерфейса:

Статус кластера	● Активный – кластер запущен. ● Подключение... – выполняется запуск кластера.
Сводная статистика по пирам	● общее количество пиров текущего узла в кластере; ● количество активных пиров; ● количество проблемных пиров.

Таблица с детальной информацией по каждому пиру

Элементы интерфейса:

Пир	Имя узла, входящего в кластер.
Статус	● активный – пир находится в рабочем состоянии, данные синхронизированы. ● сбой – есть проблемы с синхронизацией данных. ● недоступен – пир недоступен.
Upstream	Статус получения данных пиром: ● активный – upstream доступен, данные синхронизированы. ● подключение – выполняется подключение к кластеру. ● синхронизация – выполняется синхронизация (получение данных пиром). ● недоступен – upstream недоступен, получение данных невозможно.
Downstream	Статус передачи данных пиром: ● активный – downstream доступен, данные могут передаваться. ● подключение – выполняется подключение к кластеру. ● недоступен – downstream недоступен, передача данных невозможна.
Последняя синхронизация	Время, прошедшее с последней синхронизации данных.
Задержка	Задержка синхронизации данных между текущим узлом и пиром.
Ошибки	Здесь отображаются ошибки, возникающие при синхронизации данных.

Создание кластера

angle-adc1 > Кластер

Настройки кластера

Ожидает подключения Состояние на: 30.06.2026, 11:48 

Имя узла в кластере еще не задано

Создать кластер

Текущий узел станет точкой входа в этот кластер.

Подключить к кластеру

Текущий узел будет подключен к уже существующему кластеру.

Имя узла *

Секрет кластера *

IP текущего узла *

IP адрес

Создать

Элементы интерфейса:

Создать кластер	При выборе этого варианта будет создан кластер. Текущий узел будет назначен точкой входа в кластер.  Предупреждение После создания точки входа ее нельзя изменить, кластер нельзя расформировать. Внимательно проверьте все параметры перед созданием кластера.
Имя узла	Имя узла в кластере (произвольное); это имя может отличаться от <code>hostname</code> узла и отображает логическую структуру кластера. Допускаются строчные латинские буквы, цифры и одиночные дефисы внутри имени.
Секрет кластера	Секрет для подключения к кластеру.
IP текущего узла	IP-адрес подключаемого узла (только IPv4).
Создать	Создает кластер с текущим узлом в качестве точки входа.

Добавление нового узла в кластер

angle-adc1 > Кластер

Настройки кластера

Ожидает подключения Состояние на: 29.06.2026, 13:30

Узел не определён

Создать кластер
Текущий узел станет точкой входа в этот кластер.

Подключить к кластеру
Текущий узел будет подключен к уже существующему кластеру.

IP узла точки входа *

IP адрес

Имя текущего узла * Секрет кластера *

IP текущего узла *

IP адрес

Подключить

Элементы интерфейса:

Подключить к кластеру	При выборе этого варианта текущий узел будет подключен к уже существующему кластеру.
IP узла точки входа	IP-адрес узла, который был сделан точкой входа в кластер (только IPv4).
Имя текущего узла	Имя узла в кластере (произвольное); это имя может отличаться от <code>hostname</code> узла и отображает логическую структуру кластера. Допускаются строчные латинские буквы, цифры и одиночные дефисы внутри имени.
IP текущего узла	IP-адрес подключаемого узла (только IPv4).
Секрет кластера	Секрет для подключения к кластеру.
Подключить	Подключает текущий узел к кластеру.

10.8.8 Раздел "Диагностика"

10.8.8.1 Вкладка "Сеть"

angle-adc1 > Сеть

Диагностика сети

Ctrl+P Предыдущая команда Enter Выполнить команду

tracert 1.1.1.1

Отменить

Сохранить Очистить

```
$ tracert 1.1.1.1
tracert to 1.1.1.1 (1.1.1.1), 30 hops max, 60 byte packets
```

847ms Выполняется

```
$ ping -c 4 192.168.105.2
PING 192.168.105.2 (192.168.105.2) 56(84) bytes of data:
64 bytes from 192.168.105.2: icmp_seq=1 ttl=255 time=2.87 ms
64 bytes from 192.168.105.2: icmp_seq=2 ttl=255 time=3.63 ms
64 bytes from 192.168.105.2: icmp_seq=3 ttl=255 time=3.39 ms
```

56.4s Отменено

Экран предоставляет возможность проводить базовую диагностику сети с помощью команд `ping`, `curl`, `tracert`.

Элементы интерфейса:

Поле для ввода команды	Доступные команды: <code>ping</code>, <code>curl</code>, <code>tracert</code>. Команды <code>ping</code> и <code>tracert</code> поддерживают все стандартные аргументы (см. документацию <code>ping</code>, <code>tracert</code> на man7.org). Команда <code>curl</code> имеет ограничения: • Разрешены все поддерживаемые протоколы, кроме <code>file</code> (см. документацию <code>curl</code> на man7.org). • Разрешены только следующие аргументы: – <code>--v</code> (<code>--verbose</code>) – <code>--s</code> (<code>--silent</code>) – <code>--S</code> (<code>--show-error</code>) – <code>--i</code> (<code>--include</code>) – <code>--I</code> (<code>--head</code>) – <code>--L</code> (<code>--location</code>) – <code>--k</code> (<code>--insecure</code>) – <code>--m</code> (<code>--max-time</code>) – <code>--x</code> (<code>--proxy</code>) – <code>--u</code> (<code>--user</code>) – <code>--H</code> (<code>--header</code>) – <code>--A</code> (<code>--user-agent</code>) – <code>--e</code> (<code>--referer</code>) – <code>--d</code> (<code>--data</code>) – <code>--b</code> (<code>--cookie</code>) – <code>--G</code> (<code>--get</code>) – <code>--X</code> (<code>--request</code>) – <code>--w</code> (<code>--write-out</code>)
Кнопка Выполнить	Нажатие этой кнопки запускает выполнение введенной команды.
Кнопка Отменить	Нажатие этой кнопки отменяет запущенную команду.
Кнопка Сохранить	Нажатие этой кнопки позволяет выгрузить отчет о выполнении команды в HTML-формате.
Кнопка Очистить	Нажатие этой кнопки очищает историю выполненных команд.
Окно выполнения команды	Отображаются следующие параметры: • введенная команда; • время, затраченное на выполнение команды; • успешность или неуспешность выполнения команды (OK / Exit); • вывод команды; • значок повторения команды.

10.9 Парольная политика

Angie ADC поддерживает парольную политику для локальных пользователей Angie ADC. Парольная политика начинает действовать сразу при запуске Angie ADC со значениями по умолчанию. Требования к паролям пользователей можно донастроить в соответствии с политикой безопасности в вашей организации в разделе Система → Парольная политика.

adc-psi-3 > Парольная политика

Парольная политика

Парольная политика

Минимальная длина пароля (символов)

8

Максимальная длина пароля (символов)

64

Срок действия пароля (0 дней - бессрочный пароль)

0

Предупреждать об истечении срока действия пароля (дней)

0

Количество паролей, сохраняемых в истории

5

Допустимое количество неверных попыток входа за указанный интервал

2

Интервал подсчета неверных попыток входа (минут)

60

Длительность блокировки входа (минут)

1

Обязательно использовать следующие символы

Например: !@#\$%

До 63 уникальных печатных ASCII-символов без пробелов

☐ Требовать заглавную букву
 ☐ Требовать цифру
 ☐ Требовать смену при первом входе
 ☐ Требовать строчную букву

☐ Требовать двухфакторную аутентификацию
 ☐ Проверять историю паролей
 ☐ Разблокировка только администратором

Сохранить политику

Сброс истории паролей

☒ Один пользователь
 ☐ Все пользователи

Пользователь

Выберите пользователя

Сбросить историю

По умолчанию действуют следующие требования: пароль не может совпадать с логином без учета регистра, заданы минимальная длина пароля (8 символов) и максимальная длина пароля (64 символа), а также допустимое количество попыток входа и параметры блокировки пользователя:

- интервал, за который считаются неверные попытки входа пользователя (60 минут);
- допустимое количество попыток входа пользователя (2 попытки);
- длительность блокировки пользователя (1 минута).

Примечание

При таких параметрах пользователь, который в течение 60 минут три раза ввел неправильный пароль, будет заблокирован на одну минуту. При этом после разблокировки пользователя счетчик неправильных попыток **не сбрасывается**, пока не закончится указанный интервал (60 минут). При следующем вводе неправильного пароля в течение этих 60 минут пользователь будет заблокирован сразу.

Важно

Парольная политика распространяется только на *локальных пользователей Angie ADC* текущего тенанта на всех узлах *кластера*.

10.9.1 Настройка парольной политики

1. Перейдите в раздел Система → Парольная политика.
2. Настройте **требования к составу** пароля:
 - Укажите значения в полях **Минимальная длина пароля (символов)** (по умолчанию 8 символов) и **Максимальная длина пароля** (по умолчанию 64 символа).

10.9. Парольная политика

854

- Если необходимо задать более высокий уровень безопасности пароля, установите флажки **Требовать заглавную букву (A-Z)**, **Требовать строчную букву (a-z)** и **Требовать цифру (0-9)**.
 - Если необходимо, укажите дополнительные требования в поле **Обязательно использовать следующие символы** (можно указать до 63 уникальных печатных ASCII-символов без пробелов).
3. Установите флажок **Требовать смену при первом входе**, чтобы система запрашивала смену предустановленного пароля при первом входе пользователя в веб-интерфейс.
4. Настройте **срок действия пароля** и напоминание о необходимости сменить пароль:
- Укажите значение в поле **Срок действия пароля**.
Значение 0 — срок действия пароля не ограничен (бессрочно).
При указании срока действия пароля система будет напоминать о скором окончании срока за 7 дней до его истечения.
 - Укажите значение в поле **Предупреждать об истечении срока действия пароля**.
Значение 0 — предупреждение отключено; любое другое значение — количество дней, прошедшее с последней смены пароля, после которого система будет напоминать о необходимости сменить пароль.
5. Настройте **требования к истории паролей и защиту от повторения**:
- Установите флажок **Проверять историю паролей**, чтобы включить хранение истории паролей.
 - Укажите значение в поле **Количество паролей, сохраняемых в истории** (по умолчанию 5).
Это значение будет определять, сколько паролей подряд не может повторяться. Значение 0 — количество хранимых паролей не ограничено.
6. Настройте допустимое **количество попыток входа и параметры блокировки** пользователя:
- В поле **Интервал подсчета неверных попыток входа** укажите, за какое время система будет подсчитывать попытки входа пользователя (по умолчанию 60 минут).
 - В поле **Допустимое количество неверных попыток входа за указанный интервал** укажите разрешенное количество попыток входа за это время (по умолчанию 2).

 Примечание

При сбросе пароля администратором счетчик неправильных попыток ввода пароля тоже сбрасывается.

- В поле **Длительность блокировки входа** укажите, на какое время блокировать пользователя, превысившего количество разрешенных попыток входа (по умолчанию 1 минута).
 - Установите флажок **Разблокировка только администратором**, если хотите, чтобы заблокированных пользователей можно было разблокировать только вручную. В таком случае поле **Длительность блокировки входа** не будет учитываться.
7. Если необходимо, установите флажок **Требовать двухфакторную аутентификацию**.

При входе в веб-интерфейс пользователям будут предложены QR-код и секрет для подключения приложения-аутентификатора и дальнейшей аутентификации в системе через одноразовый пароль из этого приложения. Например, можно использовать приложения-аутентификаторы Яндекс.Ключ, Google Authenticator или Multifactor.

Важно

После пяти неудачных попыток ввода одноразового кода в течение пяти минут пользователь будет заблокирован на пять минут.

8. Нажмите **Сохранить** политику.

Политика будет сохранена и применена сразу.

10.9.2 Сброс истории паролей для пользователей

Администратор может вручную сбросить историю паролей пользователя.

1. Перейдите в раздел Система → Парольная политика.
2. В блоке **Сброс истории паролей** укажите, для кого вы хотите сбросить историю:
 - **Один пользователь** (необходимо выбрать пользователя из списка).
 - **Все пользователи** (будет сброшена история паролей для всех локальных пользователей Angie ADC).
3. Нажмите **Сбросить историю**.

10.9.3 Сброс пароля локального пользователя

Администратор может *вручную сбросить пароль* для локального пользователя, например, если пользователь забыл пароль. После этого пользователь должен заново войти в систему с новым паролем, назначенным администратором.

10.9.4 Сброс настроек двухфакторной аутентификации локального пользователя

Администратор может *вручную сбросить настройки двухфакторной аутентификации* для локального пользователя, например, если пользователь больше не имеет доступа к подключенному приложению для аутентификации. После этого пользователь должен заново войти в систему и настроить двухфакторную аутентификацию, если этого требует *парольная политика*.

10.10 Настройка учетной записи

Angie ADC позволяет настраивать безопасность учетной записи в Angie ADC. Настройки текущей учетной записи находятся в меню в верхнем правом углу веб-интерфейса (**Безопасность учетной записи**).

angie-adc1 > Безопасность учётной записи

Безопасность учётной записи

Изменить пароль

После смены пароля ранее выданные сессии будут отозваны.

Требования к новому паролю

От 8 до 64 символов

Пароль не должен совпадать с логином

Текущий пароль *

Новый пароль *

Повторите новый пароль *

Изменить пароль

Управление доступно только для локальных учётных записей. Для подтверждения операций используются текущий пароль и код из приложения аутентификации.

Включить TOTP

Поддерживаются приложения, совместимые со стандартом TOTP, например Яндекс.Ключ и Google Authenticator.

Текущий пароль *

Начать настройку

Отключить TOTP

Если политика требует двухфакторную аутентификацию, сервер не позволит её отключить.

Текущий пароль *

Одноразовый код *

Отключить TOTP

Важно

Настройка учетной записи доступна только локальным пользователям Angie ADC.

Поддерживается:

- Смена пароля учетной записи.
- Включение и выключение двухфакторной аутентификации для учетной записи.

Примечание

Выключение двухфакторной аутентификации возможно, если она не требуется *по парольной политике*.

10.10.1 Смена пароля учетной записи

1. В верхнем правом углу веб-интерфейса нажмите на значок с символом логина текущей учетной записи и выберите **Безопасность учетной записи**.

Откроется окно настройки безопасности для текущей учетной записи.

2. Введите текущий пароль, новый пароль, повторите новый пароль и нажмите **Изменить пароль**.

Новый пароль не должен повторять текущий пароль, не должен повторять логин без учета регистра и должен соответствовать *требованиям парольной политики*. В случае прохождения всех проверок пароль будет изменен и текущая пользовательская сессия будет завершена.

3. Войдите в систему с новым паролем.

Совет

Администратор может *вручную сбросить пароль* для локального пользователя, например, если пользователь забыл пароль. После этого пользователь должен заново войти в систему с новым паролем, назначенным администратором.

10.10.2 Включение двухфакторной аутентификации

Для включения двухфакторной аутентификации можно использовать, например, приложения Яндекс.Ключ, Google Authenticator или Multifactor.

Важно

После пяти неудачных попыток ввода одноразового кода в течение пяти минут пользователь будет заблокирован на пять минут. Коды восстановления не поддерживаются.

1. В верхнем правом углу веб-интерфейса нажмите на значок с символом логина текущей учетной записи и выберите **Безопасность учетной записи**.

Откроется окно настройки безопасности для текущей учетной записи.

2. В блоке **Включить TOTP** введите пароль от текущей учетной записи и нажмите **Начать настройку**.

Отобразятся QR-код и секрет для подключения приложения-аутентификатора.

3. Отсканируйте QR-код в приложении-аутентификаторе на своем мобильном телефоне. Если вы работаете на компьютере, то вместо QR-кода скопируйте и введите в приложении секрет.

4. Вернитесь в Angie ADC и введите одноразовый код аутентификации из приложения-аутентификатора.

5. Нажмите **Подтвердить и включить**.

Двухфакторная аутентификация будет включена для текущей учетной записи. Текущая пользовательская сессия будет завершена.

6. Войдите в систему заново, введя свой пароль и одноразовый код.

Совет

Администратор может *вручную сбросить настройки двухфакторной аутентификации* для локального пользователя, например, если пользователь больше не имеет доступа к подключенному приложению для аутентификации. После этого пользователь должен заново войти в систему и настроить двухфакторную аутентификацию, если этого требует *парольная политика*.

10.10.3 Отключение двухфакторной аутентификации

1. В верхнем правом углу веб-интерфейса нажмите на значок с символом логина текущей учетной записи и выберите **Безопасность учетной записи**.

Откроется окно настройки безопасности для текущей учетной записи.

2. В блоке **Отключить TOTP** введите пароль от текущей учетной записи, одноразовый код из подключенного приложения-аутентификатора и нажмите **Отключить TOTP**.

Двухфакторная аутентификация для текущей учетной записи будет отключена. Текущая пользовательская сессия будет завершена.

3. Войдите в систему заново, введя свой пароль.

ГЛАВА 11

Типовые задачи и примеры

В этом разделе собраны примеры настройки Angie ADC под разные задачи:

- *Настройка IPv6*
- *Настройка пула SNAT*
- *Настройка Transparent Proxy для TCP- и UDP-трафика*
- *Балансировка трафика на основе набора шифров*

11.1 Настройка IPv6

IPv6 в Angie ADC настраивается на трех уровнях:

- Настройка доступа к веб-интерфейсу Angie ADC (management plane).
- Настройка сетевых протоколов маршрутизации (control plane).
- Настройки для передачи и обработки клиентского трафика (data plane).

11.1.1 Доступ к веб-интерфейсу Angie ADC

Веб-интерфейс Angie ADC одновременно поддерживает IPv4 и с IPv6. Для доступа к веб-интерфейсу с IPv6-адресом 2001:db8::10 введите в браузере:

- `http://[2001:db8::10]:8080`
- `https://[2001:db8::10]:8443` (если требуется безопасное подключение).

Если есть DNS-запись для адреса 2001:db8::10, можно обращаться к веб-интерфейсу по имени.

11.1.2 Настройка сетевых протоколов маршрутизации

Для маршрутизации IPv6 подходят все стандартные и вспомогательные протоколы маршрутизации:

- BGPv4;
- OSPFv3 (для IPv6 только OSPFv3, в отличие от IPv4);
- BFD;
- VRRPv3.

Примечание

Сосед для протокола BGP и пир для BFD указываются в формате IPv6-адреса.

Пример настройки OSPFv3:

```
router ospf6
  ospf6 router-id 2.2.2.2
  log-adjacency-changes
  redistribute connected
exit
```

Также необходимо включить OSPFv3 на всех интерфейсах:

```
interface ens33
  description internal
  ipv6 address 2001:db8::2/64
  ipv6 ospf6 area 0
exit
```

11.1.3 Передача и обработка клиентского трафика

Для поддержки IPv6:

- Бэкенд-серверы должны быть настроены на прием подключений по протоколу IPv6.
- В конфигурации балансировщика нагрузки необходимо включить прием трафика по IPv6:

```
server {
  listen 80;
  listen [::]:80;
  location / {
```

После этого балансировщик сможет принимать подключения по IPv6.

- Также необходимо добавить AAAA-записи в DNS, чтобы клиенты могли находить балансировщик по IPv6-адресу.
- Для маршрутизации трафика к бэкенд-серверам в секции **upstream** должны быть указаны IPv6-адреса.

В примере ниже один сервер задан адресом IPv4, а второй – IPv6:

```
http {
  upstream myapp1 {
    server 192.168.0.128:80;
    server [2001:db8::4]:80;
  }
}
```

11.1.4 Поддержка смешанных подключений IPv4 и IPv6 в Angie ADC

Angie ADC поддерживает гибкую маршрутизацию трафика между клиентами и бэкенд-серверами с разными конфигурациями.

Бэкенд-серверы могут быть трех типов:

- только IPv4;
- только IPv6;
- dual-stack (одновременно поддерживаются оба протокола).

Пример:

К некоторому ресурсу подключаются IPv4-клиенты (отмечены зеленым цветом) и IPv6-клиенты (отмечены красным цветом).

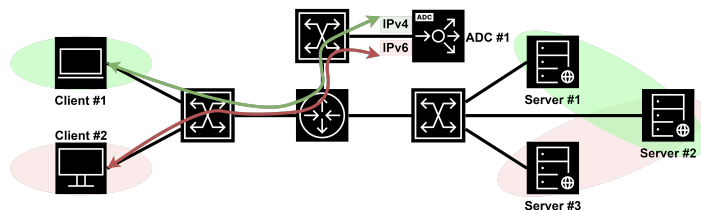


Рис. 1

Только IPv4

IPv4-сессии установлены между Angie ADC и бэкенд-серверами с поддержкой IPv4:

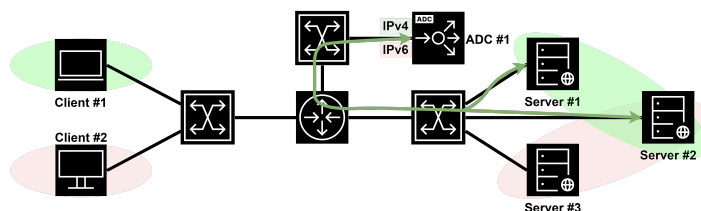


Рис. 2

Только IPv6

IPv6-сессии установлены между Angie ADC и бэкенд-серверами с поддержкой IPv6:

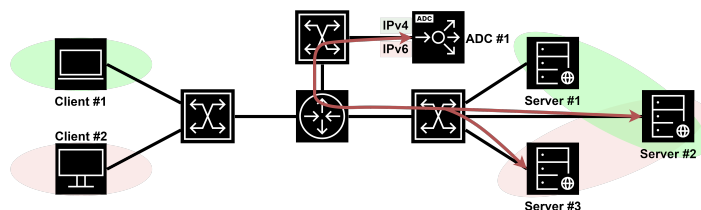


Рис. 3

Смешанные подключения

Angie ADC может проксировать входящие IPv4-подключения в любые сессии (как IPv4, так и IPv6) в сторону бэкенд-серверов. Аналогично и для IPv6-подключений.

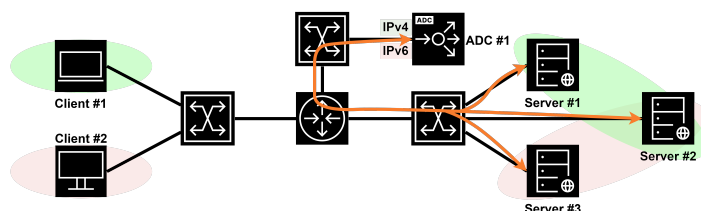


Рис. 4

Таким образом, Angie ADC позволяет гибко проксировать трафик между различными протоколами.

Преимущества такой схемы:

- Практически мгновенная публикация в сети IPv6-сервиса без поддержки IPv6 на бэкенд-серверах.
- Постепенный переход бэкенд-инфраструктуры на IPv6 без потери доступности сервисов снаружи по IPv4, т.е. возможна миграция с IPv4 на IPv6 с минимальными изменениями инфраструктуры.

11.1.5 Настройка iptables

Правила управления доступом для IPv6 похожи на правила для IPv4, но должны настраиваться отдельно вручную.

Пример типовой настройки iptables:

```
[root@angie-va angie-va]# iptables -S
-P INPUT DROP
-P FORWARD DROP
-P OUTPUT ACCEPT
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 8080 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 443 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 53 -j ACCEPT
-A INPUT -p udp -m state --state NEW -m udp --dport 53 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 2022 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 179 -j ACCEPT
-A INPUT -p ospf -j ACCEPT
-A INPUT -p udp -m state --state NEW -m udp --dport 520 -j ACCEPT
-A INPUT -p eigrp -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
```

Для IPv6 соответствующая настройка производится с помощью ip6tables:

```
[root@angie-va angie-va]# ip6tables -S
-P INPUT DROP
-P FORWARD DROP
-P OUTPUT ACCEPT
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 8080 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 443 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 53 -j ACCEPT
-A INPUT -p udp -m state --state NEW -m udp --dport 53 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 2022 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 179 -j ACCEPT
-A INPUT -p ipv6-icmp -j ACCEPT
-A INPUT -p ospf -j ACCEPT
-A INPUT -p udp -m state --state NEW -m udp --dport 521 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp6-adm-prohibited
```

11.2 Настройка пула SNAT (SNAT Pool)

11.2.1 Введение

Пул SNAT (SNAT Pool, Source Network Address Translation Pool) — это набор IP-адресов, которые Angie ADC использует для подмены исходных IP-адресов клиентов при передаче трафика на серверы. SNAT-пулы применяются в высоконагруженных средах с большим количеством одновременно открытых сессий. Они позволяют обойти ограничение на количество одновременных подключений (около 65000 на один IP-адрес). Ограничение связано с количеством доступных клиентских сокетов (TCP- и UDP-портов), используемых для установки соединений. SNAT-пулы могут использоваться как для L4-, так и для L7-балансировки.

i Примечание

Также возможен подход, основанный на горизонтальном масштабировании за счет установки нескольких Angie ADC и распределения трафика между ними. Этот подход позволяет решить проблему не только с ограничением количества сессий, но и снять ограничение на производительность системы, например когда при больших объемах трафика необходимо обрабатывать данные на скоростях более 10 Гбит/с.

11.2.2 Ручная настройка SNAT-пулов

Настройка вручную создаваемых SNAT-пулов происходит в три этапа:

1. Выбор пула адресов и их настройка.
2. Настройка маршрутизации.
3. Настройка правил балансировки.

Пример простой схемы с одним Angie ADC представлен на рисунке ниже:

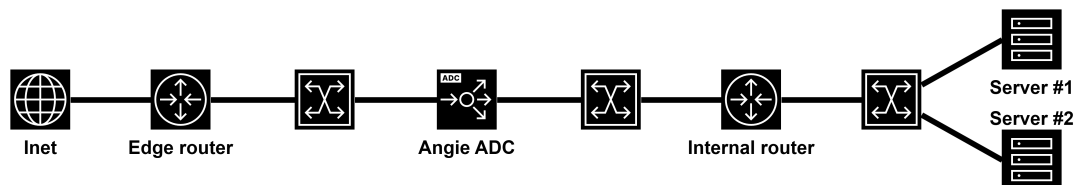


Рис. 2

11.2.2.1 1. Выбор пула адресов и их настройка

Размер пула адресов напрямую зависит от количества сессий, которые требуется одновременно поддерживать. Чем больше сессий должно работать одновременно, тем больше IP-адресов должно входить в пул.

Выбор адресов условно свободный, можно использовать адреса из разных подсетей. Мы не рекомендуем использовать чужие валидные адреса, чтобы избежать случайных пересечений с адресами клиентов. Также лучше выбирать адреса не из произвольного диапазона, а выделить конкретную подсеть, которая будет полностью использоваться в пуле. Это упростит дальнейшую настройку маршрутизации.

В примере ниже мы будем использовать подсеть 192.168.12.20/30, включающую в себя четыре адреса. Так как все четыре адреса будут использоваться в качестве адресов отправителя пакетов на участке сети между Angie ADC и апстримами, то необходимо все эти четыре адреса назначить на сетевой интерфейс системы, например на loopback-интерфейс.

Пример конфигурации представлен ниже. Адреса назначаются с маской /32. Это нужно, чтобы система обрабатывала возвращающийся от серверов трафик и устанавливала полноценные соединения.

```
angie-va#
angie-va# conf t
angie-va(config)# int lo
angie-va(config-if)# ip add 192.168.12.20/32
angie-va(config-if)# ip add 192.168.12.21/32
angie-va(config-if)# ip add 192.168.12.22/32
angie-va(config-if)# ip add 192.168.12.23/32
```

Примечание

В этом примере между апстримами и Angie ADC размещен маршрутизатор. Ситуация, когда апстримы и один из интерфейсов Angie ADC размещены в одной подсети, будет рассмотрена отдельно (см. [Заключение](#) ниже).

11.2.2.2 2. Настройка маршрутизации

В этом примере мы будем использовать протокол BGP для динамической маршрутизации. Настройка других протоколов динамической маршрутизации выполняется аналогично.

В BGP можно анонсировать все четыре адреса из пула независимо (четыре префикса с маской /32). Однако BGP позволяет производить суммаризацию (агрегирование) анонсируемых префиксов, поэтому есть способ лучше — выполнить анонс всего пула целиком в виде одного префикса с маской /30.

Создадим агрегированный маршрут для всей подсети, используемой в качестве пула SNAT:

```
angie-va# conf t
angie-va(config)# ip route 192.168.12.20/30 Null0
angie-va(config)# exi
angie-va# sho ip ro sta
Codes: K - kernel route, C - connected, L - local, S - static,
R - RIP, O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
T - Table, v - VNC, V - VNC-Direct, F - PBR,
f - OpenFabric, t - Table-Direct,
> - selected route, * - FIB route, q - queued, r - rejected, b - backup
t - trapped, o - offload failure
S>* 192.168.12.20/30 [1/0] unreachable (blackhole), weight 1, 00:01:21
```

Получившийся маршрут необходимо анонсировать по протоколу BGP:

```
angie-va# conf t
angie-va(config)# router bg 1
angie-va(config-router)# add ipv4 un
angie-va(config-router-af)# red sta
angie-va(config-router-af)# end
angie-va#
```

Настройку BGP-соседей опустим для краткости (подробнее о настройке BGP см. [BGP-маршрутизация](#)).

Примечание

На представленной выше схеме BGP-соседом должен стать маршрутизатор Internal router, так как трафик от серверов будет возвращаться через него.

Убедимся, что BGP-соседи получают корректный маршрут:

```
internal_router#sho ip bg
BGP table version is 6, local router ID is 192.168.0.150
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found
Network Next Hop Metric LocPrf Weight Path
*>i 192.168.12.20/30 192.168.0.147 0 100 0 ?
internal_router#
```

11.2.2.3 3. Настройка правил балансировки

Для распределения клиентов по адресам из пула можно использовать различные модули, например Geo, Geo IP, Map, Split Clients. Рассмотрим несколько типовых сценариев с описанием используемых модулей.

Сценарий А: использование модуля Split Clients

Стандартная ситуация, когда на Angie ADC попадают подключения пользователей из интернета напрямую. Предполагается, что пользователи распределяются равномерно по всему пространству адресов. В этом случае для настройки можно использовать модуль *Split Clients* (применяется в контексте http). Split_clients работает как при L7-, так и при L4-балансировке.

Пример конфигурации:

```
split_clients "${remote_addr}AAA" $variant {
    25% 192.168.12.20;
    25% 192.168.12.21;
    25% 192.168.12.22;
    25% 192.168.12.23;
}
```

i Примечание

К адресу клиента добавлен текст AAA. Это нужно для изменения значения, выдаваемого хэш-функцией. Если потребуется поменять распределение клиентов по адресам из пула, можно поменять добавляемый текст.

Сценарий В: использование модуля Map

Возможна ситуация, когда перед Angie ADC с точки зрения трафика, идущего от клиентов, расположен другой балансировщик:

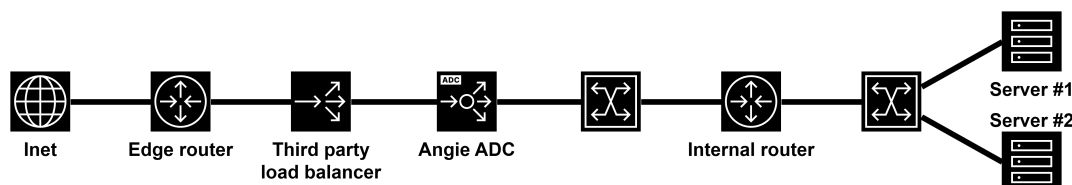


Рис. 3

Если этот балансировщик использует SNAT-пул, то переменная `remote_addr` может не обеспечить достаточного разнообразия. Это может привести к коллизиям — перекосу в распределении клиентов по адресам из пула SNAT. Решить указанную проблему можно, например, используя модуль

Map, с помощью которого можно в явном виде задать соответствие «старого» и «нового» адресов отправителя. Метод *map* работает как при L7-, так и при L4-балансировке.

Пример конфигурации (восемь адресов из предыдущего пула привязываются к четырем адресам пула SNAT):

```
map $remote_addr $variant {
    "10.10.10.0" "192.168.12.20";
    "10.10.10.1" "192.168.12.20";
    "10.10.10.2" "192.168.12.21";
    "10.10.10.3" "192.168.12.21";
    "10.10.10.4" "192.168.12.22";
    "10.10.10.5" "192.168.12.22";
    "10.10.10.6" "192.168.12.23";
    "10.10.10.7" "192.168.12.23";
}
```

Сценарий С: модуль *Split Clients* и хэш переменной *\$request_id*

Для балансировки HTTP-трафика можно использовать подход, при котором не важно, используется ли какой-либо NAT/PAT или прокси между реальным клиентом и Angie ADC. Для распределения клиентов будет использоваться модуль *Split Clients* и хэш от переменной *\$request_id*. Значение переменной генерируется автоматически и, по сути, является случайным числом. Этот метод имеет ограниченную область применения — L7-балансировка для протокола HTTP.

Пример конфигурации:

```
split_clients "${request_id}" $variant {
    25% 192.168.12.20;
    25% 192.168.12.21;
    25% 192.168.12.22;
    25% 192.168.12.23;
}
```

i Примечание

При использовании этого метода разные сессии одного и того же клиента могут привязываться к различным адресам из пула. Это не должно быть большой проблемой, так как современные веб-системы используют заголовок *X-Forwarded-For* для определения адреса клиента и не смотрят на адрес отправителя в IP-пакете.

Теперь с помощью директивы *proxy_bind* укажем, где и как использовать указанное распределение клиентов. Директиву можно задать в контекстах *http*, *server* или *location*.

Пример:

```
location / {
    proxy_pass http://backend ;
    proxy_bind ${variant};
}
```

11.2.3 Заключение

Предложенные выше способы ручного создания SNAT-пула позволят уверенно перешагнуть ограничение в 65000 одновременных сессий как для L4-, так и для L7-балансировки. Однако следует все же понимать, что ограничение количества одновременно поддерживаемых сессий зависит не только от количества сокетов, но и от объемов использования целого ряда других системных ресурсов.

Чтобы адаптировать используемый подход к ситуации, когда один из интерфейсов Angie ADC расположен в одной подсети с апстрим-серверами, потребуется следующее:

- не использовать динамическую маршрутизацию;
- настроить адреса из пула на физическом интерфейсе, а не на loopback-интерфейсе (адреса должны быть из той же IP-подсети, в которой расположены серверы);
- для обеспечения отказоустойчивости можно использовать несколько Angie ADC, а IP-адреса из пула указать как виртуальные дополнительные адреса для VRRP, вместо того чтобы назначать их непосредственно на физический интерфейс.

11.3 Настройка Transparent Proxy для TCP- и UDP-трафика

Angie ADC работает как обратный прокси для TCP- и UDP-трафика. Апстрим-серверы видят, что весь трафик приходит с IP-адреса самого прокси Angie ADC:

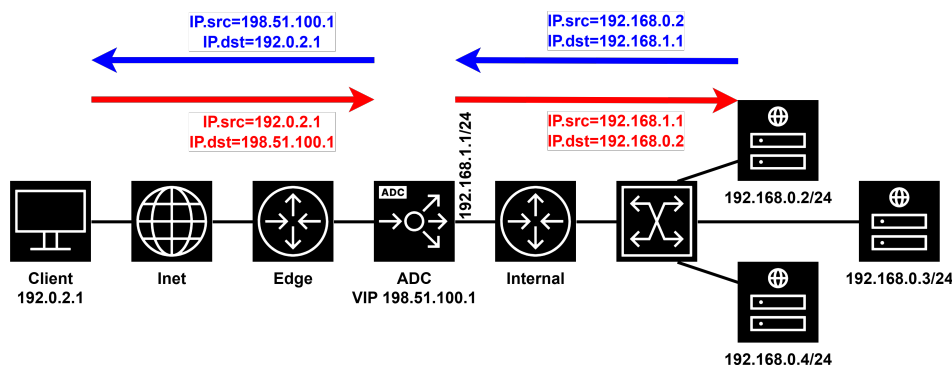


Рис. 1: красным цветом обозначен поток трафика от клиента к сервису, синим — обратный трафик.

Для определения и передачи настоящего IP-адреса клиента апстрим-серверу доступны следующие два метода:

- *IP Transparency* — апстрим-серверы видят соединение как исходящее от реального клиента (для TCP- и UDP-трафика).
- *Direct Server Return (DSR)* — дополнительно позволяет отправлять ответы от апстрим-серверов напрямую клиентам, минуя промежуточный балансировщик (только для UDP-трафика). Метод может быть реализован через NAT.

11.3.1 IP Transparency

При использовании IP Transparency апстрим-серверы будут видеть настоящий IP-адрес клиента, с которого приходят пакеты. IP Transparency может использоваться как с TCP-, так и с UDP-протоколами.

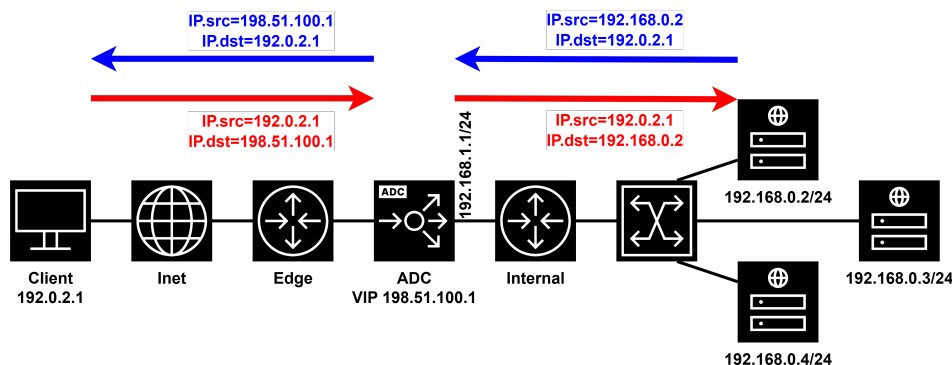


Рис. 2: пример сети, в которой на системе балансировки Angie ADC включена опция IP Transparency.

11.3.1.1 Предусловия

Должно выполняться одно из следующих условий:

- Балансировщик и апстрим-серверы находятся в одной подсети, балансировщик является маршрутизатором по умолчанию (default gateway) для апстрим-серверов.
- Балансировщик находится на пути передачи трафика между клиентом и апстрим-серверами (трафик проходит через балансировщик в обоих направлениях).

11.3.1.2 Пример настройки балансировщика

```
# in the 'http' context
upstream http_upstreams {
    server 192.168.0.2;
    server 192.168.0.3;
    server 192.168.0.4;
}

server {
    listen 80;

    location / {
        proxy_bind $remote_addr transparent;
        proxy_pass http://http_upstreams;
    }
}
```

В этом примере:

- Настроен простой виртуальный HTTP-сервер, распределяющий трафик между группой апстрим-серверов.
- Исходный адрес для трафика, направляемого на апстрим-серверы, подменяется с помощью параметра `transparent` в директиве `proxy_bind`. Чаще всего в качестве исходного адреса указывается адрес удаленного клиента.

11.3.1.3 Настройка маршрутизации

При использовании IP Transparency трафик должен проходить через Angie ADC и обрабатываться им. Для этого необходимо настроить корректную маршрутизацию.

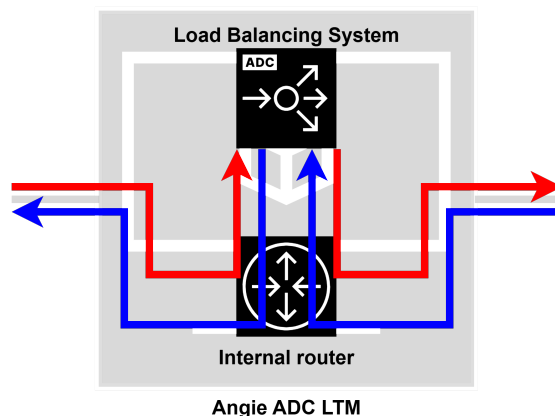


Рис. 3: движение трафика через Angie ADC с настроенным IP Transparency.

Маршрутизация настраивается на основе политик PBR (Policy-Based Routing):

1. Подключитесь к *интерфейсу командной строки* на порту 2222.
2. Добавьте маршрутизацию всего трафика в локальный интерфейс lo в таблице 100:

```
ip route add local 0.0.0.0/0 dev lo table 100
```

3. Добавьте правила: весь трафик с апстрим-серверов необходимо обрабатывать с помощью таблицы маршрутизации 100:

```
ip rule add from 192.168.0.2 table 100
ip rule add from 192.168.0.3 table 100
ip rule add from 192.168.0.4 table 100
```

Вместо добавления отдельных серверов можно добавить правило для подсети, если это необходимо (например, `ip rule add from 10.21.20.0/24 table 100`).

4. Проверьте, что весь трафик обрабатывается в таблице 100 и что есть только один маршрут — весь трафик уходит в локальный интерфейс lo:

```
# ip rule
0: from all lookup local
32764: from 192.168.0.2 lookup 100
32765: from 192.168.0.3 lookup 100
32766: from 192.168.0.4 lookup 100
32767: from all lookup main
32768: from all lookup default

# ip route show table 100
local default dev lo scope host
```

11.3.2 Direct Server Return (DSR)

Примечание

Применение DSR возможно только для UDP-трафика.

При использовании Direct Server Return (DSR) апстрим-серверы будут видеть настоящий IP-адрес клиента, с которого приходят пакеты, и будут отправлять ответ непосредственно удаленному клиенту (т. е. пакеты с ответом полностью обходят балансировщик нагрузки). DSR может дать небольшой прирост производительности:

- Позволяет не удерживать открытыми UDP-сокеты в ожидании пакета ответа (повышает масштабируемость).
- Пакеты с ответами могут полностью обходить L7-обработку (снижает задержку).

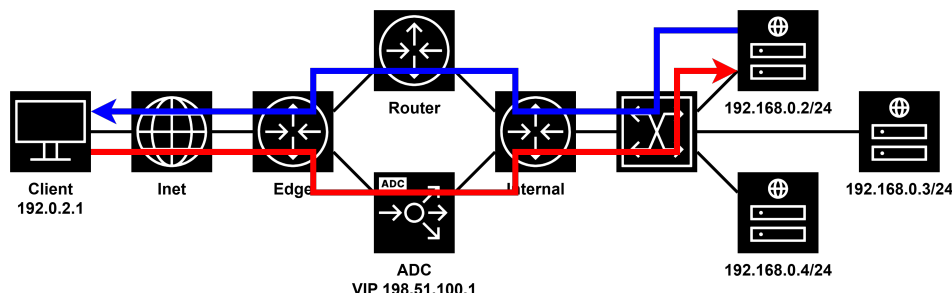


Рис. 4: движение трафика через Angie ADC с DSR.

Ограничения при использовании DSR:

- Балансировщик нагрузки не видит пакетов с ответами, поэтому не может определить, отвечает ли апстрим-сервер или вышел из строя.
- Балансировщик не может анализировать запрос глубже первого пакета, чтобы выбрать апстрим-сервер, поэтому его возможности по принятию решений о балансировке (например, маршрутизация на основе содержимого) ограничены.
- Балансировщик не может участвовать в процессах согласования или обработке состояний, таких как SSL/TLS.
- Такие функции, как кэширование и логирование, недоступны при использовании DSR.

Кроме того при использовании DSR необходимо:

- переписывать исходный адрес ответа, чтобы он соответствовал публичному адресу балансировщика нагрузки;
- настроить активные проверки для апстрим-серверов.

11.3.2.1 Пример настройки балансировщика

```
# in the 'stream' context
server {
    listen 53 udp;

    proxy_bind $remote_addr:$remote_port transparent;
    proxy_responses 0;
    #proxy_timeout 1s;
}

upstream dns_upstreams {
    server 192.168.0.2:53;
    server 192.168.0.3:53;
    server 192.168.0.4:53;
}
```

В этом примере:

- Создан кластер из трех DNS-серверов с балансировкой нагрузки. Каждый из DNS-серверов должен возвращать разные IP-адреса при запросах `www.example.com`.
- Настроена простая конфигурация обратного прокси, которая распределяет нагрузку между этими DNS-серверами.
- Параметр `transparent` в директиве `proxy_bind` позволяет апстрим-серверу видеть исходный IP-адрес и порт (переменные `$remote_addr` и `$remote_port`).
- Балансировщик не ожидает ответ от апстрим-серверов (директива `proxy_responses=0`). Ответ будет отправлен клиенту напрямую.

11.3.2.2 Переписывание IP-адреса и порта при ответе

Апстрим-сервер генерирует ответные пакеты, в которых в качестве исходного адреса используется его локальный IP-адрес. Этот исходный адрес необходимо переписать на IP-адрес и порт балансировщика нагрузки, к которому изначально подключался клиент. Есть два способа для переписывания исходящих пакетов:

- Router NAT – переписывание на промежуточном маршрутизаторе.
- Origin NAT – переписывание при отправке пакетов с каждого апстрим-DNS-сервера.

Router NAT

Вы можете изменять (переписывать) исходящие пакеты на промежуточном маршрутизаторе, который является маршрутом по умолчанию для апстрим-серверов.

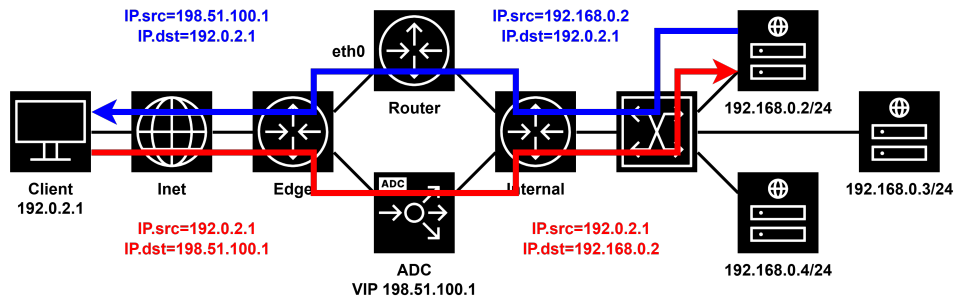


Рис. 5: движение трафика с Router NAT

Для этого настройте на маршрутизаторе stateless NAT-переписывание:

Например:

```
# tc qdisc add dev eth0 root handle 10: htb
# tc filter add dev eth0 parent 10: protocol ip prio 10 u32 match ip src 192.168.0.2
  ↳ match ip sport 53 action nat egress 192.168.0.2 198.51.100.1
# tc filter add dev eth0 parent 10: protocol ip prio 10 u32 match ip src 192.168.0.3
  ↳ match ip sport 53 action nat egress 192.168.0.3 198.51.100.1
# tc filter add dev eth0 parent 10: protocol ip prio 10 u32 match ip src 192.168.0.4
  ↳ match ip sport 53 action nat egress 192.168.0.4 198.51.100.1
```

Убедитесь, что вы указали корректный исходящий интерфейс (например, eth0) и правильные IP-адреса для каждого апстрим-сервера. В зависимости от конфигурации можно сократить количество команд `tc filter`, объединив их с помощью CIDR-масок для параметров `src` и `egress old`.

Чтобы вывести текущую конфигурацию `tc filter`, выполните:

```
# tc filter show dev eth0
```

Origin NAT

Вариант с переписыванием исходящих пакетов при их отправке с каждого апстрим-DNS-сервера применим, если у вас есть возможность настраивать сеть на апстрим-серверах, особенно если они напрямую подключены к интернету.

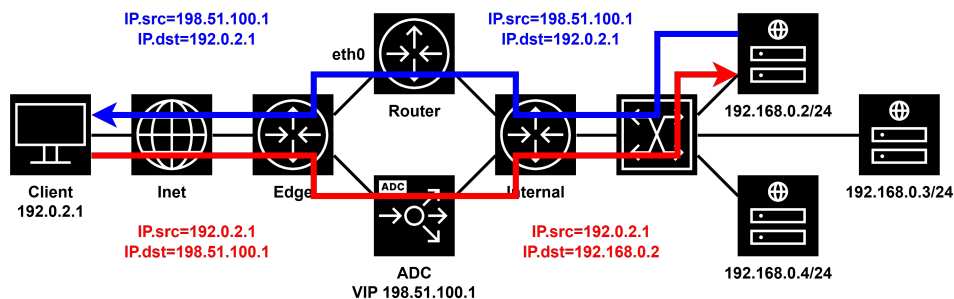


Рис. 6: движение трафика с Origin NAT

Необходимо настроить stateless NAT-переписывание на каждом апстрим-сервере.

Пример:

```
# tc qdisc add dev eth0 root handle 10: htb
# tc filter add dev eth0 parent 10: protocol ip prio 10 u32 match ip src 192.168.0.2
↪match ip sport 53 action nat egress 192.168.0.2 198.51.100.1
```

Такую конфигурацию необходимо применить на каждом апстрим-сервере, убедившись, что выбран корректный интерфейс и корректные IP-адреса.

11.4 Балансировка трафика на основе набора шифров

В этой статье рассматривается балансировка трафика на основе набора шифров. На один VIP-адрес балансировщика нагрузки поступает два типа трафика:

- TLS-трафик с шифрованием ГОСТ;
- TLS-трафик без ГОСТ.

Для стандартного TLS выполняется Offload и балансировка на Angie ADC. Трафик ГОСТ направляется на отдельный бэкенд-сервер без расшифровки.

11.4.1 Настройка

Конфигурация основного бэкенд-сервера:

```
http {
    default_type text/plain;

    server {
        listen 80;
        server_name localhost;

        location / {
            return 200 "Hello, this is default backend\n";
        }
    }
}
```

Конфигурация бэкенд-сервера для ГОСТ-трафика:

```
http {
    default_type text/plain;

    server {
        listen 443 ssl;
        server_name localhost;
        ssl_certificate /etc/angie-lb/crt/server.crt;
        ssl_certificate_key /etc/angie-lb/crt/server.key;
        ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
        ssl_ciphers IANA-GOST2012-GOST8912-GOST8912:GOST2012-MAGMA-MAGMAOMAC:GOST2012-
↪KUZNYECHIK-KUZNYECHIKOMAC;
        ssl_prefer_server_ciphers on;

        location / {
            return 200 "Hello, this is GOST-based server\n";
        }
    }
}
```

11.4.1.1 Случай 1

Трафик балансируется на ГОСТ-бэкенд, если ГОСТ присутствует в списке шифров (может присутствовать наряду с другими шифрами).

Конфигурация:

```
http {

    upstream backend {
        server IP.OF.MAIN.BACKEND:80;
    }

    server {
        listen      127.0.0.1:8443 ssl;
        ssl_certificate /etc/angie-lb/crt/cp.crt;
        ssl_certificate_key /etc/angie-lb/crt/cp.key;
        server_name localhost;
        status_zone default;
        location / {
            proxy_pass http://backend;
        }
    }
}

stream {

    map $ssl_preread_ciphers $handler {
        "~TLS_GOST" 127.0.0.1:9443;
        default     127.0.0.1:8443;
    }

    server {
        listen      443;
        ssl_preread on;
        pass $handler;
    }

    server {
        listen      127.0.0.1:9443;
        proxy_pass  IP.OF.GOST.BACKEND:443;
    }
}
```

11.4.1.2 Случай 2

Трафик балансируется на ГОСТ-бэкенд, если заданы только ГОСТ-шифры. Если помимо ГОСТ-шифров есть другие шифры, трафик отправляется на основной бэкенд.

Конфигурация:

```
http {

    upstream backend {
        server IP.OF.MAIN.BACKEND:80;
    }

    server {
        listen      127.0.0.1:8443 ssl;
```

```

    ssl_certificate /etc/angie-lb/crt/cp.crt;
    ssl_certificate_key /etc/angie-lb/crt/cp.key;
    server_name localhost;
    status_zone default;
    location / {
        proxy_pass http://backend;
    }
}

stream {

    map $ssl_preread_ciphers $backend {
        "^(TLS_GOST[^\:]*)(:TLS_GOST[^\:]*)*(:TLS_EMPTY_RENEGOTIATION_INFO_SCSV)?$"
↪127.0.0.1:9443;
        default 127.0.0.1:8443;
    }

    server {
        listen 443;
        ssl_preread on;
        pass $backend;
    }

    server {
        listen 127.0.0.1:9443;
        proxy_pass IP.OF.GOST.BACKEND:443;
    }
}

```

11.4.2 Проверка

11.4.2.1 Случай 1

Трафик без ГОСТ отправляется на основной бэкенд:

```
# curl --ciphers '!IANA-GOST2012-GOST8912-GOST8912:!GOST2012-MAGMA-MAGMAOMAC:!  
↪GOST2012-KUZNYECHIK-KUZNYECHIKOMAC:HIGH' --tlsv1.2 --tls-max 1.2 https://example.com  
Hello, this is default backend
```

Трафик с ГОСТ отправляется на ГОСТ-бэкенд:

```
# curl --ciphers 'ALL' --tlsv1.2 --tls-max 1.2 https://example.com  
Hello, this is GOST-based server
```

11.4.2.2 Случай 2

Трафик без ГОСТ отправляется на основной бэкенд:

```
# curl --ciphers '!IANA-GOST2012-GOST8912-GOST8912:!GOST2012-MAGMA-MAGMAOMAC:!  
↪GOST2012-KUZNYECHIK-KUZNYECHIKOMAC:HIGH' --tlsv1.2 --tls-max 1.2 https://example.com  
Hello, this is default backend
```

Трафик с ГОСТ отправляется на основной бэкенд:

```
# curl --ciphers 'ALL' --tlsv1.2 --tls-max 1.2 https://example.com  
Hello, this is default backend
```


Трафик, где только ГОСТ, отправляется на ГОСТ-бэкенд:

```
# curl --ciphers 'IANA-GOST2012-GOST8912-GOST8912:GOST2012-MAGMA-MAGMAOMAC:GOST2012-  
↪KUZNYECHIK-KUZNYECHIKOMAC' --tlsv1.2 --tls-max 1.2 https://example.com  
Hello, this is GOST-based server
```

ГЛАВА 12

Миграция с других решений

Для облегчения процесса миграции с других решений на Angie ADC используйте функции Angie ADC, заменяющие аналогичные решения у других производителей.

В этом разделе собраны примеры настройки таких функций. Список будет пополняться.

- *Миграция с OneConnect profile от F5*
- *Миграция с Use Source IP Mode (USIP) от Citrix NetScaler*

12.1 Keepalive как аналог OneConnect profile от F5

Для повторного использования уже открытых TCP-соединений с серверами бэкенда используйте механизм *keepalive*.

Особенности:

- Если в upstream-группе определено несколько серверов, то keepalive-соединения устанавливаются с каждым из них в соответствии с выбранным методом балансировки.
- Мастер-процесс запускает несколько рабочих процессов в зависимости от настройки директивы *worker_processes* и у каждого рабочего процесса будет собственный пул keepalive-соединений:
 - Если все запросы обрабатывает один рабочий процесс, то они могут проксироваться в одно и то же установленное соединение.
 - Если запросы распределяются между несколькими рабочими процессами, то для каждого процесса формируются свои keepalive-соединения с бэкендом.

12.1.1 Настройка

1. Задайте директиву *keepalive* в блоке *upstream*.

Пример:

```
upstream http_backend {
    server 192.168.0.1;
    server 192.168.0.2;
```

```
keepalive 32;
}
```

- Для HTTP установите директиву `proxy_http_version` в значение 1.1, а поле заголовка `Connection` в `proxy_set_header` очистите:

Пример:

```
server {
    # ...

    location / {
        proxy_pass http://http_backend;
        proxy_http_version 1.1;
        proxy_set_header Connection "";
        # ...
    }
}
```

- Если требуется разделять потоки по IP-адресам, используйте привязку по `sticky learn` в группе `upstream` в сочетании с директивой `geo`.

Пример:

```
geo $idmask {
    ranges;
    default                ZZ;
    10.21.0.0-10.21.0.255  tau;
    10.21.4.0-10.21.4.255  chi;
    10.21.8.0-10.21.8.255  alpha;
    10.21.20.0-10.21.20.255 nu;
}

upstream http_backend {
    server 10.21.8.6:80;
    server 10.21.8.13:80;

    sticky learn
        create=$idmask
        lookup=$idmask
        zone=client_sessions:1m
        timeout=60s;

    keepalive 32;
}

server {
    listen      10.21.8.8:80;
    server_name localhost;

    location / {
        proxy_http_version 1.1;
        proxy_set_header Connection "";

        proxy_pass http://http_backend;
    }
}
```

В этом случае в пределах одного рабочего процесса поведение Angie ADC будет очень похоже

на механизм OneConnect в F5 BIG-IP.

12.2 Передача IP-адреса клиента как аналог Use Source IP Mode (USIP) от Citrix NetScaler

Angie ADC может передавать настоящий IP-адрес клиента апстрим-серверу.

Для HTTP- и HTTPS-трафика используется HTTP-заголовок **X-Forwarded-For**.

Пример:

```
http {
    server {
        listen 80;

        location / {
            proxy_pass http://127.0.0.1:8080;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        }
    }
}
```

Для TCP- и UDP-трафика используется протокол PROXY.

Пример:

```
stream {
    server {
        listen 80;

        proxy_pass 127.0.0.1:8080;
        proxy_protocol on;
    }
}
```

Также для TCP- и UDP-трафика возможно использование механизма *Transparent Proxy*.

ГЛАВА 13

Права на интеллектуальную собственность

Документация на программный продукт Angie ADC является интеллектуальной собственностью ООО «Веб-Сервер».

Copyright © 2026, ООО «Веб-Сервер». Все права защищены.

Алфавитный указатель

С

сертификаты (*API*), 153

А

absolute_redirect (*http*), 70
accept_mutex (*core-ADC*), 60
accept_mutex_delay (*core-ADC*), 60
access_log (*http*), 222
access_log (*stream*), 427
acme (*http*), 118
acme (*stream*), 422
acme_client (*http*), 119
acme_client_path (*http*), 122
acme_dns_port (*http*), 122
acme_dns_ttl (*http*), 122
acme_hook (*http*), 122
acme_http_port (*http*), 123
acme_max_response_size (*http*), 123
add_after_body (*http*), 125
add_before_body (*http*), 125
add_header (*http*), 215
add_trailer (*http*), 215
addition_types (*http*), 125
aio (*http*), 70
aio_write (*http*), 72
alias (*http*), 72
allow (*http*), 117
allow (*stream*), 421
ancient_browser (*http*), 167
ancient_browser_value (*http*), 168
api (*http*), 126
api_config_files (*http*), 127
auth_basic (*http*), 162
auth_basic_user_file (*http*), 162
auth_delay (*http*), 73
auth_http, 527
auth_http_header, 527
auth_http_pass_client_cert, 527
auth_http_timeout, 527
auth_request (*http*), 164
auth_request_set (*http*), 164
auto_redirect (*http*), 73
autoindex (*http*), 164

autoindex_exact_size (*http*), 164
autoindex_format (*http*), 164
autoindex_localtime (*http*), 166

В

backup_switch (*http*), 360
backup_switch (*stream*), 485
bind_conn (*http*), 361
break (*http*), 305

С

charset (*http*), 169
charset_map (*http*), 169
charset_types (*http*), 170
chunked_transfer_encoding (*http*), 73
client (*http*), 73
client_body_buffer_size (*http*), 75
client_body_in_file_only (*http*), 75
client_body_in_single_buffer (*http*), 75
client_body_temp_path (*http*), 75
client_body_timeout (*http*), 76
client_header_buffer_size (*http*), 76
client_header_timeout (*http*), 76
client_max_body_size (*http*), 76
connection_pool_size (*http*), 77
create_full_put_path (*http*), 171

Д

daemon (*core-ADC*), 60
dav_access (*http*), 171
dav_methods (*http*), 172
debug_connection (*core-ADC*), 60
debug_points (*core-ADC*), 61
default_type (*http*), 77
deny (*http*), 117
deny (*stream*), 421
directio (*http*), 77
directio_alignment (*http*), 78
disable_symlinks (*http*), 78
doh_bind (*http*), 173
doh_buffer_size (*http*), 174
doh_connect_timeout (*http*), 174
doh_max_size (*http*), 174

doh_next_upstream (*http*), 174
doh_next_upstream_timeout (*http*), 175
doh_next_upstream_tries (*http*), 175
doh_pass (*http*), 175
doh_read_timeout (*http*), 175
doh_send_timeout (*http*), 176
doh_socket_keepalive (*http*), 176
doh_transport (*http*), 176

E

early_hints (*http*), 79
empty_gif (*http*), 177
env (*core-ADC*), 61
error_log (*core-ADC*), 62
error_log_user_tag (*http*), 79
error_log_user_tag (*stream*), 507
error_page (*http*), 79
etag (*http*), 80
events (*core-ADC*), 62
expires (*http*), 216

F

fastcgi_bind (*http*), 177
fastcgi_buffer_size (*http*), 178
fastcgi_buffering (*http*), 178
fastcgi_buffers (*http*), 179
fastcgi_busy_buffers_size (*http*), 179
fastcgi_cache (*http*), 180
fastcgi_cache_background_update (*http*), 180
fastcgi_cache_bypass (*http*), 180
fastcgi_cache_key (*http*), 181
fastcgi_cache_lock (*http*), 181
fastcgi_cache_lock_age (*http*), 181
fastcgi_cache_lock_timeout (*http*), 181
fastcgi_cache_max_range_offset (*http*), 181
fastcgi_cache_methods (*http*), 182
fastcgi_cache_min_uses (*http*), 182
fastcgi_cache_path (*http*), 182
fastcgi_cache_revalidate (*http*), 185
fastcgi_cache_use_stale (*http*), 185
fastcgi_cache_valid (*http*), 186
fastcgi_catch_stderr (*http*), 187
fastcgi_connect_timeout (*http*), 187
fastcgi_connection_drop (*http*), 188
fastcgi_force_ranges (*http*), 188
fastcgi_hide_header (*http*), 188
fastcgi_ignore_client_abort (*http*), 188
fastcgi_ignore_headers (*http*), 188
fastcgi_index (*http*), 189
fastcgi_intercept_errors (*http*), 189
fastcgi_interface (*http*), 189
fastcgi_keep_conn (*http*), 190
fastcgi_limit_rate (*http*), 190
fastcgi_max_temp_file_size (*http*), 190
fastcgi_next_upstream (*http*), 191
fastcgi_next_upstream_timeout (*http*), 192
fastcgi_next_upstream_tries (*http*), 192
fastcgi_no_cache (*http*), 192

fastcgi_param (*http*), 192
fastcgi_pass (*http*), 193
fastcgi_pass_header (*http*), 194
fastcgi_pass_request_body (*http*), 194
fastcgi_pass_request_headers (*http*), 194
fastcgi_read_timeout (*http*), 194
fastcgi_request_buffering (*http*), 194
fastcgi_send_lowat (*http*), 195
fastcgi_send_timeout (*http*), 195
fastcgi_socket_keepalive (*http*), 195
fastcgi_split_path_info (*http*), 196
fastcgi_store (*http*), 196
fastcgi_store_access (*http*), 197
fastcgi_tcp_congestion (*http*), 197
fastcgi_temp_file_write_size (*http*), 197
fastcgi_temp_path (*http*), 198
feedback (*http*), 362
feedback (*stream*), 486
flv (*http*), 199

G

geo (*http*), 199
geo (*stream*), 423
goto (*http*), 305
grpc_bind (*http*), 201
grpc_buffer_size (*http*), 202
grpc_connect_timeout (*http*), 202
grpc_connection_drop (*http*), 202
grpc_hide_header (*http*), 203
grpc_ignore_headers (*http*), 203
grpc_intercept_errors (*http*), 203
grpc_interface (*http*), 203
grpc_next_upstream (*http*), 204
grpc_next_upstream_timeout (*http*), 204
grpc_next_upstream_tries (*http*), 205
grpc_pass (*http*), 205
grpc_pass_header (*http*), 206
grpc_read_timeout (*http*), 206
grpc_send_timeout (*http*), 206
grpc_set_header (*http*), 206
grpc_socket_keepalive (*http*), 207
grpc_ssl_certificate (*http*), 207
grpc_ssl_certificate_cache (*http*), 207
grpc_ssl_certificate_key (*http*), 207
grpc_ssl_ciphers (*http*), 208
grpc_ssl_conf_command (*http*), 208
grpc_ssl_crl (*http*), 209
grpc_ssl_name (*http*), 209
grpc_ssl_password_file (*http*), 209
grpc_ssl_protocols (*http*), 209
grpc_ssl_server_name (*http*), 209
grpc_ssl_session_reuse (*http*), 210
grpc_ssl_trusted_certificate (*http*), 210
grpc_ssl_verify (*http*), 210
grpc_ssl_verify_depth (*http*), 210
grpc_tcp_congestion (*http*), 210
gunzip (*http*), 211
gunzip_buffers (*http*), 211

gzip (*http*), 212
gzip_buffers (*http*), 212
gzip_comp_level (*http*), 212
gzip_disable (*http*), 212
gzip_http_version (*http*), 213
gzip_min_length (*http*), 213
gzip_proxied (*http*), 213
gzip_static (*http*), 215
gzip_types (*http*), 214
gzip_vary (*http*), 214

H

hash (*http*), 363
hash (*stream*), 487
http (*http*), 80
http2 (*http*), 413
http2_body_preread_size (*http*), 413
http2_chunk_size (*http*), 414
http2_max_concurrent_pushes (*http*), 414
http2_max_concurrent_streams (*http*), 414
http2_push (*http*), 414
http2_push_preload (*http*), 415
http2_recv_buffer_size (*http*), 415
http3 (*http*), 417
http3_hq (*http*), 417
http3_max_concurrent_streams (*http*), 417
http3_max_table_capacity (*http*), 417
http3_stream_buffer_size (*http*), 418

I

if (*http*), 305
if_modified_since (*http*), 81
ignore_invalid_headers (*http*), 81
imap_auth, 530
imap_capabilities, 530
imap_client_buffer, 530
include (*core-ADC*), 63
index (*http*), 217
internal (*http*), 81
ip_hash (*http*), 364

J

js_access (*stream-ADC*), 558
js_body_filter (*http-ADC*), 549
js_content (*http-ADC*), 549
js_context_reuse (*http-ADC*), 549
js_context_reuse (*stream-ADC*), 559
js_engine (*http-ADC*), 550
js_engine (*stream-ADC*), 559
js_fetch_buffer_size (*http-ADC*), 550
js_fetch_buffer_size (*stream-ADC*), 559
js_fetch_ciphers (*http-ADC*), 550
js_fetch_ciphers (*stream-ADC*), 559
js_fetch_keepalive (*http-ADC*), 551
js_fetch_keepalive (*stream-ADC*), 561
js_fetch_keepalive_requests (*http-ADC*), 552
js_fetch_keepalive_requests (*stream-ADC*), 561

js_fetch_keepalive_time (*http-ADC*), 552
js_fetch_keepalive_time (*stream-ADC*), 561
js_fetch_keepalive_timeout (*http-ADC*), 552
js_fetch_keepalive_timeout (*stream-ADC*), 562
js_fetch_max_response_buffer_size (*http-ADC*), 550
js_fetch_max_response_buffer_size (*stream-ADC*), 559
js_fetch_protocols (*http-ADC*), 550
js_fetch_protocols (*stream-ADC*), 560
js_fetch_timeout (*http-ADC*), 551
js_fetch_timeout (*stream-ADC*), 560
js_fetch_trusted_certificate (*http-ADC*), 551
js_fetch_trusted_certificate (*stream-ADC*), 560
js_fetch_verify (*http-ADC*), 551
js_fetch_verify (*stream-ADC*), 560
js_fetch_verify_depth (*http-ADC*), 551
js_fetch_verify_depth (*stream-ADC*), 560
js_filter (*stream-ADC*), 562
js_header_filter (*http-ADC*), 553
js_import (*http-ADC*), 553
js_import (*stream-ADC*), 562
js_path (*http-ADC*), 553
js_path (*stream-ADC*), 563
js_periodic (*http-ADC*), 553
js_periodic (*stream-ADC*), 563
js_preload_object (*http-ADC*), 554
js_preload_object (*stream-ADC*), 564
js_preread (*stream-ADC*), 564
js_set (*http-ADC*), 555
js_set (*stream-ADC*), 564
js_shared_dict_zone (*http-ADC*), 555
js_shared_dict_zone (*stream-ADC*), 565
js_var (*http-ADC*), 556
js_var (*stream-ADC*), 566

K

keepalive (*http*), 364
keepalive_disable (*http*), 82
keepalive_requests (*http*), 83, 366
keepalive_time (*http*), 83, 367
keepalive_timeout (*http*), 83, 367

L

large_client_header_buffers (*http*), 84
least_bandwidth (*http*), 367
least_bandwidth (*stream*), 487
least_conn (*http*), 368
least_conn (*stream*), 488
least_packets (*stream*), 488
least_time (*http*), 368
least_time (*stream*), 489
limit_conn (*http*), 218
limit_conn (*stream*), 425
limit_conn_dry_run (*http*), 218
limit_conn_dry_run (*stream*), 425

limit_conn_log_level (*http*), 219
limit_conn_log_level (*stream*), 426
limit_conn_status (*http*), 219
limit_conn_zone (*http*), 219
limit_conn_zone (*stream*), 426
limit_except (*http*), 84
limit_rate (*http*), 84
limit_rate_after (*http*), 85
limit_req (*http*), 220
limit_req_dry_run (*http*), 221
limit_req_log_level (*http*), 221
limit_req_status (*http*), 221
limit_req_zone (*http*), 221
lingering_close (*http*), 85
lingering_time (*http*), 86
lingering_timeout (*http*), 86
listen (*http*), 86
listen (*mail-ADC*), 522
listen (*stream*), 507
location (*http*), 90
lock_file (*core-ADC*), 63
log_format (*http*), 224
log_format (*stream*), 428
log_not_found (*http*), 93
log_subrequest (*http*), 93

M

mail (*mail-ADC*), 524
map (*http*), 225
map (*stream*), 429
map_hash_bucket_size (*http*), 227
map_hash_bucket_size (*stream*), 430
map_hash_max_size (*http*), 227
map_hash_max_size (*stream*), 430
master_process (*core-ADC*), 63
max_commands (*mail-ADC*), 524
max_errors (*mail-ADC*), 524
max_headers (*http*), 93
max_ranges (*http*), 93
memcached_bind (*http*), 228
memcached_buffer_size (*http*), 228
memcached_connect_timeout (*http*), 228
memcached_gzip_flag (*http*), 229
memcached_interface (*http*), 229
memcached_next_upstream (*http*), 229
memcached_next_upstream_timeout (*http*), 230
memcached_next_upstream_tries (*http*), 230
memcached_pass (*http*), 230
memcached_read_timeout (*http*), 231
memcached_send_timeout (*http*), 231
memcached_socket_keepalive (*http*), 231
memcached_tcp_congestion (*http*), 231
merge_slashes (*http*), 93
metric (*http*), 233
metric (*stream*), 432
metric_complex_zone (*http*), 234
metric_complex_zone (*stream*), 433
metric_zone (*http*), 235

metric_zone (*stream*), 434
min_delete_depth (*http*), 172
mirror (*http*), 250
mirror_request_body (*http*), 250
modern_browser (*http*), 168
modern_browser_value (*http*), 168
mp4 (*http*), 252
mp4_buffer_size (*http*), 252
mp4_limit_rate (*http*), 252
mp4_limit_rate_after (*http*), 253
mp4_max_buffer_size (*http*), 253
mp4_start_key_frame (*http*), 253
mqtt_preread (*stream*), 449
msie_padding (*http*), 94
msie_refresh (*http*), 94
multi_accept (*core-ADC*), 63

N

no_port (*stream*), 489

O

open_file_cache (*http*), 94
open_file_cache_errors (*http*), 95
open_file_cache_events (*http*), 95
open_file_cache_min_uses (*http*), 95
open_file_cache_valid (*http*), 95
open_log_file_cache (*http*), 224
open_log_file_cache (*stream*), 428
output_buffers (*http*), 96
override_charset (*http*), 170

P

pass (*stream*), 451
pcre_jit (*core-ADC*), 64
pid (*core-ADC*), 64
pop3_auth, 531
pop3_capabilities, 531
port_in_redirect (*http*), 96
postpone_output (*http*), 96
preread_buffer_size (*stream*), 510
preread_timeout (*stream*), 510
prometheus (*http*), 265
prometheus_template (*http*), 265
protocol (*mail-ADC*), 524
proxy_bind (*http*), 268
proxy_bind (*stream*), 452
proxy_buffer, 532
proxy_buffer_size (*http*), 269
proxy_buffer_size (*stream*), 452
proxy_buffering (*http*), 269
proxy_buffers (*http*), 270
proxy_busy_buffers_size (*http*), 270
proxy_cache (*http*), 270
proxy_cache_background_update (*http*), 271
proxy_cache_bypass (*http*), 271
proxy_cache_convert_head (*http*), 272
proxy_cache_key (*http*), 272
proxy_cache_lock (*http*), 272

proxy_cache_lock_age (*http*), 272
 proxy_cache_lock_timeout (*http*), 273
 proxy_cache_max_range_offset (*http*), 273
 proxy_cache_methods (*http*), 273
 proxy_cache_min_uses (*http*), 273
 proxy_cache_path (*http*), 274
 proxy_cache_revalidate (*http*), 276
 proxy_cache_use_stale (*http*), 276
 proxy_cache_valid (*http*), 277
 proxy_connect_timeout (*http*), 278
 proxy_connect_timeout (*stream*), 452
 proxy_connection_drop (*http*), 278
 proxy_connection_drop (*stream*), 453
 proxy_cookie_domain (*http*), 279
 proxy_cookie_flags (*http*), 279
 proxy_cookie_path (*http*), 280
 proxy_download_rate (*stream*), 453
 proxy_force_ranges (*http*), 280
 proxy_half_close (*stream*), 453
 proxy_headers_hash_bucket_size (*http*), 281
 proxy_headers_hash_max_size (*http*), 281
 proxy_hide_header (*http*), 281
 proxy_http3_hq (*http*), 281
 proxy_http3_max_concurrent_streams (*http*), 282
 proxy_http3_max_table_capacity (*http*), 282
 proxy_http3_stream_buffer_size (*http*), 282
 proxy_http_version (*http*), 282
 proxy_ignore_client_abort (*http*), 282
 proxy_ignore_headers (*http*), 283
 proxy_intercept_errors (*http*), 283
 proxy_interface (*http*), 283
 proxy_interface (*stream*), 454
 proxy_limit_rate (*http*), 283
 proxy_max_temp_file_size (*http*), 284
 proxy_method (*http*), 284
 proxy_next_upstream (*http*), 284
 proxy_next_upstream (*stream*), 454
 proxy_next_upstream_timeout (*http*), 285
 proxy_next_upstream_timeout (*stream*), 454
 proxy_next_upstream_tries (*http*), 286
 proxy_next_upstream_tries (*stream*), 454
 proxy_no_cache (*http*), 286
 proxy_pass (*http*), 286
 proxy_pass (*stream*), 455
 proxy_pass_error_message, 532
 proxy_pass_header (*http*), 288
 proxy_pass_request_body (*http*), 288
 proxy_pass_request_headers (*http*), 288
 proxy_pass_trailers (*http*), 289
 proxy_protocol, 532
 proxy_protocol (*stream*), 455
 proxy_protocol_timeout (*stream*), 510
 proxy_protocol_tlv (*stream*), 455
 proxy_protocol_version (*stream*), 456
 proxy_quic_active_connection_id_limit (*http*), 289
 proxy_quic_gso (*http*), 289
 proxy_quic_host_key (*http*), 290
 proxy_read_timeout (*http*), 290
 proxy_redirect (*http*), 290
 proxy_request_buffering (*http*), 291
 proxy_requests (*stream*), 456
 proxy_responses (*stream*), 456
 proxy_send_lowat (*http*), 292
 proxy_send_timeout (*http*), 292
 proxy_set_body (*http*), 292
 proxy_set_header (*http*), 293
 proxy_smtp_auth, 532
 proxy_socket_keepalive (*http*), 293
 proxy_socket_keepalive (*stream*), 456
 proxy_ssl (*stream*), 457
 proxy_ssl_alpn (*stream*), 457
 proxy_ssl_certificate (*http*), 294
 proxy_ssl_certificate (*stream*), 457
 proxy_ssl_certificate_cache (*http*), 294
 proxy_ssl_certificate_cache (*stream*), 457
 proxy_ssl_certificate_key (*http*), 295
 proxy_ssl_certificate_key (*stream*), 458
 proxy_ssl_ciphers (*http*), 295
 proxy_ssl_ciphers (*stream*), 458
 proxy_ssl_conf_command (*http*), 296
 proxy_ssl_conf_command (*stream*), 459
 proxy_ssl_crl (*http*), 296
 proxy_ssl_crl (*stream*), 459
 proxy_ssl_keylog_file (*http*), 296
 proxy_ssl_keylog_file (*stream*), 460
 proxy_ssl_name (*http*), 297
 proxy_ssl_name (*stream*), 460
 proxy_ssl_ntls (*http*), 297
 proxy_ssl_ntls (*stream*), 460
 proxy_ssl_password_file (*http*), 297
 proxy_ssl_password_file (*stream*), 461
 proxy_ssl_protocols (*http*), 297
 proxy_ssl_protocols (*stream*), 461
 proxy_ssl_server_name (*http*), 298
 proxy_ssl_server_name (*stream*), 461
 proxy_ssl_session_reuse (*http*), 298
 proxy_ssl_session_reuse (*stream*), 461
 proxy_ssl_trusted_certificate (*http*), 298
 proxy_ssl_trusted_certificate (*stream*), 461
 proxy_ssl_verify (*http*), 298
 proxy_ssl_verify (*stream*), 462
 proxy_ssl_verify_depth (*http*), 298
 proxy_ssl_verify_depth (*stream*), 462
 proxy_store (*http*), 299
 proxy_store_access (*http*), 300
 proxy_tcp_congestion, 533
 proxy_tcp_congestion (*http*), 300
 proxy_tcp_congestion (*stream*), 462
 proxy_temp_file_write_size (*http*), 300
 proxy_temp_path (*http*), 301
 proxy_timeout, 533
 proxy_timeout (*stream*), 462
 proxy_upload_rate (*stream*), 462

Q

queue (*http*), 369
quic_active_connection_id_limit (*http*), 418
quic_bpf (*http*), 418
quic_gso (*http*), 418
quic_host_key (*http*), 419
quic_retry (*http*), 419

R

random (*http*), 370
random (*stream*), 490
random_index (*http*), 302
rdp_preread (*stream*), 464
read_ahead (*http*), 96
real_ip_header (*http*), 302
real_ip_recursive (*http*), 302
recursive_error_pages (*http*), 97
referrer_hash_bucket_size (*http*), 303
referrer_hash_max_size (*http*), 303
request_pool_size (*http*), 97
reset_timedout_connection (*http*), 97
resolver (*http*), 97
resolver (*mail-ADC*), 524
resolver (*stream*), 511
resolver_timeout (*http*), 99
resolver_timeout (*mail-ADC*), 525
resolver_timeout (*stream*), 512
response_time_factor (*http*), 371
response_time_factor (*stream*), 490
return (*http*), 307
return (*stream*), 465
rewrite (*http*), 307
rewrite_log (*http*), 308
root (*http*), 99

S

satisfy (*http*), 99
scgi_bind (*http*), 310
scgi_buffer_size (*http*), 310
scgi_buffering (*http*), 311
scgi_buffers (*http*), 311
scgi_busy_buffers_size (*http*), 311
scgi_cache (*http*), 312
scgi_cache_background_update (*http*), 312
scgi_cache_bypass (*http*), 313
scgi_cache_key (*http*), 313
scgi_cache_lock (*http*), 313
scgi_cache_lock_age (*http*), 313
scgi_cache_lock_timeout (*http*), 314
scgi_cache_max_range_offset (*http*), 314
scgi_cache_methods (*http*), 314
scgi_cache_min_uses (*http*), 314
scgi_cache_path (*http*), 315
scgi_cache_revalidate (*http*), 317
scgi_cache_use_stale (*http*), 317
scgi_cache_valid (*http*), 318
scgi_connect_timeout (*http*), 319
scgi_connection_drop (*http*), 319

scgi_force_ranges (*http*), 319
scgi_hide_header (*http*), 320
scgi_ignore_client_abort (*http*), 320
scgi_ignore_headers (*http*), 320
scgi_intercept_errors (*http*), 320
scgi_interface (*http*), 321
scgi_limit_rate (*http*), 321
scgi_max_temp_file_size (*http*), 321
scgi_next_upstream (*http*), 321
scgi_next_upstream_timeout (*http*), 322
scgi_next_upstream_tries (*http*), 323
scgi_no_cache (*http*), 323
scgi_param (*http*), 323
scgi_pass (*http*), 324
scgi_pass_header (*http*), 324
scgi_pass_request_body (*http*), 324
scgi_pass_request_headers (*http*), 325
scgi_read_timeout (*http*), 325
scgi_request_buffering (*http*), 325
scgi_send_timeout (*http*), 325
scgi_socket_keepalive (*http*), 325
scgi_store (*http*), 326
scgi_store_access (*http*), 327
scgi_tcp_congestion (*http*), 327
scgi_temp_file_write_size (*http*), 327
scgi_temp_path (*http*), 327
secure_link (*http*), 328
secure_link_md5 (*http*), 328
secure_link_secret (*http*), 329
send_lowat (*http*), 100
send_timeout (*http*), 100
sendfile (*http*), 100
sendfile_max_chunk (*http*), 101
server (*http*), 101, 371
server (*mail-ADC*), 526
server (*stream*), 491, 512
server_name (*http*), 101
server_name (*mail-ADC*), 526
server_name (*stream*), 513
server_name_in_redirect (*http*), 103
server_names_hash_bucket_size (*http*), 103
server_names_hash_bucket_size (*stream*), 514
server_names_hash_max_size (*http*), 103
server_names_hash_max_size (*stream*), 514
server_tokens (*http*), 104
set (*http*), 308
set (*stream*), 465
set_real_ip_from, 534
set_real_ip_from (*http*), 302
set_real_ip_from (*stream*), 464
slice (*http*), 331
smtp_auth, 534
smtp_capabilities, 535
smtp_client_buffer, 535
smtp_greeting_delay, 535
source_charset (*http*), 170
split_clients (*http*), 331
split_clients (*stream*), 466

ssi (*http*), 332
 ssi_last_modified (*http*), 332
 ssi_min_file_chunk (*http*), 333
 ssi_silent_errors (*http*), 333
 ssi_types (*http*), 333
 ssi_value_length (*http*), 333
 ssl_alpn (*stream*), 467
 ssl_buffer_size (*http*), 337
 ssl_certificate, 536
 ssl_certificate (*http*), 338
 ssl_certificate (*stream*), 468
 ssl_certificate_cache (*http*), 339
 ssl_certificate_cache (*stream*), 468
 ssl_certificate_compression, 537
 ssl_certificate_compression (*http*), 339
 ssl_certificate_compression (*stream*), 469
 ssl_certificate_key, 537
 ssl_certificate_key (*http*), 340
 ssl_certificate_key (*stream*), 469
 ssl_ciphers, 538
 ssl_ciphers (*http*), 341
 ssl_ciphers (*stream*), 470
 ssl_client_certificate, 538
 ssl_client_certificate (*http*), 341
 ssl_client_certificate (*stream*), 470
 ssl_conf_command, 538
 ssl_conf_command (*http*), 341
 ssl_conf_command (*stream*), 471
 ssl_crl, 539
 ssl_crl (*http*), 342
 ssl_crl (*stream*), 471
 ssl_dhparam, 539
 ssl_dhparam (*http*), 342
 ssl_dhparam (*stream*), 471
 ssl_early_data (*http*), 342
 ssl_early_data (*stream*), 472
 ssl_ecdh_curve, 539
 ssl_ecdh_curve (*http*), 343
 ssl_ecdh_curve (*stream*), 472
 ssl_encrypted_hello_key (*http*), 343
 ssl_encrypted_hello_key (*stream*), 472
 ssl_engine (*core-ADC*), 64
 ssl_handshake_timeout (*stream*), 473
 ssl_keylog_file (*http*), 344
 ssl_keylog_file (*stream*), 473
 ssl_ntls (*http*), 344
 ssl_ntls (*stream*), 474
 ssl_object_cache_inheritable (*core-ADC*), 65
 ssl_ocsp (*http*), 345
 ssl_ocsp (*stream*), 474
 ssl_ocsp_cache (*http*), 345
 ssl_ocsp_cache (*stream*), 474
 ssl_ocsp_responder (*http*), 345
 ssl_ocsp_responder (*stream*), 475
 ssl_password_file, 540
 ssl_password_file (*http*), 345
 ssl_password_file (*stream*), 475
 ssl_prefer_server_ciphers, 540
 ssl_prefer_server_ciphers (*http*), 346
 ssl_prefer_server_ciphers (*stream*), 476
 ssl_preread (*stream*), 484
 ssl_protocols, 541
 ssl_protocols (*http*), 346
 ssl_protocols (*stream*), 476
 ssl_reject_handshake (*http*), 346
 ssl_session_cache, 541
 ssl_session_cache (*http*), 347
 ssl_session_cache (*stream*), 476
 ssl_session_ticket_key, 541
 ssl_session_ticket_key (*http*), 348
 ssl_session_ticket_key (*stream*), 477
 ssl_session_tickets, 542
 ssl_session_tickets (*http*), 348
 ssl_session_tickets (*stream*), 477
 ssl_session_timeout, 542
 ssl_session_timeout (*http*), 348
 ssl_session_timeout (*stream*), 478
 ssl_stapling (*http*), 348
 ssl_stapling (*stream*), 478
 ssl_stapling_file (*http*), 349
 ssl_stapling_file (*stream*), 478
 ssl_stapling_responder (*http*), 349
 ssl_stapling_responder (*stream*), 478
 ssl_stapling_verify (*http*), 349
 ssl_stapling_verify (*stream*), 479
 ssl_trusted_certificate, 542
 ssl_trusted_certificate (*http*), 349
 ssl_trusted_certificate (*stream*), 479
 ssl_verify_client, 542
 ssl_verify_client (*http*), 350
 ssl_verify_client (*stream*), 479
 ssl_verify_depth, 543
 ssl_verify_depth (*http*), 350
 ssl_verify_depth (*stream*), 479
 starttls, 543
 state (*http*), 374
 state (*stream*), 494
 status_zone (*http*), 104
 status_zone (*stream*), 514
 sticky (*http*), 375
 sticky (*stream*), 495
 sticky_secret (*http*), 380
 sticky_secret (*stream*), 499
 sticky_strict (*http*), 381
 sticky_strict (*stream*), 500
 stream (*stream*), 515
 stub_status (*http*), 354
 sub_filter (*http*), 355
 sub_filter_last_modified (*http*), 356
 sub_filter_once (*http*), 356
 sub_filter_types (*http*), 356
 subrequest_output_buffer_size (*http*), 105

T

tcp_nodelay (*http*), 105
 tcp_nodelay (*stream*), 516

tcp_nopush (*http*), 106
thread_pool (*core-ADC*), 65
time_format (*http*), 106
time_format (*stream*), 516
timeout (*mail-ADC*), 526
timer_resolution (*core-ADC*), 65
try_files (*http*), 107
tunnel_bind (*http*), 357
tunnel_buffer_size (*http*), 357
tunnel_connect_timeout (*http*), 357
tunnel_next_upstream (*http*), 358
tunnel_next_upstream_timeout (*http*), 358
tunnel_next_upstream_tries (*http*), 358
tunnel_pass (*http*), 358
tunnel_read_timeout (*http*), 359
tunnel_send_lowat (*http*), 359
tunnel_send_timeout (*http*), 359
tunnel_socket_keepalive (*http*), 360
types (*http*), 109
types_hash_bucket_size (*http*), 110
types_hash_max_size (*http*), 110

U

underscores_in_headers (*http*), 110
uninitialized_variable_warn (*http*), 308
upstream (*http*), 381
upstream (*stream*), 500
upstream_probe (*http*), 385
upstream_probe (*stream*), 503
upstream_probe_timeout (*stream*), 505
use (*core-ADC*), 66
user (*core-ADC*), 66
userid (*http*), 388
userid_domain (*http*), 388
userid_expires (*http*), 388
userid_flags (*http*), 388
userid_mark (*http*), 388
userid_name (*http*), 389
userid_p3p (*http*), 389
userid_path (*http*), 389
userid_service (*http*), 389
uwsgi_bind (*http*), 390
uwsgi_buffer_size (*http*), 391
uwsgi_buffering (*http*), 391
uwsgi_buffers (*http*), 392
uwsgi_busy_buffers_size (*http*), 392
uwsgi_cache (*http*), 393
uwsgi_cache_background_update (*http*), 393
uwsgi_cache_bypass (*http*), 393
uwsgi_cache_key (*http*), 394
uwsgi_cache_lock (*http*), 394
uwsgi_cache_lock_age (*http*), 394
uwsgi_cache_lock_timeout (*http*), 394
uwsgi_cache_max_range_offset (*http*), 395
uwsgi_cache_methods (*http*), 395
uwsgi_cache_min_uses (*http*), 395
uwsgi_cache_path (*http*), 396
uwsgi_cache_revalidate (*http*), 398

uwsgi_cache_use_stale (*http*), 398
uwsgi_cache_valid (*http*), 399
uwsgi_connect_timeout (*http*), 400
uwsgi_connection_drop (*http*), 400
uwsgi_force_ranges (*http*), 400
uwsgi_hide_header (*http*), 401
uwsgi_ignore_client_abort (*http*), 401
uwsgi_ignore_headers (*http*), 401
uwsgi_intercept_errors (*http*), 401
uwsgi_interface (*http*), 402
uwsgi_limit_rate (*http*), 402
uwsgi_max_temp_file_size (*http*), 402
uwsgi_modifier1 (*http*), 402
uwsgi_modifier2 (*http*), 403
uwsgi_next_upstream (*http*), 403
uwsgi_next_upstream_timeout (*http*), 404
uwsgi_next_upstream_tries (*http*), 404
uwsgi_no_cache (*http*), 404
uwsgi_param (*http*), 404
uwsgi_pass (*http*), 405
uwsgi_pass_header (*http*), 406
uwsgi_pass_request_body (*http*), 406
uwsgi_pass_request_headers (*http*), 406
uwsgi_read_timeout (*http*), 406
uwsgi_request_buffering (*http*), 406
uwsgi_send_timeout (*http*), 407
uwsgi_socket_keepalive (*http*), 407
uwsgi_ssl_certificate (*http*), 407
uwsgi_ssl_certificate_cache (*http*), 407
uwsgi_ssl_certificate_key (*http*), 408
uwsgi_ssl_ciphers (*http*), 408
uwsgi_ssl_conf_command (*http*), 409
uwsgi_ssl_crl (*http*), 409
uwsgi_ssl_name (*http*), 409
uwsgi_ssl_password_file (*http*), 409
uwsgi_ssl_protocols (*http*), 410
uwsgi_ssl_server_name (*http*), 410
uwsgi_ssl_session_reuse (*http*), 410
uwsgi_ssl_trusted_certificate (*http*), 410
uwsgi_ssl_verify (*http*), 410
uwsgi_ssl_verify_depth (*http*), 411
uwsgi_store (*http*), 411
uwsgi_store_access (*http*), 412
uwsgi_tcp_congestion (*http*), 412
uwsgi_temp_file_write_size (*http*), 412
uwsgi_temp_path (*http*), 412

V

valid_referers (*http*), 304
variables_hash_bucket_size (*http*), 111
variables_hash_bucket_size (*stream*), 517
variables_hash_max_size (*http*), 111
variables_hash_max_size (*stream*), 517

W

worker_aio_requests (*core-ADC*), 66
worker_connections (*core-ADC*), 66
worker_cpu_affinity (*core-ADC*), 67

`worker_priority` (*core-ADC*), 67
`worker_processes` (*core-ADC*), 68
`worker_rlimit_core` (*core-ADC*), 68
`worker_rlimit_nofile` (*core-ADC*), 68
`worker_shutdown_timeout` (*core-ADC*), 68
`working_directory` (*core-ADC*), 69

X

`xclient`, 533

Z

`zone` (*http*), 382
`zone` (*stream*), 500