



Описание функциональных  
характеристик  
*версия 1.8.2*

ООО «Веб-Сервер»

февр. 13, 2025

---

## Оглавление

---

<b>1</b>	<b>Аннотация</b>	<b>1</b>
<b>2</b>	<b>Общие сведения</b>	<b>2</b>
<b>3</b>	<b>Функциональные характеристики</b>	<b>4</b>
3.1	Управление во время выполнения	4
3.1.1	Использование сигналов	5
3.1.2	Изменение конфигурации	5
3.1.3	Ротация лог-файлов	6
3.2	Соединения, сессии, запросы, логи	7
3.2.1	Механизмы обработки соединений	7
3.2.2	Обработка HTTP-запросов	8
3.2.3	Обработка TCP/UDP-сессий	8
3.2.4	Обработка запросов	9
3.2.5	Проксирование и балансировка нагрузки	13
3.2.6	Логирование	16
3.3	Встроенные модули	16
3.3.1	Основной модуль	17
3.3.2	HTTP-модуль	27
3.3.3	Потоковый модуль	359
3.3.4	Почтовый модуль	439
3.3.5	Модуль Google PerfTools	461
3.3.6	Модуль WASM	462
3.3.7	Основной модуль	466
3.3.8	HTTP-модули	466
3.3.9	Потоковые модули	467
3.3.10	Почтовые модули	468
3.3.11	Модуль Google PerfTools	468
3.3.12	Модули WASM	468
<b>4</b>	<b>Права на интеллектуальную собственность</b>	<b>469</b>
	<b>Алфавитный указатель</b>	<b>470</b>

# ГЛАВА 1

---

## Аннотация

---

Настоящий документ содержит описание функциональных характеристик Angie PRO. Angie PRO — эффективный, мощный и масштабируемый веб-сервер.

## ГЛАВА 2

---

### Общие сведения

---

Angie PRO – единственный коммерческий веб-сервер, разработка которого локализована в России. Веб-сервер — это класс программного обеспечения, предоставляющего доступ к сетевым ресурсам по протоколу HTTP конечным пользователям. Angie PRO, например, может использоваться для работы интернет-сайтов, мобильных приложений, киосков самообслуживания в метрополитене, мультимедийных систем в поездах дальнего следования. Каждый раз, когда пользователь открывает сайт, мобильное приложение, пользуется киоском самообслуживания в метрополитене, или даже мультимедийной системой в поезде «Сапсан», запрос пользователя может быть обработан Angie PRO.

Angie PRO — это:

- **Веб-сервер общего назначения.** Написан на языке C.
- **Балансировщик L4-L7.** Позволяет балансировать нагрузку между серверами как по протоколам TCP/UDP, так и по HTTP.
- **Проксирующий и кэширующий сервер.** Позволяет ускорить работу веб-сервисов с помощью гибкого механизма кеширования.
- **Доступен под все популярные платформы.** Собирается и тестируется под Alpine, Debian, Oracle, RED OS, Rocky, Ubuntu.
- **Производительность.** Один из самых производительных веб-серверов в мире.

Почему Angie PRO:

- **Совместимость с NGINX OSS.** Angie PRO полностью совместим с Nginx, таким образом любой существующий пользователь Nginx может без серьезных затрат и простоя сервисов перейти на Angie PRO.
- **Расширенная статистика и мониторинг в реальном времени.** Angie PRO имеет возможность полного мониторинга нагрузки сервера в режиме реального времени, что позволяет динамически управлять конфигурациями по профилю нагрузки и соблюдать полную доступность сервиса.
- **Динамическая конфигурация групп проксируемых серверов.** Возможность управлять настройками групп проксируемых серверов с помощью удобного REST интерфейса без остановки сервиса.
- **Удаление элементов кэша.** Возможность удаления элементов кеша через удобное API без остановки сервиса.

- **Активная проверка состояния проксируемых серверов.** Проверка на "живучесть" и проксирование только на те группы проксируемых серверов, которые отвечают по заданному алгоритму.
- **Динамическое хранилище ключ-значение».** Динамическое управление переменными конфигурации Angie PRO через HTTP API.
- **Динамическое обновление DNS.**
- **Проксирование с привязкой сессий.**
- **Репозиторий с динамическими сторонними модулями.** Angie PRO поддерживает большинство сторонних модулей NGINX и дает возможность без проблем устанавливать их, предоставляя гарантию их работоспособности и поддержку.
- **Синхронизация зон разделяемой памяти.** Возможность использовать зоны кеша, limit\_req и т.д. в кластере Angie PRO.
- **Соккрытие или персональный брендинг имени сервера в заголовках ответа.** Возможность изменить или скрыть название и версию веб-сервера от пользователей.

Перечень иностранного программного обеспечения, имеющего сходство функциональных характеристик с программным обеспечением Angie PRO: Nginx, Nginx Plus, Apache, Envoy, продукты, использующие решения NGINX (OpenResty, Tengine, Cloudflare), облачные решения Яндекса.

---

## Функциональные характеристики

---

### 3.1 Управление во время выполнения

Чтобы запустить Angie, используйте `systemd` и следующую команду:

```
$ sudo service angie start
```

Рекомендуется предварительно проверить синтаксис конфигурации. Вот как это сделать:

```
$ sudo angie -t && sudo service angie start
```

Чтобы перезагрузить конфигурацию:

```
$ sudo angie -t && sudo service angie reload
```

Чтобы остановить Angie:

```
$ sudo service angie stop
```

После установки выполните следующую команду, чтобы убедиться, что Angie работает:

```
$ curl localhost:80
```

#### Примечание

Методы запуска открытой версии Angie могут различаться в зависимости от метода установки.

Angie имеет один главный процесс и несколько рабочих процессов. Главный процесс отвечает за чтение и оценку конфигурации и управление рабочими процессами. Рабочие процессы обрабатывают фактические запросы. Angie использует модель на основе событий и механизмы, зависящие от ОС, для эффективного распределения запросов между рабочими процессами. Количество рабочих процессов определяется в конфигурационном файле и может быть фиксированным для данной конфигурации или автоматически регулироваться в зависимости от числа доступных ядер CPU (см. *worker\_processes*).

### 3.1.1 Использование сигналов

Angie также можно управлять с помощью сигналов. По умолчанию идентификатор процесса главного процесса записывается в файл `/var/run/angie.pid`. Это имя файла можно изменить во время конфигурации или в файле `angie.conf` с помощью директивы `pid`. Главный процесс поддерживает следующие сигналы:

TERM, INT	Быстрое завершение работы
QUIT	<i>Постепенное завершение</i> работы
HUP	Перезагрузка конфигурации, обновление часового пояса (только для FreeBSD и Linux), запуск новых рабочих процессов с обновленной конфигурацией, <i>постепенное завершение</i> старых рабочих процессов
USR1	Перезапуск файлов журналов
USR2	Обновление исполняемого файла
WINCH	<i>Постепенное завершение</i> рабочих процессов

Отдельные рабочие процессы также можно контролировать с помощью сигналов, хотя это не обязательно. Поддерживаются следующие сигналы:

TERM, INT	Быстрое завершение работы
QUIT	<i>Постепенное завершение</i> работы
USR1	Перезапуск файлов журналов
WINCH	Аномальное завершение работы для отладки (требуется включение <code>debug_points</code> )

### 3.1.2 Изменение конфигурации

Чтобы Angie перечитал конфигурационный файл, необходимо отправить сигнал HUP главному процессу. Сначала главный процесс проверяет корректность синтаксиса, а затем пытается применить новую конфигурацию, что включает в себя открытие новых файлов журналов и сокетов для прослушивания. Если применение новой конфигурации не удалось, главный процесс откатывает изменения и продолжает работу со старой конфигурацией. Если применение прошло успешно, главный процесс запускает новые рабочие процессы и отправляет сообщения старым рабочим процессам с просьбой завершиться *постепенно*. Старые рабочие процессы закрывают свои сокет для прослушивания и продолжают обслуживать существующих клиентов. После того как все клиенты будут обслужены, старые рабочие процессы завершатся.

Angie отслеживает изменения конфигурации для каждого процесса. Номера поколений начинаются с 1, когда сервер запускается впервые. Эти номера увеличиваются с каждой перезагрузкой конфигурации и видны в заголовках процессов:

```
$ sudo angie
$ ps aux | grep angie

angie: master process v1.8.2 #1 [angie]
angie: worker process #1
```

После успешной перезагрузки конфигурации (независимо от того, были ли фактические изменения), Angie увеличивает номер поколения для процессов, которые получили новую конфигурацию:

```
$ sudo angie -s reload
$ ps aux | grep angie

angie: master process v1.8.2 #2 [angie]
angie: worker process #2
```

Если какие-либо рабочие процессы из предыдущих поколений продолжают работать, они сразу же станут заметны:

```
$ ps aux | grep angie
```

```
angie: worker process #1
angie: worker process #2
```

### Примечание

Не путайте номер поколения конфигурации с неким 'номером процесса'; Angie не использует непрерывную нумерацию процессов для практических целей.

### 3.1.3 Ротация лог-файлов

Для ротации лог-файлов сначала переименуйте файлы. Затем отправьте сигнал `USR1` главному процессу. Главный процесс снова откроет все открытые лог-файлы и назначит их непривилегированному пользователю, под которым работают рабочие процессы. После успешного открытия файлов главный процесс закроет все открытые файлы и уведомит рабочие процессы о необходимости открыть свои лог-файлы. Рабочие процессы также немедленно откроют новые файлы и закроют старые. В результате старые файлы становятся доступными для последующей обработки, такой как сжатие, почти сразу.

#### Обновление исполняемого файла на лету

Чтобы обновить исполняемый файл сервера, сначала замените старый исполняемый файл на новый. Затем отправьте сигнал `USR2` главному процессу. Главный процесс переименует свой текущий файл с идентификатором процесса в новый файл с суффиксом `.oldbin`, например, `/usr/local/angie/logs/angie.pid.oldbin`, и затем запустит новый исполняемый файл, который, в свою очередь, запустит новые рабочие процессы.

Обратите внимание, что старый главный процесс не закрывает свои сокеты для прослушивания и может быть управляет для перезапуска своих рабочих процессов в случае необходимости. Если новый исполняемый файл работает не так, как ожидалось, вы можете предпринять следующие действия:

- Отправьте сигнал `HUP` старому главному процессу. Это запустит новые рабочие процессы без повторного чтения конфигурации. После этого вы можете завершить все новые процессы *постепенно*, отправив сигнал `QUIT` новому главному процессу.
- Отправьте сигнал `TERM` новому главному процессу. Он отправит сообщение своим рабочим процессам с просьбой немедленно выйти. Если какие-либо процессы не выходят, отправьте сигнал `KILL`, чтобы заставить их выйти. Когда новый главный процесс завершится, старый главный процесс автоматически запустит новые рабочие процессы.

Если новый главный процесс завершится, старый главный процесс удалит суффикс `.oldbin` из имени файла с идентификатором процесса.

Если обновление прошло успешно, отправьте сигнал `QUIT` старому главному процессу, и останутся только новые процессы.



## Параметры командной строки

-?, -h	Вывод справки по параметрам командной строки, затем выход.
--build-env	Вывод вспомогательной информации об окружении сборки, затем выход.
-c <i>файл</i>	Запуск с альтернативным <i>файлом</i> конфигурации вместо файла по умолчанию.
-e <i>файл</i>	Запуск с альтернативным лог- <i>файлом</i> ошибок вместо <i>файла по умолчанию</i> . Специальное значение <code>stderr</code> задает стандартный файл ошибок.
-g <i>директивы</i>	Запуск с установкой <i>глобальных директив конфигурации</i> , например: <code>angie -g "pid /var/run/angie.pid; worker_processes `sysctl -n hw.ncpu`"</code> .
-m, -M	Вывод списка встроенных (-m) или встроенных и загруженных (-M) модулей, затем выход.
-p <i>префикс</i>	Запуск с заданным <i>префиксом</i> пути <code>angie</code> (каталога, в котором будут находиться файлы сервера; по умолчанию — <code>/usr/local/angie/</code> ).
-q	Вывод только сообщений об ошибках, если заданы -t или -T; иначе эффекта нет.
-s <i>сигнал</i>	Отправка <i>сигнала</i> главному процессу: <code>stop</code> , <code>quit</code> , <code>reopen</code> , <code>reload</code> и так далее.
-t	Тестирование файла конфигурации, затем выход. Angie проверяет синтаксис конфигурации, рекурсивно включая файлы, упомянутые в ней.
-T	То же, что -t, но с выводом сводной конфигурации в стандартный поток вывода после рекурсивного включения всех упомянутых в ней файлов.
-v	Вывод версии Angie, затем выход.
-V	Вывод версии Angie, версии компилятора и использованных параметров сборки, затем выход.

## 3.2 Соединения, сессии, запросы, логи

### 3.2.1 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. Опции сборки <code>--with-select_module</code> и <code>--without-select_module</code> могут быть использованы для принудительного включения или отключения сборки этого модуля.
<code>poll</code>	Стандартный метод. Соответствующий модуль собирается автоматически на платформах, не имеющих более эффективных методов. Опции сборки <code>--with-poll_module</code> и <code>--without-poll_module</code> могут быть использованы для принудительного включения или отключения сборки этого модуля.
<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> .)

### 3.2.2 Обработка HTTP-запросов

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

Post-read	Начальная фаза. Модуль <i>RealIP</i> вызывается на этой фазе.
Server-rewrite	Фаза, на которой обрабатываются директивы из модуля <i>Rewrite</i> , определенные в блоке <i>server</i> (но вне блока <i>location</i> ).
Find-config	Специальная фаза, на которой выбирается <i>location</i> на основе URI запроса.
Rewrite	Похожа на фазу <i>Server-rewrite</i> , но применяется к правилам <i>rewrite</i> , определенным в блоке <i>location</i> , выбранном на предыдущей фазе.
Post-rewrite	Специальная фаза, на которой запрос перенаправляется в новое место, как на фазе <i>Find-config</i> , если его URI был изменен во время фазы <i>Rewrite</i> .
Preaccess	На этой фазе стандартные модули <i>Angie</i> , такие как <i>Limit Req</i> , регистрируют свои обработчики.
Access	Фаза, на которой проверяется право клиента на выполнение запроса, обычно с помощью стандартных модулей <i>Angie</i> , таких как <i>Auth Basic</i> .
Post-access	Специальная фаза, на которой обрабатывается директива <i>satisfy any</i> .
Precontent	На этой фазе стандартные модули, такие как директивы <i>try_files</i> и <i>mirror</i> , регистрируют свои обработчики.
Content	Фаза, на которой обычно генерируется ответ. На этом этапе несколько стандартных модулей <i>Angie</i> 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.
Log	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.

### 3.2.3 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> .

## 3.2.4 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 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 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 направит запрос к серверу по умолчанию для этого порта. В приведенной выше конфигурации сервером по умолчанию является первый — что является стандартным поведением Angie. 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 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 локацию, как если бы запрос был отправлен клиентом. Как упоминалось ранее, переадресованный запрос в конечном итоге будет обработан сервером FastCGI.

### 3.2.5 Проксирование и балансировка нагрузки

Одним из распространенных способов использования Angie является настройка в качестве прокси-сервера. В этой роли Angie 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 передавать запросы клиентов на сервер `backend:8080` (прокси-сервер). Существует множество дополнительных *директив*, которые можно использовать для дальнейшей настройки прокси-соединения.

#### Проксирование FastCGI

Angie 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 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 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 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 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`). Другие поддерживаемые механизмы балансировки нагрузки включают: `weight`, `least_conn` и `ip_hash`. 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`.

### 3.2.6 Логирование

#### Примечание

В дополнение к перечисленным здесь опциям, вы также можете включить отладочный лог.

#### 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=unix:/var/log/angie.sock,nohostname;
access_log syslog:server=[2001:db8::1]:12345,facility=local7,tag=angie,severity=info
↪combined;
```

#### Примечание

Сообщения в `syslog` записываются не чаще раза в секунду, чтобы предотвратить переполнение.

## 3.3 Встроенные модули

В этом справочнике описаны собственные модули Angie, даны примеры конфигурации, перечислены их директивы и параметры, а также встроенные переменные.

### 3.3.1 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

Синтаксис	accept_mutex on   off;
По умолчанию	accept_mutex off;
Контекст	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

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

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

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

## debug\_connection

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

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::;
    # ...
}
```

### Важно

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

## debug\_points

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

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

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

- наследуются во время *обновления исполняемого файла на лету*;
- используются модулем *Perl*;
- 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. Переменная `TZ` всегда наследуется и доступна модулю *Perl*, если не задано явно иное поведение.

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

Синтаксис	<code>error_log файл [уровень];</code>
По умолчанию	<code>error_log logs/error.log error;</code> (путь зависит от параметра сборки <code>--error-log-path</code> )
Контекст	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**.

### Важно

Чтобы работал уровень логирования **debug**, в сборке Angie должен быть включен отладочный лог.

## events

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

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

## include

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

## load\_module

Синтаксис	<code>load_module файл;</code>
По умолчанию	—
Контекст	main

Загружает динамический модуль из указанного файла. Относительные пути задаются от параметра сборки `--prefix`, уточнить который можно так:

```
$ sudo angie -V
```

Пример:

```
load_module modules/nginx_mail_module.so;
```

## lock\_file

Синтаксис	<code>lock_file файл;</code>
По умолчанию	<code>lock_file logs/angie.lock;</code> (путь зависит от параметра сборки <code>--lock-path</code> )
Контекст	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

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

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

## multi\_accept

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

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

### **i** Примечание

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

## pcre\_jit

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

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

Использование PCRE JIT заметно ускоряет обработку регулярных выражений.

### **ii** Важно

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

Синтаксис	<code>pid файл   off;</code>
По умолчанию	<code>pid logs/angie.pid;</code> (путь зависит от параметра сборки <code>--pid-path</code> )
Контекст	<code>main</code>

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

### **i** Примечание

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



## ssl\_engine

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

Задаёт название аппаратного SSL-акселератора.

## thread\_pool

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

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

Уменьшает разрешение таймеров времени в рабочих процессах, за счёт 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

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

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

## user

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

Задаёт пользователя и группу для рабочих процессов (см. также параметры сборки). Если задан только пользователь, для группы также задается указанное имя пользователя.

## worker\_aio\_requests

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

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

## worker\_connections

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

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

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;
```

### ❗ Важно

Директива доступна только на FreeBSD и Linux.

## worker\_priority

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

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

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

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

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

### worker\_rlimit\_nofile

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

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

### worker\_shutdown\_timeout

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

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

Синтаксис	<code>working_directory каталог;</code>
По умолчанию	—
Контекст	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.

## 3.3.2 HTTP-модуль

### 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	<code>http, server, location, limit_except</code>

Разрешает доступ для указанной сети или адреса. Специальное значение `all` означает все IP-адреса клиентов.

Добавлено в версии 1.5.1: Специальное значение `unix:` разрешает доступ для любых UNIX-сокетов.

### deny

Синтаксис	<code>deny адрес   CIDR   unix:   all;</code>
По умолчанию	—
Контекст	<code>http, server, location, limit_except</code>

Запрещает доступ для указанной сети или адреса. Специальное значение `all` означает все IP-адреса клиентов.

Добавлено в версии 1.5.1: Специальное значение `unix:` запрещает доступ для любых UNIX-сокетов.

## ACME

Обеспечивает автоматическое получение сертификатов с использованием протокола ACME.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_acme_module`. В пакетах и образах из наших репозиториев модуль включен в сборку.

## Пример конфигурации

Примеры конфигурации и инструкции по настройке см. в разделе `acme_config`.

## Директивы

### acme

Синтаксис	<code>acme имя;</code>
По умолчанию	—
Контекст	<code>server</code>

Для всех доменов, указанных в директивах `server_name` во всех блоках `server`, которые ссылаются на *клиент ACME* с именем *имя*, будет получен единый сертификат; если изменится конфигурация `server_name`, сертификат будет обновлен для учета изменений.

При каждом запуске Angie для всех доменов, 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.

#### **i** Примечание

Сейчас домены, заданные через регулярные выражения, не поддерживаются и будут пропускаться.

Домены со звездочкой поддерживаются только в режиме `challenge=dns` в `acme_client`.

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 <i>имя uri</i> [enabled=on   off] [key_type=<i>тип</i>] [key_bits=<i>число</i>] [email=<i>email</i>] [max_cert_size=<i>число</i>] [renew_before_expiry=<i>время</i>] [retry_after_error=off <i>время</i>] [challenge=dns   http] [account_key=<i>файл</i>];</code>
По умолчанию	—
Контекст	http

Определяет клиент ACME с глобально уникальным *именем*. Оно должно быть допустимым для каталога и будет использоваться без учета регистра.

Вторым обязательным параметром является *uri* каталога ACME. Например, URI каталога Let's Encrypt ACME указан как `https://acme-v02.api.letsencrypt.org/directory`.

Чтобы директива работала, в том же контексте должен быть настроен *resolver*.

#### **i** Примечание

Для тестирования удостоверяющие центры обычно предоставляют отдельные тестовые среды. Например, среда тестирования Let's Encrypt — `https://acme-staging-v02.api.letsencrypt.org/directory`.

<code>enabled</code>	Включает или отключает клиент; это полезно, например, для временного отключения клиента без его удаления из конфигурации. По умолчанию: <code>on</code> .
<code>key_type</code>	Тип алгоритма закрытого ключа для сертификата. Допустимые значения: <code>rsa</code> , <code>ecdsa</code> . По умолчанию: <code>ecdsa</code> .
<code>key_bits</code>	Количество битов в ключе сертификата. По умолчанию: 256 для <code>ecdsa</code> , 2048 для <code>rsa</code> .
<code>email</code>	Необязательный адрес электронной почты для обратной связи; используется при создании учетной записи на сервере CA.
<code>max_cert_size</code>	Задаёт максимальный допустимый размер файла нового сертификата в байтах, чтобы зарезервировать место для нового сертификата в разделяемой памяти; чем больше число доменов, для которых запрашивается сертификат, тем больше требуется места. Если в момент запуска сертификат уже существует, но его размер превышает значение <code>max_cert_size</code> , значение <code>max_cert_size</code> динамически увеличивается до размера существующего файла сертификата. Если размер сертификата, полученного при обновлении, превышает <code>max_cert_size</code> , процесс обновления завершится с ошибкой. По умолчанию: 8192.
<code>renew_before_exp</code>	Время до истечения срока действия сертификата, когда должно начаться его обновление. По умолчанию: 30d.
<code>retry_after_error</code>	Время до повторной попытки, если получить сертификат не удалось. Если задано значение <code>off</code> , клиент не будет снова пытаться получить сертификат после ошибки. По умолчанию: 2h.
<code>challenge</code>	Задаёт тип верификации для ACME-клиента. Допустимые значения: <code>dns</code> , <code>http</code> . По умолчанию: <code>http</code> .
<code>account_key</code>	Указывает полный путь к файлу, содержащему ключ в формате PEM. Это удобно, если вы хотите использовать существующий ключ аккаунта вместо автоматической генерации, или если вам нужно использовать один ключ для нескольких ACME-клиентов. Поддерживаемые типы ключей: <ul style="list-style-type: none"> <li>• RSA-ключи с длиной, кратной 8, в диапазоне от 2048 до 8192 бит.</li> <li>• ECDSA-ключи с длиной 256, 384 или 521 бит.</li> </ul>

### acme\_client\_path

Синтаксис	<code>acme_client_path <i>путь</i>;</code>
По умолчанию	—
Контекст	<code>http</code>

Переопределяет *путь* к каталогу для хранения сертификатов и ключей, заданному при сборке с помощью параметра сборки `--http-acme-client-path`.



### acme\_dns\_port

Синтаксис	acme_dns_port <i>число</i> ;
Значение по умолчанию	acme_dns_port 53;
Контекст	http

Указывает порт, который модуль использует для обработки DNS-запросов от ACME-сервера. Номер порта должен быть в диапазоне от 1 до 65535. Чтобы использовать номер порта 1024 или ниже, Angie должен работать с привилегиями *суперпользователя*.

### acme\_hook

Синтаксис	acme_hook <i>имя</i> ;
По умолчанию	—
Контекст	location

Задаёт location, куда будут проксироваться запросы и данные для внешнего приложения. *Имя* задает соответствующий ACME-клиент.

### 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*.

#### Важно

Файл сертификата доступен, только если клиент ACME получил хотя бы один сертификат, а вот файл ключа доступен сразу после запуска.

\$acme\_hook\_challenge

Тип проверки. Возможные значения: `dns`, `http`.

`$acme_hook_client`

Имя ACME-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 ACME-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.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_addition_module`. В пакетах и образах из наших репозитивов модуль включен в сборку.

## Пример конфигурации

```
location / {
    add_before_body /before_action;
    add_after_body /after_action;
}
```

## Директивы

### add\_before\_body

Синтаксис	<code>add_before_body uri;</code>
По умолчанию	—
Контекст	<code>http, server, location</code>

Добавляет перед телом ответа текст, выдаваемый в результате работы заданного подзапроса. Пустая строка ("") в качестве параметра отменяет добавление, унаследованное с предыдущего уровня конфигурации.

### add\_after\_body

Синтаксис	<code>add_after_body uri;</code>
По умолчанию	—
Контекст	<code>http, server, location</code>

Добавляет после тела 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-<i>min</i> ...;</code>
По умолчанию	<code>addition_types text/html;</code>
Контекст	<code>http, server, location</code>

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 модулей *limit\_conn http*, *limit\_conn stream*, *limit\_req* и *http upstream*.

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 Angie PRO 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/*.

### **i** Примечание

В Angie PRO 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;
}
```

## api\_config\_files

Синтаксис	api_config_files on   off;
По умолчанию	off
Контекст	location

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": "1.8.2",
  "address": "192.168.16.5",
  "generation": 1,
  "load_time": "2025-02-13T12:58:39.789Z",
  "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;

        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": "1.8.2",
    "address": "192.168.16.5",
    "generation": 1,
    "load_time": "2025-02-13T12:58:39.789Z"
  },
  "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
        },
        "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/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
```

### Примечание

По умолчанию модуль использует для дат строки в формате ISO 8601; чтобы вместо этого использовать целочисленный формат эпохи UNIX, добавьте параметр `date=epoch` к строке запроса:

```
$ curl https://www.example.com/status/angie/load_time
"2024-04-01T00:59:59+01:00"

$ curl https://www.example.com/status/angie/load_time?date=epoch
1711929599
```

## Состояние сервера

`/status/angie`

```
{
  "version": "1.8.2",
  "address": "192.168.16.5",
  "generation": 1,
  "load_time": "2025-02-13T16:15:43.805Z"
  "config_files": {
    "/etc/angie/angie.conf": "...",
    "/etc/angie/mime.types": "..."
  }
}
```

<code>version</code>	Строка; версия запущенного сервера Angie
<code>build (PRO)</code>	Строка; сборка, если указана при компиляции
<code>address</code>	Строка; адрес сервера, принявшего запрос к API
<code>generation</code>	Число; версия (поколение) конфигурации, отсчитываемая с последнего запуска Angie
<code>load_time</code>	Строка; время последней перезагрузки конфигурации в формате <i>даты</i> ; строковые значения даются с миллисекундным разрешением
<code>config_files</code>	Объект; его члены — абсолютные имена всех файлов конфигурации Angie, загруженных сейчас экземпляром сервера, а их значения — строковые представления содержимого файлов, например:

```
{
  "/etc/angie/angie.conf": "server {\n  listen 80;\n  # ... \n\n}\n\n"}
}
```

### Осторожно

Объект `config_files` есть в `/status/angie/`, только если включена директива `api_config_files`.

## 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, таких как *limit\_conn*, *limit\_req* и *HTTP cache*:

```
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;
```

В указанной таким образом зоне разделяемой памяти будет собираться следующая статистика:

pages	Объект; статистика по страницам памяти
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:

<b>queries</b>	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)
<b>responses</b>	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
<b>sent</b>	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

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;

```

В указанной таким образом зоне разделяемой памяти будет собираться следующая статистика:

<b>ssl</b>	Объект; SSL-метрики. Присутствует, если в <b>server</b> есть <b>listen ssl</b> ;
<b>handshaked</b>	Число; суммарное количество успешных SSL-рукопожатий
<b>reuses</b>	Число; суммарное количество повторных использований SSL-сессий во время SSL-рукопожатий
<b>timedout</b>	Число; суммарное количество SSL-рукопожатий с истекшим таймаутом
<b>failed</b>	Число; суммарное количество неуспешных SSL-рукопожатий
<b>requests</b>	Объект; метрики запросов
<b>total</b>	Число; суммарное количество клиентских запросов
<b>processing</b>	Число; текущее количество обслуживаемых клиентских запросов
<b>discarded</b>	Число; суммарное количество запросов завершенных без отправки ответа
<b>responses</b>	Объект; метрики ответов
<code>	Число; ненулевое количество ответов со статусом <code> (100-599)
xxx	Число; ненулевое количество ответов с другим кодом статуса
<b>data</b>	Объект; метрики данных
<b>received</b>	Число; суммарное количество байт, полученное от клиентов
<b>sent</b>	Число; суммарное количество байт, отправленное клиентам

Пример:

```

"ssl": {
  "handshaked": 4174,

```

```

"reuses": 0,
"timedout": 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:

<b>requests</b>	Объект; метрики запросов
total	Число; суммарное количество клиентских запросов
discarded	Число; суммарное количество запросов завершенных без отправки ответа
<b>responses</b>	Объект; метрики ответов
<code>	Число; ненулевое количество ответов со статусом <code> (100-599)
xxx	Число; ненулевое количество ответов с другим кодом статуса
<b>data</b>	Объект; метрики данных
received	Число; суммарное количество байт, полученное от клиентов
sent	Число; суммарное количество байт, отправленное клиентам

Пример:



```
{
  "requests": {
    "total": 4158,
    "discarded": 0
  },
  "responses": {
    "200": 4157,
    "304": 1
  },
  "data": {
    "received": 538200,
    "sent": 177606236
  }
}
```

### 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 *\$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:

<b>ssl</b>	Объект; SSL-метрики. Присутствует, если в <b>server</b> есть <b>listen ssl</b> ;
<b>handshaked</b>	Число; суммарное количество успешных SSL-рукопожатий
<b>reuses</b>	Число; суммарное количество повторных использований SSL-сессий во время SSL-рукопожатий
<b>timedout</b>	Число; суммарное количество SSL-рукопожатий с истекшим таймаутом
<b>failed</b>	Число; суммарное количество неуспешных SSL-рукопожатий
<b>connections</b>	Объект; метрики соединений
<b>total</b>	Число; суммарное количество клиентских соединений
<b>processing</b>	Число; текущее количество обслуживаемых клиентских соединений
<b>discarded</b>	Число; суммарное количество клиентских соединений, завершенных без создания сессии
<b>passed</b>	Число; суммарное количество клиентских соединений, переданных на другой прослушивающий порт директивами <b>pass</b>
<b>sessions</b>	Объект; метрики сессий
<b>success</b>	Число; количество сессий, завершенных с кодом 200, что означает успешное завершение
<b>invalid</b>	Число; количество сессий, завершенных с кодом 400, случается, когда сервер не может прочитать данные от клиента, например, заголовок <b>PROXY protocol</b>
<b>forbidden</b>	Число; количество сессий, завершенных с кодом 403, когда доступ запрещен, например, ограничен для определенного адреса клиента
<b>internal_error</b>	Число; количество сессий, завершенных с кодом 500, внутренняя ошибка сервера
<b>bad_gateway</b>	Число; количество сессий, завершенных с кодом 502, Bad Gateway, если, например, сервер в <b>upstream</b> недоступен или не может быть выбран
<b>service_unavailable</b>	Число; количество сессий, завершенных с кодом 503, Service Unavailable, если, например, доступ ограничен числом входящих соединений
<b>data</b>	Объект; метрики данных
<b>received</b>	Число; суммарное количество байт, полученное от клиентов
<b>sent</b>	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
}
```

### HTTP caches

```
proxy_cache cache_zone;
```

/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; текущий размер кэша
max_size	Число; ограничение на максимальный размер кэша, если задано в конфигурации
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

Добавлено в версии 1.2.0: PRO

В Angie PRO при включении *шардинга кэша* с помощью директив *proxy\_cache\_path* отдельные шарды указываются как объекты-члены в объекте **shards**:

<b>shards</b>	Объект; его члены — отдельные шарды
<b>&lt;shard&gt;</b>	Объект; представляет отдельный шард, а имя объекта — путь кэша
<b>sizes</b>	Число; текущий размер шарда
<b>max_size</b>	Число; максимальный размер шарда, если задан в конфигурации
<b>cold</b>	Логическое значение; <b>true</b> , 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
      }
    }
  }
}
```

### 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 сконфигурованных зон: *limit\_conn в http* или *limit\_conn в stream* содержит следующие данные:

```
{
  "passed": 73,
  "skipped": 0,
  "rejected": 0,
  "exhausted": 0
}
```

<b>passed</b>	Число; суммарное количество переданных на проксируемый сервер соединений
<b>skipped</b>	Число; суммарное количество соединений, переданных с нулевым или превосходящим 255 байт <key>
<b>rejected</b>	Число; суммарное количество соединений сверх сконфигурованного ограничения
<b>exhausted</b>	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

Добавлено в версии 1.1.0.

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;
}
```

/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
    },

    "sid": "<server_id>"
  },
  "keepalive": 2
}

```

<code>peers</code>	Объект; содержит метрики всех пиров апстрима во вложенных объектах, имена которых — канонические представления адресов этих пиров. Внутри каждого вложенного объекта:
<code>server</code>	Строка; сервер, как он указан в директиве <code>server</code>
<code>service</code>	Строка; имя сервиса, указанное в директиве <code>server</code> , если сконфигурировано
<code>slow_start</code> (PRO 1.4.0+)	Число; заданное для сервера значение <code>slow_start</code> , выраженное в секундах. При задании значения через <i>соответствующий подраздел</i> API динамической конфигурации можно указать не только число секунд, но и время с точностью до миллисекунд.
<code>backup</code>	Логическое значение; <code>true</code> для backup серверов
<code>weight</code>	Число; сконфигурированный <code>weight</code>
<code>state</code>	Строка; текущее состояние пира, и какие запросы ему отправляются: <ul style="list-style-type: none"> <li>• <code>checking</code> (PRO): настроен как <code>essential</code> и проверяется, отправляются только <i>проверочные запросы</i>;</li> <li>• <code>down</code>: отключен вручную, не отправляются никакие запросы;</li> <li>• <code>draining</code> (PRO): аналогичен <code>down</code>, но отправляются запросы сессий, привязанных ранее через <code>sticky</code>;</li> <li>• <code>recovering</code>: восстанавливается после сбоя согласно <code>slow_start</code>, отправляется все больше запросов;</li> <li>• <code>unavailable</code>: достиг предела <code>max_fails</code>, отправляются пробные клиентские запросы с интервалом <code>fail_timeout</code>;</li> <li>• <code>unhealthy</code> (PRO): неработающий, отправляются только <i>проверочные запросы</i>;</li> <li>• <code>up</code>: работоспособен, запросы отправляются как обычно.</li> </ul>
<code>selected</code>	Объект; статистика выбора пиров
<code>current</code>	Число; текущее количество соединений к пиру
<code>total</code>	Число; общее количество запросов переданных пиру
<code>last</code>	Строка или число; время последнего выбора пира в формате <i>даты</i>
<code>max_conns</code>	Число; <i>максимальное</i> количество одновременных активных соединений к пиру, если сконфигурировано
<code>responses</code>	Объект; статистика ответов
<code>&lt;code&gt;</code>	Число; ненулевое количество ответов со статусом <code>&lt;code&gt;</code> (100-599)
<code>xxx</code>	Число; ненулевое количество ответов с другим кодом статуса
<code>data</code>	Объект; метрики данных
<code>received</code>	Число; суммарное количество байт, полученное от пира
<code>sent</code>	Число; суммарное количество байт, отправленное пиру
<code>health</code>	Объект; статистика по состоянию пира
<code>fails</code>	Число; общее количество неудачных попыток работы с пиром
<code>unavailable</code>	Число; столько раз пир становился <code>unavailable</code> по достижении значения <code>max_fails</code>
<code>downtime</code>	Число; суммарное время (в миллисекундах), в течение которого пир был недоступен для выбора как <code>unavailable</code>
<code>downstart</code>	Строка или число; время, когда пир стал <code>unavailable</code> , в формате <i>даты</i>
<code>header_time</code> (PRO 1.3.0+)	Число; среднее время (в миллисекундах) получения заголовков ответа от сервера; см. директиву <code>response_time_factor</code> (PRO)
<code>response_time</code> (PRO 1.3.0+)	Число; среднее время (в миллисекундах) получения ответа от сервера; см. директиву <code>response_time_factor</code> (PRO)
<code>sid</code>	Строка; <i>id сервера</i> , указанный в конфигурации апстрима
<code>keepalive</code>	Число; текущее количество кэшированных соединений



## health/probes (PRO)

Изменено в версии 1.2.0: PRO

Если для апстрима настроены проверки *upstream\_probe (PRO)*, 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": "2025-02-13T09:56:07Z"
      }
    }
  }
}
```

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`:

<code>count</code>	Число; общее количество проверок этого сервера
<code>fails</code>	Число; количество неуспешных проверок
<code>last</code>	Строка или число; время последней проверки в формате <i>даты</i>

## queue (PRO)

Изменено в версии 1.4.0: PRO

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;
}
```

`/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
      }
    }
  }
}
```

<code>peers</code>	Объект; содержит метрики всех пиров апстрима во вложенных объектах, имена которых — канонические представления адресов этих пиров. Внутри каждого вложенного объекта:
<code>server</code>	Строка; адрес, заданный директивой <code>server</code>
<code>service</code>	Строка; имя сервиса, если оно указано в директиве <code>server</code>
<code>slow_start</code> (PRO 1.4.0+)	Число; заданное для сервера значение <code>slow_start</code> , выраженное в секундах. При задании значения через <i>соответствующий подраздел</i> API динамической конфигурации можно указать не только число секунд, но и время с точностью до миллисекунд.
<code>backup</code>	Логическое значение; <code>true</code> для запасных серверов
<code>weight</code>	Число; заданный для пира <i>вес</i>
<code>state</code>	Строка; текущее состояние пира: <ul style="list-style-type: none"> <li>• <code>up</code>: работоспособен, запросы отправляются как обычно;</li> <li>• <code>down</code>: отключен вручную, не отправляются никакие запросы;</li> <li>• <code>draining</code> (PRO): аналогичен <code>down</code>, но отправляются запросы сессий, привязанных ранее через <code>sticky</code>;</li> <li>• <code>unavailable</code>: достиг предела <code>max_fails</code>, отправляются пробные клиентские запросы с интервалом <code>fail_timeout</code>;</li> <li>• <code>recovering</code>: восстанавливается после сбоя согласно <code>slow_start</code>, отправляется все больше запросов;</li> <li>• <code>checking</code> (PRO): настроен как <code>essential</code> и проверяется, отправляются только <i>проверочные запросы</i>;</li> <li>• <code>unhealthy</code> (PRO): неработающий, отправляются только <i>проверочные запросы</i>.</li> </ul>
<code>selected</code>	Объект; статистика выбора этого пира для подключения
<code>current</code>	Число; текущее количество подключений к пиру
<code>total</code>	Число; общее количество подключений, направленных этому пиру
<code>last</code>	Строка или число; время, когда пир был выбран в последний раз, в формате <i>даты</i>
<code>max_conns</code>	Число; <i>максимальное</i> количество одновременных активных подключений к пиру, если оно задано
<code>data</code>	Объект; статистика передачи данных
<code>received</code>	Число; общее количество байт, полученное от пира
<code>sent</code>	Число; общее количество байт, отправленное пиру
<code>health</code>	Объект; статистика по состоянию пира
<code>fails</code>	Число; общее количество неудачных попыток связаться с пиром
<code>unavailable</code>	Число; общее количество переходов в состояние <code>unavailable</code> по достижении значения <code>max_fails</code>
<code>downtime</code>	Число; общее время в миллисекундах, в течение которого пир находился в состоянии <code>unavailable</code> (недоступен для выбора)
<code>downstart</code>	Строка или число; время, когда пир последний раз стал <code>unavailable</code> , в формате <i>даты</i>
<code>connect_time</code> (PRO 1.4.0+)	Число; среднее время (в миллисекундах) установления соединения с сервером; см. директиву <code>response_time_factor</code> (PRO)
<code>first_byte_time</code> (PRO 1.4.0+)	Число; среднее время (в миллисекундах) получения первого байта ответа от сервера; см. директиву <code>response_time_factor</code> (PRO)
<code>last_byte_time</code> (PRO 1.4.0+)	Число; среднее время (в миллисекундах) получения полного ответа от сервера; см. директиву <code>response_time_factor</code> (PRO)

Изменено в версии 1.4.0: PRO

Если в Angie PRO для апстрима настроены проверки `upstream_probe` (PRO), 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": "2025-02-13T11:03:54Z"
      }
    }
  }
}
```

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`:

<code>count</code>	Число; общее количество проверок этого сервера
<code>fails</code>	Число; количество неуспешных проверок
<code>last</code>	Строка или число; время последней проверки в формате <i>даты</i>

### API динамической конфигурации (PRO)

Добавлено в версии 1.2.0: PRO

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, планомерно 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 <code>upstream</code> ; чтобы настраивать его через <code>/config</code> , в нем должна быть задана директива <code>zone</code> , определяющая зону разделяемой памяти.
<имя>	Имя конкретного сервера в составе указанного <code>&lt;upstream&gt;</code> ; задается в формате <code>&lt;service&gt;@&lt;host&gt;</code> , где: <ul style="list-style-type: none"> <li>• <code>&lt;service&gt;@</code> — необязательная часть, задающая имя сервиса в целях разрешения SRV-записей.</li> <li>• <code>&lt;host&gt;</code> — доменное имя сервиса (при наличии <code>resolve</code>) или IP-адрес; также можно указать порт.</li> </ul>

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`, `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,
  "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."
}
```

**Примечание**

Даже с совместимым методом балансировки параметр `backup` можно задать лишь при добавлении нового сервера.

`/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>&lt;upstream&gt;</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>&lt;имя&gt;</code>	Имя конкретного сервера в составе указанного <code>&lt;upstream&gt;</code> ; задается в формате <code>&lt;service&gt;@&lt;host&gt;</code> , где: <ul style="list-style-type: none"> <li><code>&lt;service&gt;@</code> — необязательная часть, задающая имя сервиса в целях разрешения SRV-записей.</li> <li><code>&lt;host&gt;</code> — доменное имя сервиса (при наличии <code>resolve</code>) или IP-адрес; также можно указать порт.</li> </ul>

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`, `backup` и `down`, описанные в разделе `server`.

### **i** 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,
  "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."
}
```

### **i** Примечание

Даже с совместимым методом балансировки параметр `backup` можно задать лишь при добавлении нового сервера.

При удалении серверов можно установить аргумент `connection_drop=<значение>` (PRO), 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`:

```
$ 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,
    "backup": false,
    "down": false,
    "sid": ""
  }
}

```

## 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,
  "backup": false,
  "down": false,
  "sid": ""
}
```

Параметр `max_fails` вернулся к значению по умолчанию.

При удалении серверов можно установить аргумент `connection_drop=<значение>` (PRO), 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 допустимо использование переменных.

<code>off</code>	отменяет действие унаследованной с предыдущего уровня конфигурации директивы <code>auth_basic</code>
------------------	--

### auth\_basic\_user\_file

Синтаксис	<code>auth_basic_user_file файл;</code>
По умолчанию	—
Контекст	<code>http, server, location, limit_except</code>

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).

#### Осторожно

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.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_auth_request_module`. В пакетах и образах из наших репозиториях модуль включен в сборку.

Модуль может быть скомбинирован с другими модулями доступа, такими как *Access* и *Auth Basic* с помощью директивы *satisfy*.

## 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	http, server, location

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	http, server, location

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	<code>autoindex_format html   xml   json   jsonp;</code>
По умолчанию	<code>autoindex_format html;</code>
Контекст	<code>http, server, location</code>

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*.

## autoindex\_localtime

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

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

`$modern_browser`

равна значению, заданному директивой *modern\_browser\_value*, если браузер опознан как современный;

`$ancient_browser`

равна значению, заданному директивой *ancient\_browser\_value*, если браузер опознан как устаревший;

`$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 строка;</code>
По умолчанию	<code>ancient_browser_value 1;</code>
Контекст	http, server, location

Задаёт значение для переменных `$ancient_browser`.

### modern\_browser

Синтаксис	<code>modern_browser браузер версия;</code> <code>modern_browser 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 строка;</code>
По умолчанию	<code>modern_browser_value 1;</code>
Контекст	<code>http, server, location</code>

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	<code>charset кодировка   off;</code>
По умолчанию	<code>charset off;</code>
Контекст	<code>http, server, location, if в location</code>

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	<code>charset_map кодировка1 кодировка2 { ... }</code>
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 "`Љ#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-<i>mun</i> ...;</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 <i>кодировка</i>;</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.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_dav_module`. В пакетах и образах из наших репозиториях модуль включен в сборку.

### Важно

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   метод ...;</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 число;</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
```

## Empty GIF

Выдает однопиксельный прозрачный GIF.

### Пример конфигурации

```
location = /_gif {
    empty_gif;
}
```

### Директивы

#### empty\_gif

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

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	http, server, location

Задаёт локальный IP-адрес с необязательным портом, который будет использоваться в исходящих соединениях с FastCGI-сервером. В значении параметра допустимо использование переменных. Специальное значение `off` отменяет действие унаследованной с предыдущего уровня конфигурации директивы `fastcgi_bind`, позволяя системе самостоятельно выбирать локальный IP-адрес и порт.

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 *superпользователя*. В Linux это не требуется, так как если указан параметр *transparent*, то рабочие процессы наследуют *capability CAP\_NET\_RAW* из главного процесса.

### Важно

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.

<b>off</b>	Angie принимает ответ FastCGI-сервера как можно быстрее, сохраняя его в буферы, заданные директивами <i>fastcgi_buffer_size</i> и <i>fastcgi_buffers</i> . Если ответ не влезает целиком в память, то его часть может быть записана на диск во <i>временный файл</i> . Запись во временные файлы контролируется директивами <i>fastcgi_max_temp_file_size</i> и <i>fastcgi_temp_file_write_size</i> .
<b>on</b>	ответ синхронно передается клиенту сразу же по мере его поступления. Angie не пытается считать весь ответ FastCGI-сервера. Максимальный размер данных, который Angie может принять от сервера за один раз, задается директивой <i>fastcgi_buffer_size</i> .

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	<code>fastcgi_buffers число размер;</code>
По умолчанию	<code>fastcgi_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 FastCGI-сервера.

По умолчанию размер одного буфера равен размеру страницы. В зависимости от платформы это или 4К, или 8К.

### fastcgi\_busy\_buffers\_size

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

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 двумя буферами, заданными 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, используемой 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.

### 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	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:

```
fastcgi_cache_bypass $cookie_nocache $arg_nocache$arg_comment;
fastcgi_cache_bypass $http_pragma $http_authorization;
```

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, 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</code> <i>время</i> ;
По умолчанию	<code>fastcgi_cache_lock_timeout 5s</code> ;
Контекст	<code>http, server, location</code>

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

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</code> GET   HEAD   POST ...;
По умолчанию	<code>fastcgi_cache_methods</code> GET HEAD;
Контекст	<code>http, server, location</code>

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</code> <i>число</i> ;
По умолчанию	<code>fastcgi_cache_min_uses</code> 1;
Контекст	<code>http, server, location</code>

Задаёт число запросов, после которого ответ будет кэширован.

## fastcgi\_cache\_path

Синтаксис	<code>fastcgi_cache_path <i>путь</i> [levels=<i>уровни</i>] [use_temp_path=<i>on</i>   <i>off</i>] [keys_zone=<i>имя:размер</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.

Параметр `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.

Какой из каталогов будет использоваться для временных файлов определяется параметром `use_temp_path`.

<code>on</code>	Если параметр не задан или установлен в значение "on", 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.

Кроме того, все активные ключи и информация о данных хранятся в зоне разделяемой памяти, имя и размер которой задаются параметром `keys_zone`. Зоны размером в 1 мегабайт достаточно для хранения около 8 тысяч ключей.

Если к данным кэша не обращаются в течение времени, заданного параметром `inactive`, то данные удаляются, независимо от их свежести.

По умолчанию `inactive` 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.

<code>max_size</code>	максимальное пороговое значение размера кэша
<code>min_free</code>	минимальное пороговое значение объема свободного места на файловой системе с кэшем
<code>manager_files</code>	максимальное количество удаляемых элементов кэша за одну итерацию По умолчанию: 100
<code>manager_threshold</code>	ограничивает время работы одной итерации По умолчанию: 200 миллисекунд
<code>manager_sleep</code>	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.

<code>loader_files</code>	максимальное количество элементов кэша к загрузке в одну итерацию По умолчанию: 100
<code>loader_threshold</code>	ограничивает время работы одной итерации По умолчанию: 200 миллисекунд
<code>loader_sleep</code>	время, в течение которого выдерживается пауза между итерациями По умолчанию: 50 миллисекунд

### fastcgi\_cache\_revalidate

Синтаксис	<code>fastcgi_cache_revalidate on   off;</code>
По умолчанию	<code>fastcgi_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".

### fastcgi\_cache\_use\_stale

Синтаксис	<code>fastcgi_cache_use_stale error   timeout   invalid_header   updating   http_500   http_503   http_403   http_429   off ...;</code>
По умолчанию	<code>fastcgi_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 `fastcgi_next_upstream`.

<code>error</code>	позволяет использовать устаревший кэшированный ответ при невозможности выбора FastCGI-сервера для обработки запроса.
<code>updating</code>	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.

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	<code>fastcgi_cache_valid [код ...] время;</code>
По умолчанию	—
Контекст	<code>http, server, location</code>

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" задаёт время кэширования ответа в секундах. Значение `0` запрещает кэшировать ответ. Если значение начинается с префикса `@`, оно задаёт 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	<code>fastcgi_catch_stderr строка;</code>
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	<code>fastcgi_connect_timeout время;</code>
По умолчанию	<code>fastcgi_connect_timeout 60s;</code>
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.

### 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 соединений с проксируемым сервером, если он был удален из группы или помечен как постоянно недоступный в результате процесса *resolve* или *команды API DELETE*.

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	<code>http, server, location</code>

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	<code>http, server, location</code>

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	<code>http, server, location</code>

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	<code>http, server, location</code>

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

Задает 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\_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 <i>скорость</i>;</code>
По умолчанию	<code>fastcgi_limit_rate 0;</code>
Контекст	<code>http, server, location</code>

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 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 <i>размер</i>;</code>
По умолчанию	<code>fastcgi_max_temp_file_size 1024m;</code>
Контекст	<code>http, server, location</code>

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	<code>http, server, location</code>

Определяет, в каких случаях запрос будет передан следующему 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_503</code>	сервер вернул ответ с кодом 503;
<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 <b>неидемпотентным</b> 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;
<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, то исправить это уже невозможно.

Директива также определяет, что считается *неудачной попыткой* работы с сервером.

<code>error</code>	всегда считаются неудачными попытками, даже если они не указаны в директиве
<code>timeout</code>	
<code>invalid_header</code>	
<code>http_500</code>	считаются неудачными попытками, только если они указаны в директиве
<code>http_503</code>	
<code>http_429</code>	
<code>http_403</code>	никогда не считаются неудачными попытками
<code>http_404</code>	

Передача запроса следующему серверу может быть ограничена по *количеству попыток* и по *времени*.

### `fastcgi_next_upstream_timeout`

Синтаксис	<code>fastcgi_next_upstream_timeout</code> <i>время</i> ;
По умолчанию	<code>fastcgi_next_upstream_timeout</code> 0;
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 число;</code>
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 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:

```
fastcgi_no_cache $cookie_nocache $arg_nocache$arg_comment;
fastcgi_no_cache $http_pragma $http_authorization;
```

Можно использовать совместно с директивой `fastcgi_cache_bypass`.

### fastcgi\_param

Синтаксис	<code>fastcgi_param параметр значение [if_not_empty];</code>
По умолчанию	—
Контекст	http, server, location

Задаёт параметр, который будет передаваться FastCGI-серверу. В качестве значения можно использовать текст, переменные и их комбинации. Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы `fastcgi_param`.

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.

## 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	<code>http, server, location</code>

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	<code>http, server, location</code>

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.

<code>on</code>	тело запроса полностью <i>читается</i> от клиента перед отправкой запроса на FastCGI-сервер.
<code>off</code>	тело запроса отправляется на FastCGI-сервер сразу же по мере его поступления. В этом случае запрос не может быть передан <i>следующему серверу</i> , 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 `queue`, либо при помощи параметра сокета `SO_SNDLOWAT`, с указанным размером.

**i Примечание**

Эта директива игнорируется на Linux, Solaris и Windows.

### fastcgi\_send\_timeout

Синтаксис	<code>fastcgi_send_timeout <i>время</i>;</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 <code>SO_KEEPALIVE</code>

### fastcgi\_split\_path\_info

Синтаксис	<code>fastcgi_split_path_info <i>regex</i>;</code>
По умолчанию	—
Контекст	<code>location</code>

Задаёт регулярное выражение, выделяющее значение для переменной `$fastcgi_path_info`. Регулярное выражение должно иметь два выделения, из которых первое становится значением переменной `$fastcgi_script_name`, а второе - 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   строка;</code>
По умолчанию	<code>fastcgi_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

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	<code>http, server, location</code>

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\_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	<code>http, server, location</code>

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 путь [уровень1 [уровень2 [уровень3]]]`;</code>
По умолчанию	<code>fastcgi_temp_path fastcgi_temp;</code> (путь зависит от параметра сборки <code>--http-fastcgi-temp-path</code> )
Контекст	<code>http, server, location</code>

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_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 первого выделения, задаваемого этой директивой.

`$fastcgi_path_info`

значение второго выделения, задаваемого директивой *fastcgi\_split\_path\_info*. Эту переменную можно использовать для задания параметра PATH\_INFO.

### 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.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_flv_module`. В пакетах и образах из наших репозиториев модуль включен в сборку.

### 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 {
    ...;
}
```

### **i** 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 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:

<code>delete</code>	удаляет описанную сеть
<code>default</code>	значение переменной, если адрес клиента не соответствует ни одному из заданных адресов. При задании адресов в формате CIDR вместо <code>default</code> можно использовать " <code>0.0.0.0/0</code> " и " <code>::/0</code> ". Если параметр <code>default</code> не указан, значением по умолчанию будет пустая строка.
<code>include</code>	включает файл с адресами и значениями. Включений может быть несколько.
<code>proxy</code>	задает доверенные адреса, при запросе с которых будет использоваться адрес в переданном поле заголовка запроса " <code>X-Forwarded-For</code> ". В отличие от обычных адресов, доверенные адреса проверяются последовательно.
<code>proxy_recursive</code>	включает рекурсивный поиск адреса. При выключенном рекурсивном поиске вместо исходного адреса клиента, совпадающего с одним из доверенных адресов, будет использоваться последний адрес, переданный в " <code>X-Forwarded-For</code> ". При включенном рекурсивном поиске вместо исходного адреса клиента, совпадающего с одним из доверенных адресов, будет использоваться последний не доверенный адрес, переданный в " <code>X-Forwarded-For</code> ".
<code>ranges</code>	указывает, что адреса задаются в виде диапазонов. Этот параметр должен быть первым. Для ускорения загрузки гео-базы нужно располагать адреса в порядке возрастания.

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;
}
```

## GeoIP

Создает переменные, значения которых зависят от IP-адреса клиента, используя готовые базы данных [MaxMind](#) или их аналоги.

При использовании баз данных с поддержкой IPv6 IPv4-адреса ищутся отображенными на IPv6.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_geoip_module`.

### Важно

Для этого модуля нужна база данных [MaxMind GeoIP](#) или ее аналог, например [MaxMind GeoLite2](#) или ЦМУ ССОП.

## Пример конфигурации

```
http {
  geoip_country      GeoIP.dat;
  geoip_city         GeoLiteCity.dat;
  geoip_proxy        192.168.100.0/24;
  geoip_proxy        2001:0db8::/32;
  geoip_proxy_recursive on;
  ...
}
```

## Директивы

### geoip\_country

Синтаксис	<code>geoip_country <i>файл</i>;</code>
По умолчанию	—
Контекст	<code>http</code>

Задаёт базу данных для определения страны в зависимости от значения IP-адреса клиента. При использовании этой базы данных доступны следующие переменные:

\$geoip_country_c	двухбуквенный код страны, например, "RU", "US".
\$geoip_country_c	трехбуквенный код страны, например, "RUS", "USA".
\$geoip_country_n	название страны, например, "Russian Federation", "United States".

### geoip\_city

Синтаксис	geoip_city <i>файл</i> ;
По умолчанию	—
Контекст	http

Задаёт базу данных для определения страны, региона и города в зависимости от значения IP-адреса клиента. При использовании этой базы данных доступны следующие переменные:

\$geoip_city_cont	двухбуквенный код континента, например, "EU", "NA".
\$geoip_city_coun	двухбуквенный код страны, например, "RU", "US".
\$geoip_city_coun	трехбуквенный код страны, например, "RUS", "USA".
\$geoip_city_coun	название страны, например, "Russian Federation", "United States".
\$geoip_dma_code	DMA-код региона в США (также известный как "код агломерации"), согласно <a href="#">геотаргетингу Google AdWords API</a> .
\$geoip_latitude	широта.
\$geoip_longitude	долгота.
\$geoip_region	двухсимвольный код региона страны (область, край, штат, провинция, федеральная земля и тому подобное), например, "48", "DC".
\$geoip_region_na	название региона страны (область, край, штат, провинция, федеральная земля и тому подобное), например, "Moscow City", "District of Columbia".
\$geoip_city	название города, например, "Moscow", "Washington".
\$geoip_postal_co	почтовый индекс.

### geoip\_org

Синтаксис	geoip_org <i>файл</i> ;
По умолчанию	—
Контекст	http

Задаёт базу данных для определения названия организации в зависимости от значения IP-адреса клиента. При использовании этой базы данных доступна следующая переменная:

\$geoip_org	название организации, например, "The University of Melbourne".
-------------	--

### geoip\_proxy

Синтаксис	geoip_proxy <i>файл</i> ;
По умолчанию	—
Контекст	http

Задаёт доверенные адреса, при запросе с которых будет использоваться адрес в переданном поле заголовка запроса "X-Forwarded-For".

### geoip\_proxy\_recursive

Синтаксис	geoip_proxy_recursive on   off;
По умолчанию	geoip_proxy_recursive off;
Контекст	http

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".

### gRPC

Позволяет передавать запросы gRPC-серверу.

#### **Важно**

Для работы этого модуля необходим модуль *HTTP2*.

### Пример конфигурации

```
server {
    listen 9000;

    http2 on;

    location / {
        grpc_pass 127.0.0.1:9000;
    }
}
```

## Директивы

### grpc\_bind

Синтаксис	<code>grpc_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 gRPC-сервером. В значении параметра допустимо использование переменных. Специальное значение `off` отменяет действие унаследованной с предыдущего уровня конфигурации директивы `grpc_bind`, позволяя системе самостоятельно выбирать локальный IP-адрес и порт.

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.

#### Важно

Необходимо настроить таблицу маршрутизации ядра для перехвата сетевого трафика с gRPC-сервера.

### grpc\_buffer\_size

Синтаксис	<code>grpc_buffer_size размер;</code>
По умолчанию	<code>grpc_buffer_size 4k 8k;</code>
Контекст	http, server, location

Задаёт размер буфера, в который будет читаться первая часть ответа, получаемого от gRPC-сервера. Ответ синхронно передаётся клиенту сразу же по мере его поступления.

### grpc\_connect\_timeout

Синтаксис	<code>grpc_connect_timeout время;</code>
По умолчанию	<code>grpc_connect_timeout 60s;</code>
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.

### grpc\_connection\_drop

Синтаксис	<code>grpc_connection_drop <i>время</i>   on   off;</code>
По умолчанию	<code>grpc_connection_drop off;</code>
Контекст	http, server, location

Настраивает завершение всех соединений с проксируемым сервером, если он был удален из группы или помечен как постоянно недоступный в результате процесса *resolve* или *команды API DELETE*.

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 <i>поле</i>;</code>
По умолчанию	—
Контекст	http, server, location

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 <i>поле ...</i>;</code>
По умолчанию	—
Контекст	http, server, location

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	grpc_intercept_errors on   off;
По умолчанию	grpc_intercept_errors off;
Контекст	http, server, location

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\_next\_upstream

Синтаксис	grpc_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	grpc_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 <b>неидемпотентным</b> 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_502 http_503 http_504 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*.

### grpc\_next\_upstream\_timeout

Синтаксис	grpc_next_upstream_timeout <i>время</i> ;
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.

### grpc\_pass\_header

Синтаксис	grpc_pass_header <i>поле</i> ;
По умолчанию	—
Контекст	http, server, location

Разрешает передавать от gRPC-сервера клиенту *запрещенные для передачи* поля заголовка.

### grpc\_read\_timeout

Синтаксис	grpc_read_timeout <i>время</i> ;
По умолчанию	grpc_read_timeout 60s;
Контекст	http, server, location

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	grpc_send_timeout <i>время</i> ;
По умолчанию	grpc_send_timeout 60s;
Контекст	http, server, location

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	http, server, location

Позволяет переопределять или добавлять поля заголовка запроса, *передаваемые* проксируемому серверу. В качестве *значения* можно использовать текст, переменные и их комбинации. 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	<code>grpc_socket_keepalive on   off;</code>
По умолчанию	<code>grpc_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 <code>SO_KEEPALIVE</code>

### grpc\_ssl\_certificate

Синтаксис	<code>grpc_ssl_certificate файл;</code>
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\_key

Синтаксис	<code>grpc_ssl_certificate_key файл;</code>
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.

В имени файла можно использовать переменные.

### grpc\_ssl\_ciphers

Синтаксис	<code>grpc_ssl_ciphers <i>шифры</i>;</code>
По умолчанию	<code>grpc_ssl_ciphers DEFAULT;</code>
Контекст	<code>http, server, location</code>

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`.

### grpc\_ssl\_conf\_command

Синтаксис	<code>grpc_ssl_conf_command <i>имя значение</i>;</code>
По умолчанию	—
Контекст	<code>http, server, location</code>

Задаёт произвольные конфигурационные команды OpenSSL при установлении соединения с gRPC SSL-сервером.

#### Важно

Директива поддерживается при использовании OpenSSL 1.0.2 и выше.

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`.

#### Осторожно

Следует учитывать, что изменение настроек OpenSSL напрямую может привести к неожиданному поведению.

### grpc\_ssl\_crl

Синтаксис	grpc_ssl_crl <i>файл</i> ;
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	grpc_ssl_name <i>имя</i> ;
По умолчанию	grpc_ssl_name `имя хоста из grpc_pass`;
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.

По умолчанию используется имя хоста из *grpc\_pass*.

### grpc\_ssl\_password\_file

Синтаксис	grpc_ssl_password_file <i>файл</i> ;
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	grpc_ssl_protocols [SSLv2] [SSLv3] [TLSv1] [TLSv1.1] [TLSv1.2] [TLSv1.3];
По умолчанию	grpc_ssl_protocols TLSv1.2 TLSv1.3;
Контекст	http, server, location

Изменено в версии 1.2.0: Параметр TLSv1.3 добавлен к используемым по умолчанию.

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 CA 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	<code>grpc_ssl_verify_depth <i>число</i>;</code>
По умолчанию	<code>grpc_ssl_verify_depth 1;</code>
Контекст	<code>http, server, location</code>

Устанавливает глубину проверки в цепочке сертификатов gRPC SSL-сервера.

## 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.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_gunzip_module`. В пакетах и образах из наших репозиториях модуль включен в сборку.

### Пример конфигурации

```
location /storage/ {
    gunzip on;
    # ...
}
```

## Директивы

### gunzip

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

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	<code>gunzip_buffers <i>число размер</i>;</code>
По умолчанию	<code>gunzip_buffers 32 4k   16 8k;</code>
Контекст	<code>http, server, location</code>

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.

### Осторожно

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 число 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	<code>http, server, location</code>

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	<code>http, server, location</code>

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	<code>http, server, location</code>

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	<code>http, server, location</code>

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	<code>http, server, location</code>

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:

<code>off</code>	запрещает сжатие для всех проксированных запросов, игнорируя остальные параметры;
<code>expired</code>	разрешает сжатие, если в заголовке ответа есть поле "Expires" со значением, запрещающим кэширование;
<code>no-cache</code>	разрешает сжатие, если в заголовке ответа есть поле "Cache-Control" с параметром "no-cache";
<code>no-store</code>	разрешает сжатие, если в заголовке ответа есть поле "Cache-Control" с параметром "no-store";
<code>private</code>	разрешает сжатие, если в заголовке ответа есть поле "Cache-Control" с параметром "private";
<code>no_last_modified</code>	разрешает сжатие, если в заголовке ответа нет поля "Last-Modified";
<code>no_etag</code>	разрешает сжатие, если в заголовке ответа нет поля "ETag";
<code>auth</code>	разрешает сжатие, если в заголовке запроса есть поле "Authorization";
<code>any</code>	разрешает сжатие для всех проксированных запросов.

## gzip\_types

Синтаксис	<code>gzip_types mime-mun ...;</code>
По умолчанию	<code>gzip_types text/html;</code>
Контекст	<code>http, server, location</code>

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	<code>http, server, location</code>

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".

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_gzip_static_module`. В пакетах и образах из наших репозиториях модуль включен в сборку.

## 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	http, server, location, if в location

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] <i>время</i>;</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.

<code>epoch</code>	задает время "Thu, 01 Jan 1970 00:00:01 GMT" (1 января 1970 00:00:01 GMT) для поля "Expires" и "no-cache" для поля "Cache-Control".
<code>max</code>	задает время "Thu, 31 Dec 2037 23:55:55 GMT" (31 декабря 2037 23:55:55 GMT) для поля "Expires" и 10 лет для поля "Cache-Control".
<code>off</code>	запрещает добавлять или менять поля "Expires" и "Cache-Control" 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;
```

## Image Filter

Фильтр для преобразования изображений в форматах JPEG, GIF, PNG и WebP.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_image_filter_module`.

В наших репозиториях модуль собран динамически и доступен отдельным пакетом *angie-module-image-filter* или *angie-pro-module-image-filter*.

### Важно

Для этого модуля необходима библиотека `libgd`. Рекомендуется использовать самую последнюю версию библиотеки.

Для преобразования изображений в формате WebP библиотека `libgd` должна быть собрана с поддержкой WebP.

### Пример конфигурации

```
location /img/ {
    proxy_pass http://backend;
    image_filter resize 150 100;
    image_filter rotate 90;
    error_page 415 = /empty;
}

location = /empty {
    empty_gif;
}
```

### Директивы

#### `image_filter`

Синтаксис	<code>image_filter off;</code> <code>image_filter test; image_filter size; image_filter rotate 90   180   270;</code> <code>image_filter resize <i>ширина высота</i>; image_filter crop <i>ширина высота</i>;</code>
По умолчанию	<code>image_filter off;</code>
Контекст	<code>location</code>

Задаёт тип преобразования изображения:

<code>off</code>		отключает обработку данным модулем во вложенном location.
<code>test</code>		проверяет, что ответ действительно является изображением в формате JPEG, GIF, PNG или WebP. В противном случае возвращается ошибка 415 (Unsupported Media Type).
<code>size</code>		<b>выдает информацию об изображении в формате JSON, например:</b> <pre>{ "img" : { "width": 100, "height": 100, "type": "gif" } }</pre> <b>В случае ошибки выдается</b> <pre>{ }</pre>
<code>rotate</code> <code>90 180 270</code>		поворачивает изображение против часовой стрелки на указанное число градусов. В значении параметра допустимо использование переменных. Можно использовать как отдельно, так и совместно с преобразованиями <i>resize</i> и <i>crop</i> .
<code>resize</code>	<i>ширина</i> <i>высота</i>	пропорционально уменьшает изображение до указанных размеров. Если требуется уменьшить только по одному измерению, то в качестве второго можно указать "-". В случае ошибки сервер возвращает код 415 (Unsupported Media Type). В значениях параметров допустимо использование переменных. При использовании совместно с <i>rotate</i> , поворот изображения происходит <b>после</b> уменьшения размеров изображения.
<code>crop</code>	<i>ширина</i> <i>вы-</i> <i>сота</i>	пропорционально уменьшает изображение до размера большей стороны и обрезает лишние края по другой стороне. Если требуется уменьшить только по одному измерению, то в качестве второго можно указать "-". В случае ошибки сервер возвращает код 415 (Unsupported Media Type). В значениях параметров допустимо использование переменных. При использовании совместно с <i>rotate</i> , поворот изображения происходит <b>до</b> уменьшения размеров изображения.

### image\_filter\_buffer

Синтаксис	<code>image_filter_buffer размер;</code>
По умолчанию	<code>image_filter_buffer 1M;</code>
Контекст	<code>http, server, location</code>

Задаёт максимальный размер буфера для чтения изображения. При превышении размера сервер вернет ошибку 415 (Unsupported Media Type).

### image\_filter\_interlace

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

Если включено, то итоговые изображения будут с чересстрочностью. В случае JPEG итоговые изображения будут в формате "progressive JPEG".



### image\_filter\_jpeg\_quality

Синтаксис	<code>image_filter_jpeg_quality <i>качество</i>;</code>
По умолчанию	<code>image_filter_jpeg_quality 75;</code>
Контекст	<code>http, server, location</code>

Задаёт желаемое качество преобразованного изображения в формате JPEG. Допустимые значения находятся в диапазоне от 1 до 100. Меньшим значениям обычно соответствует худшее качество изображения и меньший объём передаваемых данных. Максимальное рекомендуемое значение — 95. В значении параметра допустимо использование переменных.

### image\_filter\_sharpen

Синтаксис	<code>image_filter_sharpen <i>процент</i>;</code>
По умолчанию	<code>image_filter_sharpen 0;</code>
Контекст	<code>http, server, location</code>

Повышает резкость итогового изображения. Процент резкости может быть больше 100. Значение 0 отключает повышение резкости. В значении параметра допустимо использование переменных.

### image\_filter\_transparency

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

Определяет, сохранять ли прозрачность при обработке изображений в формате GIF и в формате PNG с цветами, заданными палитрой. Потеря прозрачности позволяет получить более качественное изображение. Прозрачность альфа-канала в формате PNG сохраняется всегда.

### image\_filter\_webp\_quality

Синтаксис	<code>image_filter_webp_quality <i>качество</i>;</code>
По умолчанию	<code>image_filter_webp_quality 80;</code>
Контекст	<code>http, server, location</code>

Задаёт желаемое качество преобразованного изображения в формате WebP. Допустимые значения находятся в диапазоне от 1 до 100. Меньшим значениям обычно соответствует худшее качество изображения и меньший объём передаваемых данных. В значении параметра допустимо использование переменных.

## 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 файл ...;</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'e. Например, в такой конфигурации:

```
location = / {
    index index.html;
}

location / {
#    ...
}
```

запрос "/" будет фактически обработан во втором location'e как `/index.html`.

## JS

Позволяет задавать обработчики на njs — подмножестве языка JavaScript.

В наших репозиториях модуль собран динамически и доступен отдельным пакетом `angie-module-njs` или `angie-pro-module-njs`; подключить его можно с помощью директивы `load_module`.

## Пример конфигурации

```
http {
    js_import http.js;

    js_set $foo      http.foo;
    js_set $summary http.summary;
    js_set $hash     http.hash;

    resolver 127.0.0.53;

    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://google.com/'),
    ngx.fetch('https://google.ru/')]);

  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

Синтаксис	js_body_filter <i>функция</i>   <i>модуль.функция</i> [buffer_type=строка   буфер];
По умолчанию	—
Контекст	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:

<b>r</b>	объект <code>HTTP request</code>
<b>data</b>	входящий блок данных может быть строкой или буфером в зависимости от значения <code>buffer_type</code> , по умолчанию является строкой.
<b>flags</b>	объект со следующими свойствами: <code>last</code> — логическое значение <code>true</code> — 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.

#### **i** 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

Синтаксис	<code>js_content функция   модуль.функция;</code>
По умолчанию	—
Контекст	<code>location, if in location, limit_except</code>

Задаёт функцию `пjs` в качестве обработчика содержимого `location`. Можно ссылаться на функцию модуля.

### js\_fetch\_buffer\_size

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

Задаёт размер буфера, который будет использоваться для чтения и записи для [Fetch API](#).

### js\_fetch\_ciphers

Синтаксис	<code>js_fetch_ciphers <i>шифры</i>;</code>
По умолчанию	<code>js_fetch_ciphers HIGH:!aNULL:!MD5;</code>
Контекст	<code>http, server, location</code>

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

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

Задаёт максимальный размер ответа, полученного при помощи [Fetch API](#).

### js\_fetch\_protocols

Синтаксис	<code>js_fetch_protocols [TLSv1] [TLSv1.1] [TLSv1.2] [TLSv1.3];</code>
По умолчанию	<code>js_fetch_protocols TLSv1 TLSv1.1 TLSv1.2;</code>
Контекст	<code>http, server, location</code>

Разрешает указанные протоколы для HTTPS-соединений при помощи [Fetch API](#).

### js\_fetch\_timeout

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

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

Синтаксис	<code>js_fetch_trusted_certificate <i>файл</i>;</code>
По умолчанию	—
Контекст	<code>http, server, location</code>

Задаёт файл с доверенными сертификатами CA в формате PEM, используемыми при проверке HTTPS-сертификата при помощи [Fetch API](#).

### js\_fetch\_verify

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

Разрешает или запрещает проверку сертификата HTTPS-сервера при помощи [Fetch API](#).

### js\_fetch\_verify\_depth

Синтаксис	<code>js_fetch_verify_depth <i>число</i>;</code>
По умолчанию	<code>js_fetch_verify_depth 100;</code>
Контекст	<code>http, server, location</code>

Устанавливает глубину проверки в цепочке HTTPS-сертификатов при помощи [Fetch API](#).

### js\_header\_filter

Синтаксис	<code>js_header_filter <i>функция</i>   <i>модуль.функция</i>;</code>
По умолчанию	—
Контекст	<code>location, if in location, limit_except</code>

Задаёт функцию `пjs` 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

Синтаксис	<code>js_import модуль.js   имя_экспорта from модуль.js;</code>
По умолчанию	—
Контекст	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

Синтаксис	<code>js_path путь;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт дополнительный путь для модулей pjs.

## js\_preload\_object

Синтаксис	<code>js_preload_object имя.json   имя from файл.json;</code>
По умолчанию	—
Контекст	http, server, location

Предварительно загружает неизменяемый объект во время конфигурации. *Имя* используется в качестве имени глобальной переменной, через которую объект доступен в коде pjs. 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

Синтаксис	<code>js_set \$переменная функция   модуль.функция;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт функцию `pjs` 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.

### 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

Синтаксис	<code>js_shared_dict_zone zone=<i>имя:размер</i> [timeout=<i>время</i>] [type=string   number] [evict];</code>
По умолчанию	—
Контекст	http

Задаёт *имя* и *размер* зоны разделяемой памяти, в которой хранится словарь ключей и значений, разделяемый между рабочими процессами.

<code>type</code>	необязательный параметр, позволяет изменить тип значения на числовой ( <code>number</code> ), по умолчанию в качестве ключа и значения используется строка ( <code>string</code> ).
<code>timeout</code>	необязательный параметр, задаёт время, по завершении которого все записи в словаре удаляются из зоны
<code>evict</code>	необязательный параметр, удаляет самую старую пару ключ-значение при переполнении зоны

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;
```

```
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.

### 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	<code>limit_conn зона число;</code>
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 SPDY 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	<code>limit_conn_dry_run on   off;</code>
По умолчанию	<code>limit_conn_dry_run off;</code>
Контекст	<code>http, server, location</code>

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 текст, переменные и их комбинации. 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=<i>название</i> [burst=<i>число</i>] [nodelay   delay=<i>число</i>];</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` задает 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 меньшим, 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 ключ zone=название:размер rate=скорость;</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 текст, переменные и их комбинации. 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*'а, 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	http, server, location, if в location, limit_except

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 predetermined формат *"combined"*.

Если задан размер буфера с помощью параметра *buffer* или указан параметр *gzip*, то запись будет буферизованной.

### Осторожно

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;
```

### Важно

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 такимилогами;
- 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	<code>log_format имя [escape=:samp:default   json   none ] строка ...;</code>
По умолчанию	<code>log_format combined "...";</code>
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 предопределенный 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	<code>open_log_file_cache max=N [inactive=время] [min_uses=N] [valid=время];</code> <code>open_log_file_cache off;</code>
По умолчанию	<code>open_log_file_cache off;</code>
Контекст	<code>http, server, location</code>

Задаёт кэш, в котором хранятся дескрипторы файлов часто используемых логов, имена которых заданы с использованием переменных. Параметры:

<b>max</b>	задаёт максимальное число дескрипторов в кэше; при переполнении кэша наименее востребованные (LRU) дескрипторы закрываются
<b>inactive</b>	задаёт время, после которого кэшированный дескриптор закрывается, если к нему не было обращений в течение этого времени; по умолчанию 10 секунд
<b>min_uses</b>	задаёт минимальное число использований файла в течение времени, заданного параметром <i>inactive</i> , после которого дескриптор файла будет оставаться открытым в кэше; по умолчанию 1
<b>valid</b>	задаёт, через какое время нужно проверять, что файл ещё существует под тем же именем; по умолчанию 60 секунд
<b>off</b>	запрещает кэш

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	map <i>строка</i> <i>\$переменная</i> { ... }
По умолчанию	—
Контекст	http

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 выделения, 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:

<b>default</b> <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.
<b>hostnames</b>	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;
```

<b>include</b> <i>файл</i>	включает файл со значениями. Включений может быть несколько.
<b>volatile</b>	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	map_hash_bucket_size <i>размер</i> ;
По умолчанию	map_hash_bucket_size 32 64 128;
Контекст	http

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 <i>размер</i> ;
По умолчанию	map_hash_max_size 2048;
Контекст	http

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	memcached_bind <i>адрес</i> [transparent]   off;
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 допустимо использование переменных. Специальное значение `off` отменяет действие унаследованной с предыдущего уровня конфигурации директивы `memcached_bind`, позволяя системе самостоятельно выбирать 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:

```
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.

#### Важно

Необходимо настроить таблицу маршрутизации ядра для перехвата сетевого трафика с сервера memcached.

### memcached\_buffer\_size

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

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 <i>время</i>;</code>
По умолчанию	<code>memcached_connect_timeout 60s;</code>
Контекст	<code>http, server, location</code>

Задаёт таймаут для установления соединения с сервером memcached. Необходимо иметь в виду, что этот таймаут обычно не может превышать 75 секунд.

### memcached\_gzip\_flag

Синтаксис	<code>memcached_gzip_flag <i>флаг</i>;</code>
По умолчанию	<code>—</code>
Контекст	<code>http, server, location</code>

Включает проверку указанного флага в ответе сервера memcached и установку поля "Content-Encoding" заголовка ответа в "gzip", если этот флаг установлен.

### memcached\_next\_upstream

Синтаксис	<code>memcached_next_upstream error   timeout   invalid_response   not_found   off ...;</code>
По умолчанию	<code>memcached_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_response</code>	сервер вернул пустой или неверный ответ;
<code>not_found</code>	сервер не нашел ответ;
<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</code>	<code>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 время;</code>
По умолчанию	<code>memcached_next_upstream_timeout 0;</code>
Контекст	<code>http, server, location</code>

Ограничивает время, в течение которого возможна передача запроса *следующему* серверу.

0	отключает это ограничение
---	---------------------------

### memcached\_next\_upstream\_tries

Синтаксис	memcached_next_upstream_tries <i>число</i> ;
По умолчанию	memcached_next_upstream_tries 0;
Контекст	http, server, location

Ограничивает число допустимых попыток для передачи запроса *следующему* серверу.

0	отключает это ограничение
---	---------------------------

### memcached\_pass

Синтаксис	memcached_pass <i>адрес</i> ;
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*.

### memcached\_read\_timeout

Синтаксис	memcached_read_timeout <i>время</i> ;
По умолчанию	memcached_read_timeout 60s;
Контекст	http, server, location

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	memcached_send_timeout <i>время</i> ;
По умолчанию	memcached_send_timeout 60s;
Контекст	http, server, location

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	memcached_socket_keepalive on   off;
По умолчанию	memcached_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>

## Встроенные переменные

\$memcached\_key

Задаёт ключ для получения ответа из сервера memcached.

### 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 "moov-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 отправляется непосредственно как статический ресурс. Некоторые плееры также поддерживают запросы с указанием диапазона запрашиваемых байт (byte-range requests), для них этот модуль не требуется.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_mp4_module`. В пакетах и образах из наших репозиториях модуль включен в сборку.

#### **Осторожно**

Если ранее использовался сторонний модуль mp4, следует его отключить.

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 <i>размер</i> ;
По умолчанию	mp4_buffer_size 512K;
Контекст	http, server, location

Задаёт начальный размер буфера, используемого при обработке MP4-файлов.

### mp4\_max\_buffer\_size

Синтаксис	mp4_max_buffer_size <i>размер</i> ;
По умолчанию	mp4_max_buffer_size 10M;
Контекст	http, server, location

В ходе обработки метаданных может понадобиться буфер большего размера. Его размер не может превышать указанного, иначе Angie вернет 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\_limit\_rate

Синтаксис	mp4_limit_rate on   off   <i>коэффициент</i> ;
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	mp4_limit_rate_after <i>время</i> ;
По умолчанию	mp4_limit_rate_after 60s;
Контекст	http, server, location

Задаёт (в пересчёте на время воспроизведения) объём медиаданных, после передачи которого скорость будет ограничена директивой *mp4\_limit\_rate*.

## mp4\_start\_key\_frame

Синтаксис	mp4_start_key_frame on   off;
По умолчанию	mp4_start_key_frame off;
Контекст	http, server, location

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-листа`. Edit-листы поддерживаются большинством плееров и браузеров включая Chrome, Safari, QuickTime и ffmpeg, частично поддерживаются в Firefox.

## Perl

Модуль позволяет писать обработчики `location` и переменных на Perl, а также вставлять вызовы Perl в SSI.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_perl_module`.

В наших репозиториях модуль собран динамически и доступен отдельным пакетом `angie-module-perl` или `angie-pro-module-perl`; подключить его можно с помощью директивы *load\_module*.

### Важно

Для сборки этого модуля необходим Perl версии 5.6.1 и выше. Компилятор C должен быть совместим с тем, которым был собран Perl.

### Известные проблемы

Модуль экспериментальный, поэтому возможно все.

Для того чтобы во время переконфигурации Perl перекомпилировал измененные модули, его нужно собрать с параметрами `-Dusemultiplicity=yes` или `-Dusethreads=yes`. Кроме того, чтобы во время работы Perl терял меньше памяти, его нужно собрать с параметром `-Dusemymalloc=no`. Узнать значения этих параметров у уже собранного Perl можно так (в примере приведены желаемые значения параметров):

```
$ perl -V:usemultiplicity -V:usemymalloc |br|
usemultiplicity='define'; |br|
usemymalloc='n';
```

Необходимо учитывать, что после пересборки Perl с новыми параметрами `-Dusemultiplicity=yes` или `-Dusethreads=yes` придется также пересобрать и все бинарные модули Perl, так как они просто перестанут работать с новым Perl.

Возможно, главный процесс, а вслед за ним и рабочие процессы, будут увеличиваться в размерах при каждой переконфигурации. Когда главный процесс вырастет до неприемлемых размеров, можно воспользоваться процедурой *обновления сервера на лету*, не меняя при этом сам исполняемый файл.

Если модуль Perl выполняет длительную операцию, например, определяет адрес по имени, соединяется с другим сервером, делает запрос к базе данных, то на это время все остальные запросы, обслуживаемые данным рабочим процессом, не будут обрабатываться. Поэтому рекомендуется ограничиться операциями, время исполнения которых короткое и предсказуемое, например, обращение к локальной файловой системе.

### Пример конфигурации

```
http {

    perl_modules perl/lib;
    perl_require hello.pm;

    perl_set $msie6 '

        sub {
            my $r = shift;
            my $ua = $r->header_in("User-Agent");

            return "" if $ua =~ /Opera/;
            return "1" if $ua =~ / MSIE [6-9]\.\d+/;
            return "";
        }

    ';

    server {
        location / {
```

```

        perl hello::handler;
    }
}

```

Модуль `perl/lib/hello.pm`:

```

package hello;

use nginx;

sub handler {
    my $r = shift;

    $r->send_http_header("text/html");
    return OK if $r->header_only;

    $r->print("hello!\n<br/>");

    if (-f $r->filename or -d _) {
        $r->print($r->uri, " exists!\n");
    }

    return OK;
}

1;
__END__

```

## Директивы

### perl

Синтаксис	<code>perl <i>модуль</i> :: <i>функция</i>   'sub { ... }'</code> ;
По умолчанию	—
Контекст	location, limit_except

Устанавливает обработчик Perl для данного location.

### perl\_modules

Синтаксис	<code>perl_modules <i>путь</i></code> ;
По умолчанию	—
Контекст	http

Задаёт дополнительный путь для модулей Perl.

## perl\_require

Синтаксис	<code>perl_require <i>модуль</i>;</code>
По умолчанию	—
Контекст	http

Задаёт имя модуля, который будет подгружаться при каждой переконфигурации. Директив `perl_require` может быть несколько.

## perl\_set

Синтаксис	<code>perl_set <i>\$переменная модуль</i> :: <i>функция</i>   'sub { ... }';</code>
По умолчанию	—
Контекст	http

Устанавливает обработчик Perl для указанной переменной.

## Вызов Perl из SSI

Формат команды SSI с вызовом Perl следующий:

```
<!--# perl sub="модуль:функция" arg="параметр1" arg="параметр2" ...
-->
```

## Методы объекта запроса \$r

`$r->args`

возвращает аргументы запроса.

`$r->filename`

возвращает имя файла, соответствующее URI запроса.

`$r->has_request_body` (**обработчик**)

возвращает 0, если в запросе нет тела. Если же в запросе есть тело, то устанавливается указанный обработчик и возвращается 1. По окончании чтения тела запроса Angie вызовет установленный обработчик. Обратите внимание, что нужно передавать ссылку на функцию обработчика. Пример:

```
package hello;

use nginx;

sub handler {
    my $r = shift;
```



```

if ($r->request_method ne "POST") {
    return DECLINED;
}

if ($r->has_request_body(\&post)) {
    return OK;
}

return HTTP_BAD_REQUEST;
}

sub post {
    my $r = shift;

    $r->send_http_header;

    $r->print("request_body: \", $r->request_body, "\"<br/>");
    $r->print("request_body_file: \", $r->request_body_file, "\"<br/>\n");

    return OK;
}

1;

__END__

```

`$r->allow_ranges`

разрешает использовать диапазоны байт (byte ranges) при передаче ответа.

`$r->discard_request_body`

указывает Angie игнорировать тело запроса.

`$r->header_in` (*поле*)

возвращает значение заданного поля в заголовке запроса клиента.

`$r->header_only`

определяет, нужно ли передавать клиенту только заголовок ответа или весь ответ.

`$r->header_out` (*поле, значение*)

устанавливает значение для заданного поля в заголовке ответа.

`$r->internal_redirect` (*uri*)

делает внутреннее перенаправление на указанный *uri*. Перенаправление происходит уже после завершения обработчика Perl. Метод принимает экранированные URI и поддерживает перенаправления в *именованные location*.

`$r->log_error` (*код\_ошибки, сообщение*)

записывает указанное сообщение в *error\_log*. Если *код\_ошибки* ненулевой, то к сообщению будет добавлен код ошибки и ее описание.

`$r->print` (*текст, ...*)

метод передает клиенту данные.

`$r->request_body`

возвращает тело запроса клиента при условии, что тело не записано во временный файл. Для того чтобы тело запроса клиента гарантированно находилось в памяти, нужно ограничить его размер с помощью *client\_max\_body\_size* и задать достаточной размер для буфера *client\_body\_buffer\_size*.

`$r->request_body_file`

возвращает имя файла, в котором хранится тело запроса клиента. По завершению обработки файл необходимо удалить. Для того чтобы тело запроса клиента всегда записывалось в файл, следует включить *client\_body\_in\_file\_only*.

`$r->request_method`

возвращает HTTP-метод запроса клиента.

`$r->remote_addr`

возвращает IP-адрес клиента.

`$r->flush`

немедленно передает данные клиенту.

`$r->sendfile (имя [, смещение [, длина ]])`

передает клиенту содержимое указанного файла. Необязательные параметры задают начальное смещение и длину передаваемых данных. Непосредственно передача данных происходит уже после завершения обработчика Perl.

`$r->send_http_header ([тип])`

передает клиенту заголовок ответа. Необязательный параметр тип устанавливает значение поля "Content-Type" в заголовке ответа. Пустая строка в качестве типа запрещает передачу поля "Content-Type".

`$r->status (код)`

устанавливает код ответа.

`$r->sleep (миллисекунды, обработчик)`

устанавливает указанный обработчик и останавливает обработку запроса на заданное время. Angie в это время продолжает обрабатывать другие запросы. По истечении указанного времени Angie вызовет установленный обработчик. Обратите внимание, что нужно передавать ссылку на функцию обработчика. Для передачи данных между обработчиками следует использовать `$r->variable()`.  
Пример:

```
package hello;

use nginx;

sub handler {
    my $r = shift;

    $r->discard_request_body;
    $r->variable("var", "OK");
    $r->sleep(1000, \&next);

    return OK;
}

sub next {
    my $r = shift;

    $r->send_http_header;
    $r->print($r->variable("var"));

    return OK;
}

1;

__END__
```

`$r->unescape` (*текст*)

декодирует текст, заданный в виде "%XX".

`$r->uri`

возвращает URI запроса.

`$r->variable` (*имя* [, *значение* ])

возвращает или устанавливает значение указанной переменной. Переменные локальны для каждого запроса.

## Prometheus

Собирает *статистику* Angie, исходя из определенных в конфигурации шаблонов, и возвращает сформированные на основании этих шаблонов метрики в формате Prometheus.

### Внимание

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 входит вспомогательный файл `prometheus_all.conf`, куда включен набор общепотребимых метрик, сведенных в шаблон `all`:

### Содержимое файла (Angie)

```
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_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, or 4 - recovering.';

'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_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_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_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_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, or 4 - recovering.';

'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_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".';
}

map $p8s_value $p8st_all_ups_state {
  volatile;
  "up"          1;
  "down"        2;
  "unavailable" 3;
  "recovering" 4;
  default       0;
}
```

### Содержимое файла (Angie PRO)

```
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_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, or 7 - draining.';

'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_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_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.';
```

```

↪result.';

'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_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, or 7 - draining.';

'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.';
}

map $p8s_value $p8st_all_ups_state {
volatile;
"up"          1;
"down"        2;
"unavailable" 3;
"recovering"  4;
"unhealthy"   5;
"checking"    6;
"draining"    7;
default      0;
}

```

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	<code>location</code>

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>];`

<b>имя_метрики</b>	<p>Задаёт имя метрики, под которым она будет добавлена в формате Prometheus в ответ на запрос. Может содержать необязательную часть с метками (...), например:</p> <pre>http_requests_total{method="\$1",code="\$2"}</pre> <p>В значениях меток можно использовать переменные Angie; если <code>строка_сопоставления</code> задана регулярным выражением, то можно также использовать определенные в этом выражении группы захвата. Такие переменные и группы вычисляются при получении значения метрики, которое задает <code>переменная</code>.</p>
<b>переменная</b>	<p>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.</p>

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="1.8.2"} 1695119820.562
```

<b>path=строка_соп</b>	Сопоставляется со всеми конечными путями метрик в поддереве <code>/status</code> API-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:

<b>path=точное_соот.</b>	Проверяется посимвольным сравнением.
<b>path=~регулярное</b>	Проверяется при помощи библиотеки PCRE; может задавать группы захвата для использования в метках поля <code>имя метрики</code> .

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*.

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.

## Прoxy

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;
}
```

## 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 допустимо использование переменных. Специальное значение `off` отменяет действие унаследованной с предыдущего уровня конфигурации директивы `proxy_bind`, позволяя системе самостоятельно выбирать 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:

```
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.

#### Важно

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>

<code>on</code>	Angie принимает ответ проксируемого сервера как можно быстрее, сохраняя его в буферы, заданные директивами <code>proxy_buffer_size</code> и <code>proxy_buffers</code> . Если ответ не вмещается целиком в память, то его часть может быть записана на диск во <i>временный файл</i> . Запись во временные файлы контролируется директивами <code>proxy_max_temp_file_size</code> и <code>proxy_temp_file_write_size</code> .
<code>off</code>	Ответ синхронно передается клиенту сразу же по мере его поступления. Angie не пытается считать весь ответ проксируемого сервера. Максимальный размер данных, который Angie может принять от сервера за один раз, задается директивой <code>proxy_buffer_size</code> .

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 4К, или 8К.

### proxy\_busy\_buffers\_size

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

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	proxy_cache <i>зона</i>   off [path= <i>путь</i> ];
По умолчанию	proxy_cache off;
Контекст	http, server, location

Задаёт зону разделяемой памяти для кэширования. Одну и ту же зону можно использовать в нескольких местах конфигурации. В значении параметра допускаются переменные.

off	запрещает кэширование, унаследованное с предыдущего уровня конфигурации.
-----	--

Добавлено в версии 1.2.0: PRO

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*:

<i>path=путь</i>	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;
}
}

```

Здесь есть два пути кэша и отображение переменной, чтобы их различать. Если Content-Type начинается с text/, будет выбран первый путь, иначе — второй.

### proxy\_cache\_background\_update

Синтаксис	proxy_cache_background_update on   off;
По умолчанию	proxy_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.

#### **Внимание**

Использование устаревшего кэшированного ответа в момент его обновления должно быть *разрешено*.

### proxy\_cache\_bypass

Синтаксис	proxy_cache_bypass ...;
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;

```

Можно использовать совместно с директивой *proxy\_no\_cache*.

### proxy\_cache\_convert\_head

Синтаксис	proxy_cache_convert_head on   off;
По умолчанию	proxy_cache_convert_head on;
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, 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	<code>proxy_cache_lock_age время;</code>
По умолчанию	<code>proxy_cache_lock_age 5s;</code>
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	<code>proxy_cache_lock_timeout время;</code>
По умолчанию	<code>proxy_cache_lock_timeout 5s;</code>
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.

### proxy\_cache\_path

Синтаксис	proxy_cache_path <i>путь</i> [levels= <i>уровни</i> ] [use_temp_path=on   off] keys_zone= <i>имя:размер</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> ];
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>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. Зоны размером в 1 мегабайт достаточно для хранения около 8000 ключей.
<code>inactive</code>	Если к данным кэша не обращаются в течение времени, заданного этим параметром, то данные удаляются, независимо от их свежести. По умолчанию <code>inactive</code> равен 10 минутам.

#### Примечание

Добавлено в версии 1.2.0: PRO

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.

<code>max_size</code>	максимальное пороговое значение размера кэша
<code>min_free</code>	минимальное пороговое значение объема свободного места на файловой системе с кэшем
<code>manager_files</code>	максимальное количество удаляемых элементов кэша за одну итерацию По умолчанию: 100.
<code>manager_threshold</code>	ограничивает время работы одной итерации По умолчанию: 200 миллисекунд.
<code>manager_sleep</code>	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.

<code>loader_files</code>	максимальное количество элементов кэша к загрузке в одну итерацию По умолчанию: 100
<code>loader_threshold</code>	ограничивает время работы одной итерации По умолчанию: 200 миллисекунд
<code>loader_sleep</code>	время, в течение которого выдерживается пауза между итерациями По умолчанию: 50 миллисекунд

### proxy\_cache\_revalidate

Синтаксис	<code>proxy_cache_revalidate on   off;</code>
По умолчанию	<code>proxy_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".

### 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   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`.

<code>error</code>	Позволяет использовать устаревший кэшированный ответ при невозможности выбора проксируемого сервера для обработки запроса.
<code>updating</code>	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.

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	proxy_cache_valid [код ...] время;
По умолчанию	—
Контекст	http, server, location

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" задаёт время кэширования ответа в секундах. Значение 0 запрещает кэшировать ответ. Если значение начинается с префикса @, оно задаёт 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 проксируемым сервером. Необходимо иметь в виду, что этот таймаут обычно не может превышать 75 секунд.

### 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 соединений с проксируемым сервером, если он был удален из группы или помечен как постоянно недоступный в результате процесса *resolve* или команды *API DELETE*.

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 выделения, 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	http, server, location

Задаёт один или несколько флагов для cookie. В качестве cookie можно использовать текст, переменные и их комбинации. В качестве флага можно использовать текст, переменные и их комбинации.

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 ~ nosecure 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 путь замена;</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 выделения, 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	<code>proxy_force_ranges off;</code>
По умолчанию	<code>proxy_force_ranges off;</code>
Контекст	http, server, location

Включает поддержку диапазонов запрашиваемых байт (byte-range) для кэшированных и некашированных ответов проксируемого сервера вне зависимости от наличия поля "Accept-Ranges" в заголовках этих ответов.

### proxy\_headers\_hash\_bucket\_size

Синтаксис	<code>proxy_headers_hash_bucket_size <i>размер</i>;</code>
По умолчанию	<code>proxy_headers_hash_bucket_size 64;</code>
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	<code>proxy_headers_hash_max_size размер;</code>
По умолчанию	<code>proxy_headers_hash_max_size 512;</code>
Контекст	<code>http, server, location</code>

Задаёт максимальный размер хэш-таблиц, используемых директивами `proxy_hide_header` и `proxy_set_header`. Подробнее настройка хэш-таблиц обсуждается отдельно.

### proxy\_hide\_header

Синтаксис	<code>proxy_hide_header поле;</code>
По умолчанию	—
Контекст	<code>http, server, location</code>

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\_http\_version

Синтаксис	<code>proxy_http_version 1.0   1.1   3;</code>
По умолчанию	<code>proxy_http_version 1.0;</code>
Контекст	<code>http, server, location, if в location, limit_except</code>

Задаёт версию протокола HTTP для проксирования. По умолчанию используется версия 1.0. Для работы *постоянных соединений* рекомендуется версия 1.1 или выше.

### proxy\_http3\_hq

Синтаксис	<code>proxy_http3_hq on   off;</code>
По умолчанию	<code>proxy_http3_hq off;</code>
Контекст	<code>http, server</code>

Отключает или включает особый режим согласования `hq-interop`, используемый в функциональных тестах *QUIC*, на которые полагается Angie.

### 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 <i>размер</i> ;
По умолчанию	proxy_http3_stream_buffer_size 64k;
Контекст	http, server

Задаёт размер буфера, используемого для чтения и записи *QUIC-потоков*.

### proxy\_ignore\_client\_abort

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

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\_limit\_rate

Синтаксис	<code>proxy_limit_rate скорость;</code>
По умолчанию	<code>proxy_limit_rate 0;</code>
Контекст	<code>http, server, location</code>

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 <i>размер</i> ;
По умолчанию	proxy_max_temp_file_size 1024m;
Контекст	http, server, location

Если включена *буферизация* ответов проксируемого сервера, и ответ не вмещается 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 допустимо использование переменных.

### 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 <b>неидемпотентным</b> 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_502	
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 <i>время</i> ;
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	<code>proxy_next_upstream_tries число;</code>
По умолчанию	<code>proxy_next_upstream_tries 0;</code>
Контекст	<code>http, server, location</code>

Ограничивает число допустимых попыток для передачи запроса *следующему* серверу.

0	отключает это ограничение
---	---------------------------

### proxy\_no\_cache

Синтаксис	<code>proxy_no_cache строка ...;</code>
По умолчанию	—
Контекст	<code>http, server, location</code>

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;
```

Можно использовать совместно с директивой `proxy_cache_bypass`.

### proxy\_pass

Синтаксис	<code>proxy_pass uri;</code>
По умолчанию	—
Контекст	<code>location, if в location, limit_except</code>

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.

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`'ах.

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*.

### proxy\_pass\_header

Синтаксис	<code>proxy_pass_header поле ...;</code>
По умолчанию	—
Контекст	<code>http, server, location</code>

Разрешает передавать от проксируемого сервера клиенту *запрещенные для передачи* поля заголовка.

### proxy\_pass\_request\_body

Синтаксис	proxy_pass_request_body on   off;
По умолчанию	proxy_pass_request_body on;
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	proxy_pass_request_headers on   off;
По умолчанию	proxy_pass_request_headers on;
Контекст	http, server, location

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	proxy_pass_trailers on   off;
По умолчанию	proxy_pass_trailers off;
Контекст	http, server, location

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	proxy_quic_active_connection_id_limit <i>число</i> ;
По умолчанию	proxy_quic_active_connection_id_limit 2;
Контекст	http, server

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 <i>время</i> ;
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	<code>proxy_redirect default;</code> <code>proxy_redirect off;</code> <code>proxy_redirect <i>перенаправление замена</i>;</code>
По умолчанию	<code>proxy_redirect default;</code>
Контекст	<code>http, server, location</code>

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/;
```

### Осторожно

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 выделения, 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.

<b>on</b>	тело запроса полностью <i>читается</i> от клиента перед отправкой запроса на проксируемый сервер.
<b>off</b>	тело запроса отправляется на проксируемый сервер сразу же по мере его поступления. В этом случае запрос не может быть передан <i>следующему серверу</i> , 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 также не *включен* HTTP/1.1.

### 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.

**i Примечание**

Эта директива игнорируется на Linux, Solaris и Windows.

### proxy\_send\_timeout

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

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</code> <i>значение</i> ;
По умолчанию	—
Контекст	http, server, location

Позволяет переопределить тело запроса, передаваемое на проксируемый сервер. В качестве значения можно использовать текст, переменные и их комбинации.

### proxy\_set\_header

Синтаксис	<code>proxy_set_header</code> <i>поле значение</i> ;
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 текст, переменные и их комбинации. 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.

Неизменное 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	proxy_socket_keepalive on   off;
По умолчанию	proxy_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>

### proxy\_ssl\_certificate

Синтаксис	proxy_ssl_certificate <i>файл</i> [ <i>файл</i> ];
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.

Добавлено в версии 1.2.0.

При включенном *proxy\_ssl\_nls* директива принимает два аргумента вместо одного:

```
location /proxy {
    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 https://backend:443;
}
```



### 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:*имя*:id", которое загружает ключ с указанным *id* из OpenSSL engine с заданным именем.

В имени файла можно использовать переменные.

Добавлено в версии 1.2.0.

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`.

### proxy\_ssl\_conf\_command

Синтаксис	proxy_ssl_conf_command <i>имя значение</i> ;
По умолчанию	—
Контекст	http, server, location

Задаёт произвольные конфигурационные команды OpenSSL при установлении соединения с проксируемым HTTPS-сервером.

### Важно

Директива поддерживается при использовании OpenSSL 1.0.2 и выше.

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`.

### Осторожно

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\_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.

По умолчанию используется имя хоста из URL'а, заданного директивой `proxy_pass`.

## proxy\_ssl\_ntls

Добавлено в версии 1.2.0.

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 `TongSuo`.

```
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;
}
```

### Важно

Angie необходимо собрать с использованием параметра конфигурации `--with-ntls`, с соответствующей SSL библиотекой с поддержкой NTLS

```
./configure --with-openssl=../Tongsuo-8.3.0 \
            --with-openssl-opt=enable-ntls \
            --with-ntls
```

### proxy\_ssl\_password\_file

Синтаксис	<code>proxy_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.

### 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	http, server, location

Изменено в версии 1.2.0: Параметр TLSv1.3 добавлен к используемым по умолчанию.

Разрешает указанные протоколы для запросов к проксируемому HTTPS-серверу.

### proxy\_ssl\_server\_name

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

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	proxy_ssl_session_reuse on   off;
По умолчанию	proxy_ssl_session_reuse on;
Контекст	http, server, location

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	http, server, location

Задаёт файл с доверенными сертификатами CA 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	http, server, location

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 <i>пользователи:права ...</i>;</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\_temp\_file\_write\_size

Синтаксис	<code>proxy_temp_file_write_size <i>размер</i>;</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 ограничен 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 трёх 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_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;

`$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`.

## Random Index

Обслуживает запросы, оканчивающиеся косой чертой (/), и выдает случайный файл в качестве индексного файла каталога. Модуль выполняется до модуля `http_index`.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_random_index_module`. В пакетах и образах из наших репозиториях модуль включен в сборку.

### Пример конфигурации

```
location / {
    random_index on;
}
```

### Директивы

#### random\_index

Синтаксис	<code>random_index on   off;</code>
По умолчанию	<code>random_index off;</code>
Контекст	location

Разрешает или запрещает в содержащем *location* обработку этим модулем.

### RealIP

Позволяет менять адрес и необязательный порт клиента на переданные в указанном поле заголовка.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_realip_module`. В пакетах и образах из наших репозиториев модуль включен в сборку.

### 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

#### set\_real\_ip\_from

Синтаксис	<code>set_real_ip_from адрес   CIDR   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.



## real\_ip\_header

Синтаксис	<code>real_ip_header поле   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.

## 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 <i>размер</i>;</code>
По умолчанию	<code>referer_hash_bucket_size 64;</code>
Контекст	<code>server, location</code>

Задаёт размер корзины хэш-таблиц со значениями "Referer". Подробнее настройка хэш-таблиц обсуждается отдельно.

### referer\_hash\_max\_size

Синтаксис	<code>referer_hash_max_size <i>размер</i>;</code>
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   <i>строка</i> ...;</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 учета регистра символов.

Параметры могут быть следующие:

<b>none</b>	поле "Referer" в заголовке запроса отсутствует;
<b>blocked</b>	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://";
<b>server_names</b>	в поле "Referer" заголовка запроса указано одно из имен сервера;
<b>произвольная строка</b>	задает имя сервера и необязательное начало URI. В начале или конце имени сервера может быть "*". При проверке порт сервера в поле "Referer" игнорируется;
<b>регулярное выражение</b>	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*, *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, location, 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;
}
```

## 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.

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 выделения, 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;
}
```

### **i** 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.

Необязательный параметр **флаг** может быть одним из:

<code>last</code>	завершает обработку текущего набора директив модуля <code>http_rewrite</code> , после чего ищется новый <code>location</code> , соответствующий измененному URI;
<code>break</code>	завершает обработку текущего набора директив модуля <code>http_rewrite</code> аналогично директиве <code>break</code> ;
<code>redirect</code>	возвращает временное перенаправление с кодом 302; используется, если строка <b>замены</b> не начинается с " <code>http://</code> ", " <code>https://</code> " или " <code>\$scheme</code> ";
<code>permanent</code>	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	<code>http, server, location, if</code>

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	<code>server, location, if</code>

Устанавливает значение указанной переменной. В качестве значения можно использовать текст, переменные и их комбинации.

## uninitialized\_variable\_warn

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

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 адрес [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 допустимо использование переменных. Специальное значение `off` отменяет действие унаследованной с предыдущего уровня конфигурации директивы `scgi_bind`, позволяя системе самостоятельно выбирать локальный IP-адрес и порт.

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 `superпользователя`. В Linux это не требуется, так как если указан параметр `transparent`, то рабочие процессы наследуют `capability CAP_NET_RAW` из главного процесса.

#### Важно

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	<code>http, server, location</code>

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 размер;</code>
По умолчанию	<code>scgi_buffering on;</code>
Контекст	<code>http, server, location</code>

<code>on</code>	Angie принимает ответ SCGI-сервера как можно быстрее, сохраняя его в буферы, заданные директивами <code>scgi_buffer_size</code> и <code>scgi_buffers</code> . Если ответ не помещается целиком в память, то его часть может быть записана на диск во <i>временный файл</i> . Запись во временные файлы контролируется директивами <code>scgi_max_temp_file_size</code> и <code>scgi_temp_file_write_size</code> .
<code>off</code>	Ответ синхронно передается клиенту сразу же по мере его поступления. Angie не пытается считать весь ответ SCGI-сервера. Максимальный размер данных, который Angie может принять от сервера за один раз, задается директивой <code>scgi_buffer_size</code> .

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

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	<code>scgi_cache зона   off;</code>
По умолчанию	<code>scgi_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.
------------------	--

### scgi\_cache\_background\_update

Синтаксис	<code>scgi_cache_background_update on   off;</code>
По умолчанию	<code>scgi_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.

#### **Внимание**

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;
```

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, 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 ещё один запрос.

### scgi\_cache\_lock\_timeout

Синтаксис	<code>scgi_cache_lock_timeout</code> <i>время</i> ;
По умолчанию	<code>scgi_cache_lock_timeout 5s</code> ;
Контекст	<code>http, server, location</code>

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	<code>http, server, location</code>

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	<code>http, server, location</code>

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	<code>http, server, location</code>

Задаёт число запросов, после которого ответ будет кэширован.

## 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>] [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*.

Параметр `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.

Какой из каталогов будет использоваться для временных файлов, определяется параметром `use_temp_path`.

<code>on</code>	Если параметр не задан или установлен в значение "on", 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

Кроме того, все активные ключи и информация о данных хранятся в зоне разделяемой памяти, имя и размер которой задаются параметром `keys_zone`. Зоны размером в 1 мегабайт достаточно для хранения около 8 тысяч ключей.

Если к данным кэша не обращаются в течение времени, заданного параметром `inactive`, то данные удаляются, независимо от их свежести.

По умолчанию `inactive` 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.

<code>max_size</code>	максимальное пороговое значение размера кэша
<code>min_free</code>	минимальное пороговое значение объема свободного места на файловой системе с кэшем
<code>manager_files</code>	максимальное количество удаляемых элементов кэша за одну итерацию По умолчанию: 100
<code>manager_threshold</code>	ограничивает время работы одной итерации По умолчанию: 200 миллисекунд
<code>manager_sleep</code>	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.

<code>loader_files</code>	максимальное количество элементов кэша к загрузке в одну итерацию По умолчанию: 100
<code>loader_threshold</code>	ограничивает время работы одной итерации По умолчанию: 200 миллисекунд
<code>loader_sleep</code>	время, в течение которого выдерживается пауза между итерациями По умолчанию: 50 миллисекунд

### scgi\_cache\_revalidate

Синтаксис	<code>scgi_cache_revalidate on   off;</code>
По умолчанию	<code>scgi_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".

### scgi\_cache\_use\_stale

Синтаксис	<code>scgi_cache_use_stale error   timeout   invalid_header   updating   http_500   http_503   http_403   http_404   http_429   off ...;</code>
По умолчанию	<code>scgi_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 `scgi_next_upstream`.

<code>error</code>	Позволяет использовать устаревший кэшированный ответ при невозможности выбора SCGI-сервера для обработки запроса.
<code>updating</code>	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.

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	<code>http, server, location</code>

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" задаёт время кэширования ответа в секундах. Значение 0 запрещает кэшировать ответ. Если значение начинается с префикса @, оно задаёт 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 <i>время</i>;</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 <i>время</i>   on   off;</code>
По умолчанию	<code>scgi_connection_drop off;</code>
Контекст	<code>http, server, location</code>

Настраивает завершение всех соединений с проксируемым сервером, если он был удален из группы или помечен как постоянно недоступный в результате процесса `eresolve` или команды `API DELETE`.

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 некэшированных 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	<code>http, server, location</code>

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	<code>http, server, location</code>

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	<code>http, server, location</code>

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	<code>http, server, location</code>

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\_limit\_rate

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

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 <i>размер</i>;</code>
По умолчанию	<code>scgi_max_temp_file_size 1024m;</code>
Контекст	<code>http, server, location</code>

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	<code>scgi_next_upstream error   timeout   invalid_header   http_500   http_503   http_403   http_404   http_429   non_idempotent   off ...;</code>
По умолчанию	<code>scgi_next_upstream error timeout;</code>
Контекст	<code>http, server, location</code>

Определяет, в каких случаях запрос будет передан следующему 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_503</code>	сервер вернул ответ с кодом 503;
<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 <b>неидемпотентным</b> 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</code>	всегда считаются неудачными попытками, даже если они не указаны в директиве
<code>timeout</code>	
<code>invalid_header</code>	
<code>http_500</code>	считаются неудачными попытками, только если они указаны в директиве
<code>http_503</code>	
<code>http_429</code>	
<code>http_403</code>	никогда не считаются неудачными попытками
<code>http_404</code>	

Передача запроса следующему серверу может быть ограничена по *количеству попыток* и по *времени*.

### scgi\_next\_upstream\_timeout

Синтаксис	<code>scgi_next_upstream_timeout</code> <i>время</i> ;
По умолчанию	<code>scgi_next_upstream_timeout 0</code> ;
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</code> <i>число</i> ;
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</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:

```
scgi_no_cache $cookie_nocache $arg_nocache$arg_comment;
scgi_no_cache $http_pragma $http_authorization;
```

Можно использовать совместно с директивой `scgi_cache_bypass`.

### scgi\_param

Синтаксис	<code>scgi_param</code> <i>параметр значение</i> [if_not_empty];
По умолчанию	—
Контекст	http, server, location

Задаёт параметр, который будет передаваться SCGI-серверу. В качестве значения можно использовать текст, переменные и их комбинации. 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;
    # ...
}
```

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.

### 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	scgi_pass_request_body on   off;
По умолчанию	scgi_pass_request_body on;
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	scgi_pass_request_headers on   off;
По умолчанию	scgi_pass_request_headers on;
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	scgi_read_timeout <i>время</i> ;
По умолчанию	scgi_read_timeout 60s;
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	scgi_request_buffering on   off;
По умолчанию	scgi_request_buffering on;
Контекст	http, server, location

Разрешает или запрещает использовать буферизацию тела запроса клиента.

on	тело запроса полностью <i>читается</i> 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 <i>следующему серверу</i> , 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</code> <i>время</i> ;
По умолчанию	<code>scgi_send_timeout 60s</code> ;
Контекст	<code>http, server, location</code>

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.

<code>""</code>	По умолчанию для сокета действуют настройки операционной системы.
<code>on</code>	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.

<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:

```
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	<code>scgi_store_access <i>пользователи:права ...</i>;</code>
По умолчанию	<code>scgi_store_access user:rw;</code>
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\_temp\_file\_write\_size

Синтаксис	<code>scgi_temp_file_write_size <i>размер</i>;</code>
По умолчанию	<code>scgi_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 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	<code>scgi_temp_path путь [уровень1 [уровень2 [уровень3]]]`;</code>
По умолчанию	<code>scgi_temp_path scgi_temp;</code> (путь зависит от параметра сборки <code>--http-scgi-temp-path</code> )
Контекст	http, server, location

Задаёт имя каталога для хранения временных файлов с данными, полученными от SCGI-серверов. В каталоге может использоваться иерархия подкаталогов до трёх 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.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http-secure-link-module`. В пакетах и образах из наших репозиториях модуль включен в сборку.

## Директивы

### secure\_link

Синтаксис	<code>secure_link выражение;</code>
По умолчанию	—
Контекст	http, server, location

Задаёт строку с переменными, из которой будет выделено значение контрольной суммы и время 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-хэш передается в запросе закодированным в `base64url`.

Если ссылка имеет ограниченный срок действия, время ее действия 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 действия ссылки, переданное в запросе, делается доступным в переменной `$secure_link_expires` для использования в директиве `secure_link_md5`. Если время действия ссылки не задано, ссылка имеет неограниченный срок действия.

### secure\_link\_md5

Синтаксис	<code>secure_link_md5</code> <i>выражение</i> ;
По умолчанию	—
Контекст	<code>http, server, location</code>

Задаёт выражение, для которого считается значение MD5-хэш, сравниваемое с переданным в запросе.

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 слово;</code>
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 действия ссылки, 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.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_slice_module`. В пакетах и образах из наших репозиториях модуль включен в сборку.

## 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 <i>размер</i>;</code>
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 строка \$переменная { ... }</code>
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</code> в <code>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 <i>размер</i>;</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-<i>mun</i> ...;</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 <i>длина</i>;</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 выделения, 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 продолжить 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\_local

текущее время в локальной временной зоне. Формат задается командой *config* с параметром *timefmt*.

\$date\_gmt

текущее время в GMT. Формат задается командой *config* с параметром *timefmt*.

### SSL

Обеспечивает работу по протоколу HTTPS.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_ssl_module`. В пакетах и образах из наших репозиториев модуль включен в сборку.

#### Важно

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 <i>размер</i> ;
По умолчанию	ssl_buffer_size 16k;
Контекст	http, server

Задаёт размер буфера, используемого при отправке данных.

По умолчанию размер буфера равен 16к, 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.

**Важно**

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*.

**Важно**

Нужно иметь в виду, что из-за ограничения протокола HTTPS для максимальной совместимости виртуальные серверы должны слушать на разных IP-адресах.

Добавлено в версии 1.2.0: 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\_key

Синтаксис	<code>ssl_certificate_key <i>файл</i>;</code>
По умолчанию	—
Контекст	<code>http, server</code>

Указывает файл с секретным ключом в формате PEM для данного виртуального сервера.

### Важно

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 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*.

Добавлено в версии 1.2.0: 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\_ciphers

Синтаксис	<code>ssl_ciphers <i>шифры</i>;</code>
По умолчанию	<code>ssl_ciphers HIGH:!aNULL:!MD5;</code>
Контекст	<code>http, server</code>

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`.

## ssl\_client\_certificate

Синтаксис	<code>ssl_client_certificate файл;</code>
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 имя значение;</code>
По умолчанию	—
Контекст	http, server

Задаёт произвольные конфигурационные команды OpenSSL.

### Важно

Директива поддерживается при использовании OpenSSL 1.0.2 и выше.

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`.

### Осторожно

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 <i>файл</i>;</code>
По умолчанию	—
Контекст	<code>http, server</code>

Указывает файл с параметрами для DHE-шифров.

#### **Осторожно**

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	<code>http, server</code>

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;
```

#### **Важно**

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.

#### **Важно**

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.

#### Важно

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\_ntls

Добавлено в версии 1.2.0.

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

Включает серверную поддержку NTLS при использовании TLS библиотеки [TongSuo](#)

```
listen ... ssl;
ssl_ntls on;
```

#### Важно

Angie необходимо собрать с использованием параметра конфигурации `--with-ntls`, с соответствующей SSL библиотекой с поддержкой NTLS

```
./configure --with-openssl=../Tongsuo-8.3.0 \
            --with-openssl-opt=enable-ntls \
            --with-ntls
```

## ssl\_ocsp

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

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	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	ssl_prefer_server_ciphers on   off;
По умолчанию	ssl_prefer_server_ciphers off;
Контекст	http, 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	http, server

Изменено в версии 1.2.0: Параметр TLSv1.3 добавлен к используемым по умолчанию.

Разрешает указанные протоколы.

**Важно**

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	ssl_reject_handshake on   off;
По умолчанию	ssl_reject_handshake off;
Контекст	http, server

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:

<b>off</b>	жёсткое запрещение использования кэша сессий: Angie явно сообщает клиенту, что сессии не могут использоваться повторно.
<b>none</b>	мягкое запрещение использования кэша сессий: Angie сообщает клиенту, что сессии могут использоваться повторно, но на самом деле не хранит параметры сессии в кэше.
<b>builtin</b>	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.
<b>shared</b>	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 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 файл;</code>
По умолчанию	—
Контекст	http, 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	http, 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	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	<code>http, server</code>

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`.

### Внимание

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	<code>http, server</code>

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	<code>http, server</code>

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>http, server</code>

Задаёт файл с доверенными сертификатами CA 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	<code>http, server</code>

Разрешает проверку клиентских сертификатов. Результат проверки доступен через переменную `$ssl_client_verify`.

<code>optional</code>	запрашивает клиентский сертификат, и если сертификат был предоставлен, проверяет его
<code>optional_no_ca</code>	запрашивает сертификат клиента, но не требует, чтобы он был подписан доверенным сертификатом CA. Это предназначено для случаев, когда фактическая проверка сертификата осуществляется внешним по отношению к Angie сервисом.

## ssl\_verify\_depth

Синтаксис	<code>ssl_verify_depth</code> <i>число</i> ;
По умолчанию	<code>ssl_verify_depth 1</code> ;
Контекст	<code>http, server</code>

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
```

#### **Важно**

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_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`

#### ❗ Важно

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`

#### ❗ Важно

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_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 `".`.

### Stub Status

Предоставляет доступ к базовой информации о состоянии сервера.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_stub_status_module`. В пакетах и образах из наших репозиториях модуль включен в сборку.

### 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	<code>server, location</code>

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_writing`

то же, что и значение *Writing*;

`$connections_waiting`

то же, что и значение *Waiting*.

### Sub

Фильтр, изменяющий в ответе одну заданную строку на другую.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_sub_module`. В пакетах и образах из наших репозиториев модуль включен в сборку.

### Пример конфигурации

```
location / {
    sub_filter '<a href="http://127.0.0.1:8080/' '<a href="https://$host/';
    sub_filter 'sub_filter</code> <i>строка замена</i> ; |
| 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     | <code>http, server, location</code>             |

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     | <code>http, server, location</code>    |

Определяет, сколько раз нужно искать каждую из заменяемых строк: один раз или многократно.

### sub\_filter\_types

|              |                                             |
|--------------|---------------------------------------------|
| Синтаксис    | <code>sub_filter_types mime-min ...;</code> |
| По умолчанию | <code>sub_filter_types text/html;</code>    |
| Контекст     | <code>http, server, location</code>         |

Разрешает замену строк в ответах с указанными MIME-типами в дополнение к *text/html*. Специальное значение "\*" соответствует любому MIME-типу.

## 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

### bind\_conn (PRO)

|              |                                  |
|--------------|----------------------------------|
| Синтаксис    | <code>bind_conn значение;</code> |
| По умолчанию | —                                |
| Контекст     | upstream                         |

Позволяет привязать серверное соединение к клиентскому в момент, когда *значение*, заданное строкой из переменных, становится отличным от "" и "0".

#### Внимание

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`.

#### Внимание

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 {
    ~*~N(?:TLM|egotiate) 1;
```

```

}

upstream ntlm_backend {
    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 (PRO)

Добавлено в версии 1.6.0: PRO

|              |                                                                                                                           |
|--------------|---------------------------------------------------------------------------------------------------------------------------|
| Синтаксис    | <code>feedback <i>переменная</i> [inverse] [factor=<i>число</i>] [account=<i>условная_переменная</i>] [last_byte];</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 берется значение обратной связи. Она должна представлять собой метрику производительности или состояния; предполагается, что сервер передает ее в заголовках или иным образом.<br>Значение оценивается при каждом ответе от сервера и учитывается в скользящем среднем согласно настройкам <code>inverse</code> и <code>factor</code> .                  |
| <code>inverse</code> | Если параметр задан, значение обратной связи интерпретируется наоборот: более низкие значения указывают на лучшую производительность.                                                                                                                                                                                                                                           |
| <code>factor</code>  | Коэффициент, по которому значение обратной связи учитывается при расчете среднего. Допустимы целые числа от 0 до 99. По умолчанию — 90.<br>Среднее рассчитывается по формуле <b>экспоненциального сглаживания</b> .<br>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. |
| <code>account</code> | 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.

|                        |                                                                                                               |
|------------------------|---------------------------------------------------------------------------------------------------------------|
| <code>last_byte</code> | 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 может использоваться текст, переменные и их комбинации. 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`.

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` равным 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 {
    ip_hash;

    server backend1.example.com;
    server backend2.example.com;
    server backend3.example.com down;
    server backend4.example.com;
}
```

## keepalive

|              |                                    |
|--------------|------------------------------------|
| Синтаксис    | <code>keepalive соединения;</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.

### Внимание

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 {
    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 {
    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 {
    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</code> <i>число</i> ; |
| По умолчанию | <code>keepalive_requests</code> 1000;          |
| 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 время;</code> |
| По умолчанию | <code>keepalive_time 1h;</code>    |
| Контекст     | <code>upstream</code>              |

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 время;</code> |
| По умолчанию | <code>keepalive_timeout 60s;</code>   |
| Контекст     | <code>upstream</code>                 |

Задаёт таймаут, в течение которого неактивное постоянное соединение с сервером группы не будет закрыто.

### least\_conn

|              |                          |
|--------------|--------------------------|
| Синтаксис    | <code>least_conn;</code> |
| По умолчанию | —                        |
| Контекст     | <code>upstream</code>    |

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 (PRO)

|              |                                                                                          |
|--------------|------------------------------------------------------------------------------------------|
| Синтаксис    | <code>least_time header   last_byte [factor=число] [account=условная_переменная];</code> |
| По умолчанию | —                                                                                        |
| Контекст     | <code>upstream</code>                                                                    |

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.

|                  |                                                                |
|------------------|----------------------------------------------------------------|
| <b>header</b>    | Директива учитывает среднее время получения заголовков ответа. |
| <b>last_byte</b> | Директива использует среднее время получения полного ответа.   |

Добавлено в версии 1.7.0: PRO

|                |                                                                                                                                                                                      |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>factor</b>  | Выполняет ту же функцию, что и <i>response_time_factor (PRO)</i> , и переопределяет его, если параметр задан.                                                                        |
| <b>account</b> | 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 ответы в ходе *активных проверок* 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 анстрима* в API.

### queue (PRO)

Добавлено в версии 1.4.0: PRO

|              |                                                         |
|--------------|---------------------------------------------------------|
| Синтаксис    | <code>queue <i>число</i> [timeout=<i>время</i>];</code> |
| По умолчанию | —                                                       |
| Контекст     | 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).

**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.

**Внимание**

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 (PRO)

|              |                                          |
|--------------|------------------------------------------|
| Синтаксис    | <code>response_time_factor число;</code> |
| По умолчанию | <code>response_time_factor 90;</code>    |
| Контекст     | upstream                                 |

Задаёт для метода балансировки нагрузки *least\_time (PRO)* 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 подсчете 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>    | задает 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 <i>зоне</i> разделяемой памяти, то ограничение работает отдельно для каждого рабочего процесса. |

#### Примечание

При включенных *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` — задает 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 признается неработающим с точки зрения `upstream_probe (PRO)`; клиентские запросы не будут направляться к нему, пока проверки не признают его работающим.

#### 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.

|                          |                            |
|--------------------------|----------------------------|
| <code>max_fails=1</code> | число попыток по умолчанию |
| <code>max_fails=0</code> | 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.



|               |                                                                                                                                                                                                             |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>backup</b> | помечает сервер как запасной. На него будут передаваться запросы в случае, если не работают основные серверы.                                                                                               |
| <b>down</b>   | помечает сервер как постоянно недоступный.                                                                                                                                                                  |
| <b>drain</b>  | помечает сервер как разгружаемый ( <i>draining</i> ); это значит, что он получает только запросы сессий, привязанных ранее через <i>sticky</i> . В остальном поведение такое же, как в режиме <i>down</i> . |

### Осторожно

Параметр `backup` нельзя использовать совместно с методами балансировки нагрузки *hash*, *ip\_hash* и *random*.

Параметры `down` и `drain` взаимно исключающие.

Добавлено в версии 1.1.0.

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>resolve</b>     | позволяет отслеживать изменения списка IP-адресов, соответствующего доменному имени, и обновлять его без перезагрузки конфигурации. При этом группа должна находиться в <i>зоне разделяемой памяти</i> ; также должен быть определен <i>преобразователь имен в адреса</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>service=имя</b> | включает преобразование SRV-записей DNS и задает имя сервиса. Для работы параметра необходимо задать параметр <i>resolve</i> у сервера, не указывая порт сервера при имени хоста.<br>Если в имени службы нет точек, формируется имя по стандарту RFC: к имени службы добавляется префикс <code>_</code> , затем через точку добавляется <code>_tcp</code> . Так, имя службы <code>http</code> даст в результате <code>_http._tcp</code> .<br>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. <ul style="list-style-type: none"> <li>• SRV-записи с наивысшим приоритетом (те, которые имеют минимальное значение приоритета) разрешаются как основные серверы, а прочие записи становятся запасными серверами. Если <code>backup</code> установлено с <code>server</code>, SRV-записи с наивысшим приоритетом разрешаются как запасные серверы, а прочие записи игнорируются.</li> <li>• Вес аналогичен параметру <code>weight</code> директивы <code>server</code>. Если вес задан как в самой директиве, так и в SRV-записи, используется вес, установленный в директиве.</li> </ul> |

В этом примере выполняется поиск записи `_http._tcp.backend.example.com`:

```
server backend.example.com service=http resolve;
```

Добавлено в версии 1.2.0: Angie

Добавлено в версии 1.1.0-P1: Angie PRO

|               |                                                                                                                                          |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------|
| <b>sid=id</b> | задает ID сервера в группе. Если параметр не задан, то ID задается как шестнадцатеричный MD5-хэш IP-адреса и порта или пути UNIX-сокета. |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------|

Добавлено в версии 1.4.0.

`slow_start=время` задает время восстановления веса сервера, возвращающегося к работе при балансировке нагрузки методом *round-robin* или *least\_conn*.  
Если параметр задан и сервер после сбоя снова считается работающим с точки зрения *max\_fails* и *upstream\_probe (PRO)*, 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 (PRO)

Добавлено в версии 1.2.0: PRO

|              |                          |
|--------------|--------------------------|
| Синтаксис    | <code>state файл;</code> |
| По умолчанию | —                        |
| Контекст     | <code>upstream</code>    |

Указывает *файл*, где постоянно хранится список серверов апстрима. При установке из наших пакетов для хранения таких файлов специально создается каталог `/var/lib/angie/state/` (`/var/db/angie/state/` во FreeBSD) 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. Файл считывается при запуске Angie или перезагрузке конфигурации.

#### Осторожно

Чтобы использовать директиву `state` в блоке `upstream`, в нем не должно быть директив `server`, но нужна зона разделяемой памяти (*zone*).

## sticky

Добавлено в версии 1.2.0: Angie

Добавлено в версии 1.1.0-P1: Angie PRO

|              |                                                                                                                                                                                                                                                                                                                                                                               |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Синтаксис    | <code>sticky cookie name [attr=значение]...;</code><br><code>sticky route \$variable...;</code><br><code>sticky learn zone=зона create=\$create_var1... lookup=\$lookup_var1... [header]</code><br><code>[timeout=время];</code><br><code>sticky learn zone=зона lookup=\$lookup_var1... remote_action=uri</code><br><code>remote_result=\$remote_var [timeout=время];</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`.

### Внимание

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` (*PRO*), 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 {
    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 predefined идентификаторы маршрутов, которые могут быть встроены в URL, cookie или другие свойства запроса. Он менее гибок, так как зависит от predefined 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 и клиенту, и серверу. 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 переменные для маршрутизации. Чтобы выбрать сервер, куда передается поступивший запрос, используется первая непустая переменная; 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 {
    server backend1.example.com:8080 "sid=server 1";
    server backend2.example.com:8080 "sid=server 2";

    sticky route $cookie_route $arg_route;
}
```

### Режим learn (PRO 1.4.0+)

В этом режиме для привязки клиента к конкретному проксируемому серверу используется динамически генерируемый ключ; он более гибок, так как назначает серверы на ходу, хранит сеансы в зоне общей 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 общей памяти; ее имя и размер задаются параметром *zone*. Если к сессии не было обращений в течение времени *timeout*, она удаляется. Значение по умолчанию — 10 минут.

Последующие запросы от клиентов, желающих использовать сессию, должны содержать ее идентификатор, причем так, чтобы он попал в непустую переменную, указанную с *lookup*; тогда его значение будет сопоставлено с сессиями в общей 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 {
    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` (PRO 1.8.0+)

Параметры `remote_action` и `remote_result` позволяют динамически назначать идентификаторы сессий и управлять ими с использованием удаленного хранилища сессий. При этом зона общей 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. Настройка `remote_action` не влияет на время ожидания. Значение по умолчанию — 10 минут.

Начальный идентификатор сессии всегда поступает из `lookup`; если он найден в локальной общей памяти, Angie переходит к выбору соответствующего сервера.

Если идентификатор сессии не найден локально, Angie отправляет синхронный подзапрос в удаленное хранилище. Параметр `remote_action` задает URI удаленного хранилища, которое должно вести поиск и создание сессий следующим образом:

- Хранилище принимает идентификатор сессии из `lookup` и локально предложенный идентификатор сервера, связанный с данной сессией, через пользовательские заголовки или другим способом. На стороне Angie для этого предоставляются две специальные переменные: `$sticky_sessid` и `$sticky_sid`, соответственно. В `sticky_sid` содержится значение параметра `sid=` из блока `upstream` директивы `server`, если оно задано, либо MD5-хэш имени сервера.
- Ответ с кодом 200 от удаленного хранилища указывает на то, что оно приняло сессию и сохранило ее с предложенными значениями для дальнейшего использования.
- Ответ с кодом 409 от удаленного хранилища указывает на то, что данный идентификатор сессии уже существует. В этом случае ответ должен содержать альтернативный идентификатор сессии в заголовке `X-Sticky-Sid`. Angie сохраняет этот идентификатор в переменной, заданной параметром `remote_result`.

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;
    }
}

```

### sticky\_secret

Добавлено в версии 1.2.0: Angie

Добавлено в версии 1.1.0-P1: Angie PRO

|              |                                    |
|--------------|------------------------------------|
| Синтаксис    | <code>sticky_secret строка;</code> |
| По умолчанию | —                                  |
| Контекст     | upstream                           |

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 {
    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

Добавлено в версии 1.2.0: Angie

Добавлено в версии 1.1.0-P1: Angie PRO

|              |                                      |
|--------------|--------------------------------------|
| Синтаксис    | <code>sticky_strict on   off;</code> |
| По умолчанию | <code>sticky_strict off;</code>      |
| Контекст     | upstream                             |

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 имя { ... }</code> |
| По умолчанию | —                                 |
| Контекст     | http                              |

Описывает группу серверов. Серверы могут слушать на разных портах. Кроме того, можно одновременно использовать серверы, слушающие на TCP- и UNIX-сокетах.

Пример:

```
upstream backend {
    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.

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.

## zone

|              |                                 |
|--------------|---------------------------------|
| Синтаксис    | <code>zone имя [размер];</code> |
| По умолчанию | —                               |
| Контекст     | upstream                        |

Задаёт имя и размер зоны разделяемой памяти, в которой хранятся конфигурация группы и ее 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.

## 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_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_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_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 ответа, полученного от сервера группы. Статусы нескольких ответов разделяются запятыми и двоеточиями подобно адресам в переменной `$upstream_addr`. Если сервер не может быть выбран, то переменная хранит статус 502 (Bad Gateway).

### `$upstream_sticky_status`

Статус привязанных запросов.

|                   |                                                                              |
|-------------------|------------------------------------------------------------------------------|
| <code>""</code>   | Запрос отправлен в группу серверов, где привязка не включена.                |
| <code>NEW</code>  | Запрос не содержит информации о привязке к серверу.                          |
| <code>HIT</code>  | Запрос с привязкой отправлен на желаемый сервер.                             |
| <code>MISS</code> | 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 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 (PRO)

Добавлено в версии 1.2.0: PRO

|              |                                                                                                                                                                                                                                                                                                                                                                                            |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Синтаксис    | <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>essential</code> ] [ <code>persistent</code> ] [ <code>fails=число</code> ] [ <code>passes=число</code> ] [ <code>max_body=размер</code> ] [ <code>mode=always   idle   onfail</code> ]; |
| По умолчанию | —                                                                                                                                                                                                                                                                                                                                                                                          |
| Контекст     | location                                                                                                                                                                                                                                                                                                                                                                                   |

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 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:

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>имя</b>        | Обязательное имя проверки.                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <b>uri</b>        | URI запроса, который добавляется к аргументу <i>proxy_pass</i> , <i>uwsgi_pass</i> и т. д. По умолчанию — /.                                                                                                                                                                                                                                                                                                                               |
| <b>port</b>       | Альтернативный порт для запроса.                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>interval</b>   | Интервал между проверками. По умолчанию — 5s.                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>method</b>     | HTTP-метод запроса проверки. По умолчанию — GET.                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>test</b>       | Проверяемое при запросе условие; задается строкой с переменными. Если результат подстановки переменных — "" или "0", проверка не пройдена.                                                                                                                                                                                                                                                                                                 |
| <b>essential</b>  | Если параметр задан, то изначально состояние сервера подлежит уточнению и клиентские запросы не передаются ему, пока проверка не будет пройдена.                                                                                                                                                                                                                                                                                           |
| <b>persistent</b> | Установка этого параметра требует сначала включить <b>essential</b> ; серверы с <b>persistent</b> , работавшие до перезагрузки конфигурации, начинают получать запросы без необходимости сначала пройти эту проверку.                                                                                                                                                                                                                      |
| <b>fails</b>      | Число последовательных неуспешных запросов, при котором проверка считает сервер неработающим. По умолчанию — 1.                                                                                                                                                                                                                                                                                                                            |
| <b>passes</b>     | Число последовательных успешных запросов, при котором проверка считает сервер работающим. По умолчанию — 1.                                                                                                                                                                                                                                                                                                                                |
| <b>max_body</b>   | Максимальный объем памяти для тела ответа. По умолчанию — 256k.                                                                                                                                                                                                                                                                                                                                                                            |
| <b>mode</b>       | Режим проверки в зависимости от работоспособности серверов: <ul style="list-style-type: none"> <li>• <b>always</b> — серверы проверяются независимо от состояния;</li> <li>• <b>idle</b> — проверяются неработающие серверы, а также серверы, где с последнего клиентского запроса прошло время <b>interval</b>.</li> <li>• <b>onfail</b> — проверяются серверы только в неработающем состоянии.</li> </ul> По умолчанию — <b>always</b> . |

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 (PRO)

Имя активной сейчас проверки *upstream\_probe*.

### \$upstream\_probe\_body (PRO)

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:

|            |                                                                       |
|------------|-----------------------------------------------------------------------|
| <b>on</b>  | разрешает выдачу cookie версии 2 и запись приходящих cookie в лог;    |
| <b>v1</b>  | разрешает выдачу cookie версии 1 и запись приходящих cookie в лог;    |
| <b>log</b> | запрещает выдачу cookie, но разрешает запись приходящих cookie в лог; |
| <b>off</b> | запрещает выдачу cookie и запись приходящих cookie в лог.             |

### userid\_domain

|              |                                        |
|--------------|----------------------------------------|
| Синтаксис    | <code>userid_domain имя   none;</code> |
| По умолчанию | <code>userid_domain none;</code>       |
| Контекст     | <code>http, server, location</code>    |

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     | <code>http, server, location</code>            |

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   флаг ...;</code> |
| По умолчанию | <code>userid_flags off;</code>            |
| Контекст     | <code>http, server, location</code>       |

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     | <code>http, server, location</code>               |

Если параметр не `off`, включает механизм маркировки cookie и задаёт 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 "P3P".

### userid\_name

|              |                          |
|--------------|--------------------------|
| Синтаксис    | userid_name <i>имя</i> ; |
| По умолчанию | userid_name uid;         |
| Контекст     | http, server, location   |

Задает имя cookie.

### userid\_p3p

|              |                                  |
|--------------|----------------------------------|
| Синтаксис    | userid_p3p <i>строка</i>   none; |
| По умолчанию | userid_p3p none;                 |
| Контекст     | http, server, location           |

Задает значение для поля заголовка "P3P", которое будет выдаваться вместе с cookie. Если задано специальное значение none, то в ответе не будет заголовка "P3P".

### userid\_path

|              |                           |
|--------------|---------------------------|
| Синтаксис    | userid_path <i>путь</i> ; |
| По умолчанию | userid_path /;            |
| Контекст     | http, server, location    |

Задает путь, для которого устанавливается cookie.

### userid\_service

|              |                                    |
|--------------|------------------------------------|
| Синтаксис    | userid_service <i>номер</i> ;      |
| По умолчанию | userid_service <IP-адрес сервера>; |
| 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 допустимо использование переменных. Специальное значение `off` отменяет действие унаследованной с предыдущего уровня конфигурации директивы `uwsgi_bind`, позволяя системе самостоятельно выбирать локальный IP-адрес и порт.

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 `superпользователя`. В Linux это не требуется, так как если указан параметр `transparent`, то рабочие процессы наследуют `capability CAP_NET_RAW` из главного процесса.



### Важно

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 размер;</code> |
| По умолчанию | <code>uwsgi_buffering on;</code>     |
| Контекст     | <code>http, server, location</code>  |

|            |                                                                                                                                                                                                                                                                                                                                                                                                                      |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>on</b>  | Angie принимает ответ uwsgi-сервера как можно быстрее, сохраняя его в буферах, заданные директивами <code>uwsgi_buffer_size</code> и <code>uwsgi_buffers</code> . Если ответ не влезает целиком в память, то его часть может быть записана на диск во <i>временный файл</i> . Запись во временные файлы контролируется директивами <code>uwsgi_max_temp_file_size</code> и <code>uwsgi_temp_file_write_size</code> . |
| <b>off</b> | Ответ синхронно передается клиенту сразу же по мере его поступления. Angie не пытается считать весь ответ uwsgi-сервера. Максимальный размер данных, который Angie может принять от сервера за один раз, задается директивой <code>uwsgi_buffer_size</code> .                                                                                                                                                        |

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 число размер;</code> |
| По умолчанию | <code>uwsgi_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 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 <i>размер</i>;</code> |
| По умолчанию | <code>uwsgi_busy_buffers_size 8k   16k;</code>      |
| Контекст     | <code>http, server, location</code>                 |

При включенной *буферизации* ответов uwsgi-сервера директива 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 <i>зона</i>   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. |
|------------------|--------------------------------------------------------------------------|

### 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.

#### **Внимание**

Использование устаревшего кэшированного ответа в момент его обновления должно быть *разрешено*.

### uwsgi\_cache\_bypass

|              |                                      |
|--------------|--------------------------------------|
| Синтаксис    | <code>uwsgi_cache_bypass ...;</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 берётся из кэша:

```
uwsgi_cache_bypass $cookie_nocache $arg_nocache$arg_comment;
uwsgi_cache_bypass $http_pragma $http_authorization;
```

Можно использовать совместно с директивой `uwsgi_no_cache`.

### uwsgi\_cache\_key

|              |                                      |
|--------------|--------------------------------------|
| Синтаксис    | <code>uwsgi_cache_key строка;</code> |
| По умолчанию | —                                    |
| Контекст     | <code>http, server, location</code>  |

Задаёт ключ для кэширования, 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     | <code>http, server, location</code>     |

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     | <code>http, server, location</code>      |

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    | <code>uwsgi_cache_lock_timeout</code> <i>время</i> ; |
| По умолчанию | <code>uwsgi_cache_lock_timeout 5s</code> ;           |
| Контекст     | <code>http, server, location</code>                  |

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    | <code>uwsgi_cache_max_range_offset</code> <i>число</i> ; |
| По умолчанию | —                                                        |
| Контекст     | <code>http, server, location</code>                      |

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    | <code>uwsgi_cache_methods</code> GET   HEAD   POST ...; |
| По умолчанию | <code>uwsgi_cache_methods</code> GET HEAD;              |
| Контекст     | <code>http, server, location</code>                     |

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    | <code>uwsgi_cache_min_uses</code> <i>число</i> ; |
| По умолчанию | <code>uwsgi_cache_min_uses 1</code> ;            |
| Контекст     | <code>http, server, location</code>              |

Задаёт число запросов, после которого ответ будет кэширован.

## uwsgi\_cache\_path

|              |                                                                                                                                                                                                                                                                                                                                                                                                     |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Синтаксис    | <code>uwsgi_cache_path <i>путь</i> [levels=<i>уровни</i>] [use_temp_path=<i>on</i>   <i>off</i>] [keys_zone=<i>имя:размер</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*.

Параметр `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.

Какой из каталогов будет использоваться для временных файлов, определяется параметром `use_temp_path`.

|                  |                                                                                                                                                                           |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>on</code>  | Если параметр не задан или установлен в значение "on", 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                                                                                                       |

Кроме того, все активные ключи и информация о данных хранятся в зоне разделяемой памяти, имя и размер которой задаются параметром `keys_zone`. Зоны размером в 1 мегабайт достаточно для хранения около 8 тысяч ключей.

Если к данным кэша не обращаются в течение времени, заданного параметром `inactive`, то данные удаляются, независимо от их свежести.

По умолчанию `inactive` 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.

|                                |                                                                                                |
|--------------------------------|------------------------------------------------------------------------------------------------|
| <code>max_size</code>          | максимальное пороговое значение размера кэша                                                   |
| <code>min_free</code>          | минимальное пороговое значение объема свободного места на файловой системе с кэшем             |
| <code>manager_files</code>     | максимальное количество удаляемых элементов кэша за одну итерацию<br>По умолчанию: 100         |
| <code>manager_threshold</code> | ограничивает время работы одной итерации<br>По умолчанию: 200 миллисекунд                      |
| <code>manager_sleep</code>     | время, в течение которого выдерживается пауза между итерациями<br>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.

|                               |                                                                                                |
|-------------------------------|------------------------------------------------------------------------------------------------|
| <code>loader_files</code>     | максимальное количество элементов кэша к загрузке в одну итерацию<br>По умолчанию: 100         |
| <code>loader_threshold</code> | ограничивает время работы одной итерации<br>По умолчанию: 200 миллисекунд                      |
| <code>loader_sleep</code>     | время, в течение которого выдерживается пауза между итерациями<br>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   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`.

|                       |                                                                                                                                                                                                                            |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>error</code>    | Позволяет использовать устаревший кэшированный ответ при невозможности выбора uwsgi-сервера для обработки запроса.                                                                                                         |
| <code>updating</code> | 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. |

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.

#### **i** 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     | <code>http, server, location</code>             |

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;
```

#### **i** 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" задаёт время кэширования ответа в секундах. Значение 0 запрещает кэшировать ответ. Если значение начинается с префикса @, оно задаёт 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 соединений с проксируемым сервером, если он был удален из группы или помечен как постоянно недоступный в результате процесса `resolve` или команды `API DELETE`.

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 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 некэшированных 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     | <code>http, server, location</code>  |

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     | <code>http, server, location</code>              |

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     | <code>http, server, location</code>         |

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     | <code>http, server, location</code>           |

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\_limit\_rate

|              |                                         |
|--------------|-----------------------------------------|
| Синтаксис    | <code>uwsgi_limit_rate скорость;</code> |
| По умолчанию | <code>uwsgi_limit_rate 0;</code>        |
| Контекст     | <code>http, server, location</code>     |

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     | <code>http, server, location</code>           |

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:

|                             |                                                                                                                                                                                                                                           |
|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <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_503</code>       | сервер вернул ответ с кодом 503;                                                                                                                                                                                                          |
| <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 <b>неидемпотентным</b> 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; |
| <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.

|                |                                                                             |
|----------------|-----------------------------------------------------------------------------|
| 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*.

### uwsgi\_next\_upstream\_timeout

|              |                                                        |
|--------------|--------------------------------------------------------|
| Синтаксис    | <code>uwsgi_next_upstream_timeout <i>время</i>;</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 <i>число</i>;</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 <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:

```
uwsgi_no_cache $cookie_nocache $arg_nocache$arg_comment;
uwsgi_no_cache $http_pragma $http_authorization;
```

Можно использовать совместно с директивой `uwsgi_cache_bypass`.

### 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.

### uwsgi\_pass\_header

|              |                                          |
|--------------|------------------------------------------|
| Синтаксис    | <code>uwsgi_pass_header поле ...;</code> |
| По умолчанию | —                                        |
| Контекст     | http, server, location                   |

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     | http, server, location                         |

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 <i>время</i>;</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.

|                  |                                                                                                                                                                                                |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>on</code>  | тело запроса полностью <i>читается</i> от клиента перед отправкой запроса на uwsgi-сервер.                                                                                                     |
| <code>off</code> | тело запроса отправляется на uwsgi-сервер сразу же по мере его поступления. В этом случае запрос не может быть передан <i>следующему серверу</i> , 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 <i>время</i>;</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 suwsgi-сервере. В имени файла можно использовать переменные.

### uwsgi\_ssl\_certificate\_key

|              |                                                     |
|--------------|-----------------------------------------------------|
| Синтаксис    | <code>uwsgi_ssl_certificate_key <i>файл</i>;</code> |
| По умолчанию | —                                                   |
| Контекст     | <code>http, server, location</code>                 |

Задаёт файл с секретным ключом в формате PEM для аутентификации на suwsgi-сервере.

Вместо файла можно указать значение "engine:*имя*:id", которое загружает ключ с указанным *id* из OpenSSL engine с заданным именем.

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     | <code>http, server, location</code>          |

Описывает разрешенные шифры для запросов к suwsgi-серверу. Шифры задаются в формате, поддерживаемом библиотекой OpenSSL.

Список шифров зависит от установленной версии OpenSSL. Полный список можно посмотреть с помощью команды `openssl ciphers`.

## uwsgi\_ssl\_conf\_command

|              |                                                          |
|--------------|----------------------------------------------------------|
| Синтаксис    | <code>uwsgi_ssl_conf_command <i>имя значение</i>;</code> |
| По умолчанию | —                                                        |
| Контекст     | <code>http, server, location</code>                      |

Задаёт произвольные конфигурационные команды OpenSSL при установлении соединения с uwsgi HTTPS-сервером.

### Важно

Директива поддерживается при использовании OpenSSL 1.0.2 и выше.

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`.

### Осторожно

Следует учитывать, что изменение настроек OpenSSL напрямую может привести к неожиданному поведению.

## uwsgi\_ssl\_crl

|              |                                         |
|--------------|-----------------------------------------|
| Синтаксис    | <code>uwsgi_ssl_crl <i>файл</i>;</code> |
| По умолчанию | —                                       |
| Контекст     | <code>http, server, location</code>     |

Указывает файл с отозванными сертификатами (CRL) в формате PEM, используемыми при *проверке* сертификата suwsgi-сервера.

## uwsgi\_ssl\_name

|              |                                                        |
|--------------|--------------------------------------------------------|
| Синтаксис    | <code>uwsgi_ssl_name <i>имя</i>;</code>                |
| По умолчанию | <code>uwsgi_ssl_name `имя хоста из uwsgi_pass`;</code> |
| Контекст     | <code>http, server, location</code>                    |

Позволяет переопределить имя сервера, используемое при *проверке* сертификата uwsgi HTTPS-сервера, а также для *передачи его через SNI* при установлении соединения с suwsgi-сервером.

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     | <code>http, server, location</code>               |

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     | <code>http, server, location</code>                                                     |

Изменено в версии 1.2.0: Параметр `TLSv1.3` добавлен к используемым по умолчанию.

Разрешает указанные протоколы для запросов к suwsgi-серверу.

### uwsgi\_ssl\_server\_name

|              |                                              |
|--------------|----------------------------------------------|
| Синтаксис    | <code>uwsgi_ssl_server_name on   off;</code> |
| По умолчанию | <code>uwsgi_ssl_server_name off;</code>      |
| Контекст     | <code>http, server, location</code>          |

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     | <code>http, server, location</code>            |

Определяет, использовать ли повторно SSL-сессии при работе с suwsgi-сервером. Если в логах появляются ошибки `"SSL3_GET_FINISHED:digest check failed"`, то можно попробовать выключить повторное использование сессий.

### uwsgi\_ssl\_trusted\_certificate

|              |                                             |
|--------------|---------------------------------------------|
| Синтаксис    | uwsgi_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 uwsgi HTTPS-сервера.

### uwsgi\_ssl\_verify

|              |                            |
|--------------|----------------------------|
| Синтаксис    | uwsgi_ssl_verify on   off; |
| По умолчанию | uwsgi_ssl_verify off;      |
| Контекст     | http, server, location     |

Разрешает или запрещает проверку сертификата suwsgi-сервера.

### uwsgi\_ssl\_verify\_depth

|              |                                       |
|--------------|---------------------------------------|
| Синтаксис    | uwsgi_ssl_verify_depth <i>число</i> ; |
| По умолчанию | uwsgi_ssl_verify_depth 1;             |
| Контекст     | http, server, location                |

Устанавливает глубину проверки в цепочке сертификатов suwsgi-сервера.

### uwsgi\_store

|              |                                        |
|--------------|----------------------------------------|
| Синтаксис    | uwsgi_store on   off   <i>строка</i> ; |
| По умолчанию | uwsgi_store off;                       |
| 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                                                                   |

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 <i>пользователи:права</i> ...;</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\_temp\_file\_write\_size

|              |                                                        |
|--------------|--------------------------------------------------------|
| Синтаксис    | <code>uwsgi_temp_file_write_size <i>размер</i>;</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.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_v2_module`. В пакетах и образах из наших репозиторий модуль включен в сборку.

### Пример конфигурации

```
server {
    listen 443 ssl;

    http2 on;

    ssl_certificate server.crt;
    ssl_certificate_key server.key;
}
```

#### Важно

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 черному списку RFC 9113, Appendix A а также поддерживаться клиентами.

## Директивы

### http2

Добавлено в версии 1.2.0.

|              |                              |
|--------------|------------------------------|
| Синтаксис    | <code>http2 on   off;</code> |
| По умолчанию | <code>http2 off;</code>      |
| Контекст     | <code>http, server</code>    |

Разрешает протокол HTTP/2.

### http2\_body\_preload\_size

|              |                                                     |
|--------------|-----------------------------------------------------|
| Синтаксис    | <code>http2_body_preload_size <i>размер</i>;</code> |
| По умолчанию | —                                                   |
| Контекст     | <code>http, server</code>                           |

Задаёт размер буфера для каждого запроса, в который может сохраняться тело запроса до того, как оно начнет обрабатываться.

### http2\_chunk\_size

|              |                                              |
|--------------|----------------------------------------------|
| Синтаксис    | <code>http2_chunk_size <i>размер</i>;</code> |
| По умолчанию | <code>http2_chunk_size 8k;</code>            |
| Контекст     | <code>http, server, location</code>          |

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, начиная с версии 1.2.0.

|              |                                                        |
|--------------|--------------------------------------------------------|
| Синтаксис    | <code>http2_max_concurrent_pushes <i>число</i>;</code> |
| По умолчанию | <code>http2_max_concurrent_pushes 10;</code>           |
| Контекст     | <code>http, server</code>                              |

Ограничивает максимальное число параллельных *push*-запросов в соединении.

### http2\_max\_concurrent\_streams

|              |                                                         |
|--------------|---------------------------------------------------------|
| Синтаксис    | <code>http2_max_concurrent_streams <i>число</i>;</code> |
| По умолчанию | <code>http2_max_concurrent_streams 128;</code>          |
| Контекст     | <code>http, server</code>                               |

Задаёт максимальное число параллельных HTTP/2-потоков в соединении.

### http2\_push

Устарело, начиная с версии 1.2.0.

|              |                                           |
|--------------|-------------------------------------------|
| Синтаксис    | <code>http2_push <i>uri</i>   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` допустимо использование переменных.

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, начиная с версии 1.2.0.

|              |                                           |
|--------------|-------------------------------------------|
| 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 <i>размер</i>;</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`

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_v3_module`. В пакетах и образах из наших репозиториев модуль включен в сборку.

## 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';
    }
}
}

```

**Важно**

Чтобы принимать HTTP/3-соединения по TLS, необходимо наличие поддержки протокола TLSv1.3, появившейся в OpenSSL версии 1.1.1.

## 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.

### 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</code> <i>число</i> ; |
| Значение по умолчанию | <code>http3_max_table_capacity</code> 4096;          |
| Контекст              | http, server                                         |

Определяет емкость динамической таблицы для серверных соединений.

#### 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> <i>размер</i> ; |
| По умолчанию | <code>http3_stream_buffer_size</code> 64k;            |
| Контекст     | http, server                                          |

Задаёт размер буфера, используемого для чтения и записи QUIC-потоков.

### quic\_active\_connection\_id\_limit

|              |                                                             |
|--------------|-------------------------------------------------------------|
| Синтаксис    | <code>quic_active_connection_id_limit</code> <i>число</i> ; |
| По умолчанию | <code>quic_active_connection_id_limit</code> 2;             |
| 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</code> on   off; |
| По умолчанию | <code>quic_bpf</code> off;      |
| Контекст     | main                            |

Разрешает маршрутизацию пакетов QUIC при помощи eBPF. Если маршрутизация включена, то обеспечивается поддержка миграции QUIC-соединений.

#### Важно

Директива поддерживается только на Linux 5.7+.

### quic\_gso

|              |                    |
|--------------|--------------------|
| Синтаксис    | quic_gso on   off; |
| По умолчанию | quic_gso off;      |
| Контекст     | http, server       |

Разрешает отправку оптимизированного пакетного режима при помощи segmentation offloading.

#### Важно

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:

|                 |                                     |
|-----------------|-------------------------------------|
| <code>h3</code> | для HTTP/3-соединений               |
| <code>hq</code> | для hq-соединений                   |
| <code>""</code> | пустая строка для остальных случаев |

`$quic_connection`

порядковый номер QUIC-соединения

## XSLT

Фильтр, преобразующий XML-ответ с помощью одного или нескольких XSLT-шаблонов.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-http_xslt_module`.

В наших репозиториях модуль собран динамически и доступен отдельным пакетом `angie-module-xslt` или `angie-pro-module-xslt`.

### Важно

Для этого модуля нужны библиотеки `libxml2` и `libxslt`.

## Пример конфигурации

```
location / {
    xml_entities    /site/dtd/entities.dtd;
    xslt_stylesheet /site/xslt/one.xslt param=value;
    xslt_stylesheet /site/xslt/two.xslt;
}
```

## Директивы

### `xml_entities`

|              |                                        |
|--------------|----------------------------------------|
| Синтаксис    | <code>xml_entities <i>путь</i>;</code> |
| По умолчанию | —                                      |
| Контекст     | <code>http, server, location</code>    |

Задаёт файл DTD, в котором описаны символьные сущности. Этот файл компилируется на стадии конфигурации. По техническим причинам модуль не имеет возможности использовать внешнее подмножество, заданное в обрабатываемом XML, поэтому оно игнорируется, а вместо него используется специально заданный файл. В этом файле не нужно описывать структуру XML, достаточно только объявления необходимых символьных сущностей, например:

```
<!ENTITY nbsp " ">
```

### xslt\_last\_modified

|              |                                           |
|--------------|-------------------------------------------|
| Синтаксис    | <code>xslt_last_modified on   off;</code> |
| По умолчанию | <code>xslt_last_modified off;</code>      |
| Контекст     | <code>http, server, location</code>       |

Позволяет сохранить поле заголовка "Last-Modified" исходного ответа во время XSLT-преобразований для лучшего кэширования ответов.

По умолчанию поле заголовка удаляется, так как содержимое ответа изменяется во время преобразования и может содержать динамически созданные элементы или части, которые изменились независимо от исходного ответа.

### xslt\_param

|              |                                            |
|--------------|--------------------------------------------|
| Синтаксис    | <code>xslt_param параметр значение;</code> |
| По умолчанию | —                                          |
| Контекст     | <code>http, server, location</code>        |

Задаёт параметры для XSLT-шаблонов. Значение рассматривается как выражение XPath. В значении можно использовать переменные. Если нужно передать в шаблон строковое значение, можно воспользоваться директивой *xslt\_string\_param*.

Директив *xslt\_param* может быть несколько. Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы *xslt\_param* и *xslt\_string\_param*.

### xslt\_string\_param

|              |                                                   |
|--------------|---------------------------------------------------|
| Синтаксис    | <code>xslt_string_param параметр значение;</code> |
| По умолчанию | —                                                 |
| Контекст     | <code>http, server, location</code>               |

Задаёт строковые параметры для XSLT-шаблонов. Выражения XPath в значении параметра не интерпретируются. В значении можно использовать переменные.

Директив *xslt\_string\_param* может быть несколько. Директивы наследуются с предыдущего уровня конфигурации при условии, что на данном уровне не описаны свои директивы *xslt\_param* и *xslt\_string\_param*.

## xslt\_stylesheet

|              |                                                              |
|--------------|--------------------------------------------------------------|
| Синтаксис    | <code>xslt_stylesheet шаблон [параметр=значение ...];</code> |
| По умолчанию | —                                                            |
| Контекст     | location                                                     |

Задаёт XSLT-шаблон и необязательные параметры для этого шаблона. Шаблон компилируется на стадии конфигурации.

Параметры можно задавать как по отдельности, так и группировать в одной строке, разделяя символом ":". Если же в самих параметрах встречается символ ":", то его нужно экранировать в виде "%3A". Кроме того, libxslt требует, чтобы параметры, содержащие не только алфавитно-цифровые символы, были заключены в одинарные или двойные кавычки, например:

```
param1='http%3A/www.example.com':param2=value2
```

В описании параметров можно использовать переменные, например, целая строка параметров может быть взята из одной переменной:

```
location / {
    xslt_stylesheet /site/xslt/one.xslt
        $arg_xslt_params
        param1='$value1':param2=value2
        param3=value3;
}
```

Можно указать несколько шаблонов — в этом случае они будут применяться последовательно в порядке их описания.

## xslt\_types

|              |                                       |
|--------------|---------------------------------------|
| Синтаксис    | <code>xslt_types mime-mun ...;</code> |
| По умолчанию | <code>xslt_types text/xml;</code>     |
| Контекст     | http, server, location                |

Разрешает преобразования в ответах с указанными MIME-типами в дополнение к `text/xml`. Специальное значение `*` соответствует любому MIME-типу. Если в результате преобразования выдается HTML-ответ, то его MIME-тип меняется на `text/html`.

Базовый HTTP-модуль реализует основную функциональность HTTP-сервера: это определение серверных блоков, настройка локаций для маршрутизации запросов, передача статических файлов и контроль доступа, настройка редиректов, 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    | <code>absolute_redirect on   off;</code> |
| По умолчанию | <code>absolute_redirect on;</code>       |
| Контекст     | <code>http, server, location</code>      |

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 [=null];</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;
```

По умолчанию поддержка многопоточности выключена, ее сборку следует разрешить с помощью конфигурационного параметра *--with-threads*. 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 путь;</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 выделения, а сам *alias* — ссылки на эти выделения, 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;
}
```

## auth\_delay

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

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     | http, server, location                           |

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\_body\_buffer\_size

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

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

Определяет, сохранять ли все тело запроса клиента в файл. Директиву можно использовать для отладки и при использовании переменной `$request_body_file` или метода `$r->request_body_file` модуля *Perl*.

|                    |                                                                              |
|--------------------|------------------------------------------------------------------------------|
| <code>on</code>    | временные файлы по окончании обработки запроса не удаляются                  |
| <code>clean</code> | разрешает удалять временные файлы, оставшиеся по окончании обработки запроса |

### client\_body\_in\_single\_buffer

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

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 <i>путь</i> [<i>уровень1</i> [<i>уровень2</i> [<i>уровень3</i>]]];</code>                              |
| По умолчанию | <code>client_body_temp_path client_body_temp;</code> (путь зависит от параметра сборки <code>--http-client-body-temp-path</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 <i>время</i>;</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 <i>размер</i>;</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</code> <i>время</i> ; |
| По умолчанию | <code>client_header_timeout</code> 60s;           |
| 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> <i>размер</i> ; |
| По умолчанию | <code>client_max_body_size</code> 1m;             |
| 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> <i>размер</i> ; |
| По умолчанию | <code>connection_pool_size</code> 256   512;      |
| 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> <i>mime-тип</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 размер   off;</code> |
| По умолчанию | <code>directio off;</code>          |
| 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><br><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:

|                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|---------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>off</code>          | Символические ссылки в пути допускаются и не проверяются. Это стандартное поведение.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <code>on</code>           | Если любой компонент пути является символической ссылкой, доступ к файлу запрещается.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <code>if_not_owner</code> | Доступ к файлу запрещается, если любой компонент пути является символической ссылкой, а ссылка и объект, на который она ссылается, имеют разных владельцев.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <code>from=часть</code>   | При проверке символических ссылок (параметры <code>on</code> и <code>if_not_owner</code> ) обычно проверяются все компоненты пути. Можно не проверять символические ссылки в начальной части пути, указав дополнительно параметр <code>from=часть</code> . 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.

## error\_page

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

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    | <code>etag on   off;</code>         |
| По умолчанию | <code>etag on;</code>               |
| Контекст     | <code>http, server, location</code> |

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 "If-Modified-Since" 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 клиенту будет возвращаться ошибка 404 (Not Found). 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*.

Пример:

```
error_page 404 /404.html;

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

### 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    | <code>keepalive_disable none   браузер ...;</code> |
| По умолчанию | <code>keepalive_disable msie6;</code>              |
| Контекст     | <code>http, server, location</code>                |

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

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

## keepalive\_requests

|              |                                        |
|--------------|----------------------------------------|
| Синтаксис    | <code>keepalive_requests число;</code> |
| По умолчанию | <code>keepalive_requests 1000;</code>  |
| Контекст     | <code>http, server, location</code>    |

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

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

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

Второй, *необязательный*, параметр задает 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.

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

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

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    | <code>limit_rate_after размер;</code>              |
| По умолчанию | <code>limit_rate_after 0;</code>                   |
| Контекст     | <code>http, server, location, if в location</code> |

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    | <code>lingering_close on   always   off;</code> |
| По умолчанию | <code>lingering_close on;</code>                |
| Контекст     | <code>http, server, location</code>             |

Управляет закрытием соединений с клиентами.

|                     |                                                                                                                                                                                                                 |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>on</code>     | Angie будет <i>ждать</i> и <i>обрабатывать</i> дополнительные данные, поступающие от клиента, перед полным закрытием соединения, но только если эвристика указывает на то, что клиент может ещё послать данные. |
| <code>always</code> | Angie всегда будет ждать и обрабатывать дополнительные данные, поступающие от клиента.                                                                                                                          |
| <code>off</code>    | 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    | <code>lingering_time время;</code>  |
| По умолчанию | <code>lingering_time 30s;</code>    |
| Контекст     | <code>http, server, location</code> |

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    | <code>lingering_timeout время;</code> |
| По умолчанию | <code>lingering_timeout 5s;</code>    |
| Контекст     | <code>http, server, location</code>   |

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*.

## listen

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

Задает *адрес* и *порт* для слушающего сокета или путь для UNIX-сокета, на которых сервер будет принимать запросы. Можно указать *адрес* и *порт*, либо только *адрес* или только *порт*. Кроме того, *адрес* 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 задаются в квадратных скобках:

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

UNIX-сокеты задаются при помощи префикса `unix::`:

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

Если указан только *адрес*, то используется порт *80*.

Если директива не указана, то используется либо *\*:80*, если Angie работает с привилегиями суперпользователя, либо *\*:8000*.

|                             |                                                                                                                                                                                                                                                                                                                                  |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <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. |
| <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, начиная с версии 1.2.0. 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 <i>HTTP3</i> . 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. Эти параметры можно задать в любой директиве `listen`, но только один раз для слушающего сокета.

|                             |                                                                                                                                                                   |
|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>setfib=число</code>   | задает таблицу маршрутизации, FIB (параметр <code>SO_SETFIB</code> ) для слушающего сокета. В настоящий момент 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. |

### Осторожно

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 <i>backlog</i> в вызове <i>listen()</i> , который ограничивает максимальный размер очереди ожидающих приема соединений. По умолчанию <i>backlog</i> устанавливается равным <i>-1</i> для FreeBSD, DragonFly BSD и macOS, и <i>511</i> для других платформ.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <code>rcvbuf=размер</code>     | задает размер буфера приема (параметр <i>SO_RCVBUF</i> ) для слушающего сокета.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <code>sndbuf=размер</code>     | задает размер буфера передачи (параметр <i>SO_SNDBUF</i> ) для слушающего сокета.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <code>accept_filter=фи</code>  | задает название accept-фильтра (параметр <i>SO_ACCEPTFILTER</i> ) для слушающего сокета, который включается для фильтрации входящих соединений перед передачей их в <i>accept()</i> . Работает только на FreeBSD и NetBSD 5.0+. Можно использовать два фильтра: <i>dataready</i> и <i>httpready</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <code>deferred</code>          | указывает использовать отложенный <i>accept()</i> (параметр <i>TCP_DEFER_ACCEPT</i> сокета) на Linux.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <code>bind</code>              | 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>setfib</i> , <i>fastopen</i> , <i>backlog</i> , <i>rcvbuf</i> , <i>sndbuf</i> , <i>accept_filter</i> , <i>deferred</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> . |
| <code>ipv6only=on   off</code> | 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.<br>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 <i>SO_REUSEPORT</i> для Linux 3.9+ и DragonFly BSD или <i>SO_REUSEPORT_LB</i> 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+.                                                                                                                                                                                                                                                                                                                                                                                                                                              |

### Осторожно

Ненадлежащее использование параметра *reuseport* может быть небезопасно.

`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 <i>SO_KEEPALIVE</i>                                |
| <code>off</code> | 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\_KEEPIIDLE*) в 30 минут, для интервала проб (*TCP\_KEEPIINTVL*) будет действовать стандартная системная настройка, а счетчик проб (*TCP\_KEEPCNT*) будет равен 10.

Пример:

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

## location

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

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

Для сопоставления используется URI запроса в нормализованном виде, 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:

|    |                                                   |
|----|---------------------------------------------------|
| ~* | Для поиска совпадения без учета регистра символов |
| ~  | 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.

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 {  
    #...  
}
```

### **Осторожно**

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 число;</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 число;</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><br><code>open_file_cache max=<i>N</i> [<i>inactive=время</i>];</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`.

|                 |                                                                                                                                         |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| <b>max</b>      | задает максимальное число элементов в кэше; при переполнении кэша удаляются наименее востребованные элементы (LRU)                      |
| <b>inactive</b> | задает время, после которого элемент кэша удаляется, если к нему не было обращений в течение этого времени.<br>По умолчанию: 60 секунд. |
| <b>off</b>      | 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     | <code>http, server, location</code>           |

Разрешает или запрещает кэширование ошибок поиска файлов в *open\_file\_cache*.

### 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     | <code>http, server, location</code>          |

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 время;</code> |
| По умолчанию | <code>open_file_cache_valid 60s;</code>   |
| Контекст     | <code>http, server, location</code>       |

Определяет время, через которое следует проверять актуальность информации об элементе в *open\_file\_cache*.

### output\_buffers

|              |                                                         |
|--------------|---------------------------------------------------------|
| Синтаксис    | <code>output_buffers <i>число</i> <i>размер</i>;</code> |
| По умолчанию | <code>output_buffers 2 32k;</code>                      |
| Контекст     | <code>http, server, location</code>                     |

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     | <code>http, server, location</code>     |

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 <i>размер</i>;</code> |
| По умолчанию | <code>postpone_output 1460;</code>          |
| Контекст     | <code>http, server, location</code>         |

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 <i>размер</i>;</code> |
| По умолчанию | <code>read_ahead 0;</code>             |
| Контекст     | <code>http, server, location</code>    |

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 `fentl(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     | <code>http, server, location</code>          |

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     | <code>http, server</code>              |

Позволяет производить точную настройку выделений 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     | <code>http, server, location</code>             |

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 ... [valid=время] [ipv4=on   off] [ipv6=on   off] [status_zone=зона];</code> |
| По умолчанию | —                                                                                                 |
| Контекст     | <code>http, server, location, upstream</code>                                                     |

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.

По умолчанию Angie кэширует ответы, используя значение TTL из ответа.

|              |                                                                                  |
|--------------|----------------------------------------------------------------------------------|
| <b>valid</b> | <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.

|                 |                              |
|-----------------|------------------------------|
| <b>ipv4=off</b> | запрещает поиск IPv4-адресов |
| <b>ipv6=off</b> | запрещает поиск IPv6-адресов |

|                    |                                                                                                                                                |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>status_zone</b> | <i>необязательный</i> параметр; включает сбор метрик запросов и ответов DNS-сервера ( <i>/status/resolvers/&lt;зона&gt;</i> ) в указанной зоне |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------|

#### Совет

Для предотвращения DNS-спуфинга рекомендуется использовать DNS-серверы в защищенной доверенной локальной сети.

#### Подсказка

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     | http, server, location, upstream     |

Задаёт таймаут для преобразования имени в адрес, например:

```
resolver_timeout 5s;
```

## root

|              |                                       |
|--------------|---------------------------------------|
| Синтаксис    | <code>root <i>путь</i>;</code>        |
| 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`.

## satisfy

|              |                                 |
|--------------|---------------------------------|
| Синтаксис    | <code>satisfy all   any;</code> |
| По умолчанию | <code>satisfy all;</code>       |
| 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    | <code>send_lowat <i>размер</i>;</code> |
| По умолчанию | <code>send_lowat 0;</code>             |
| 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 `queue`, либо при помощи параметра сокета `SO_SNDLOWAT`. В обоих случаях будет использован указанный размер.



## send\_timeout

|              |                                          |
|--------------|------------------------------------------|
| Синтаксис    | <code>send_timeout</code> <i>время</i> ; |
| По умолчанию | <code>send_timeout 60s</code> ;          |
| 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    | <code>sendfile on   off</code> ;      |
| По умолчанию | <code>sendfile off</code> ;           |
| 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 128К байт файла, однако при последующих чтениях файл подгружается частями только по 16К. Изменить это можно с помощью директивы `read_ahead`.

## sendfile\_max\_chunk

|              |                                                 |
|--------------|-------------------------------------------------|
| Синтаксис    | <code>sendfile_max_chunk</code> <i>размер</i> ; |
| По умолчанию | <code>sendfile_max_chunk 2m</code> ;            |
| Контекст     | http, server, location                          |

Ограничивает объём 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     | <code>http</code>           |

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     | <code>server</code>               |

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 выделения, которые могут затем использоваться в других директивах:

```
server {
    server_name ~^(www\.)?(.+)$;
```

```
location / {
    root /sites/$2;
}

server {
    server_name _;

    location / {
        root /sites/default;
    }
}
```

Именованные выделения 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.

**⚠ Внимание**

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 <i>размер</i>;</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 <i>размер</i>;</code> |
| По умолчанию | <code>server_names_hash_max_size 512;</code>           |
| Контекст     | <code>http</code>                                      |

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     | <code>http, server, location</code>                   |

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 `configure`.

Добавлено в версии 1.1.0: PRO

В Angie PRO 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     | <code>server, location, if в location</code>                  |

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.

### subrequest\_output\_buffer\_size

|              |                                                           |
|--------------|-----------------------------------------------------------|
| Синтаксис    | <code>subrequest_output_buffer_size <i>размер</i>;</code> |
| По умолчанию | <code>subrequest_output_buffer_size 4k   8k;</code>       |
| Контекст     | <code>http, server, 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 *4K*, или *8K*, 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 включаются 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.

## try\_files

|              |                                                                               |
|--------------|-------------------------------------------------------------------------------|
| Синтаксис    | <code>try_files файл ... uri;</code><br><code>try_files файл ... =код;</code> |
| По умолчанию | —                                                                             |
| Контекст     | <code>server, location</code>                                                 |

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, то делается внутреннее перенаправление на `uri`, заданный последним параметром. Например:

```
location /images/ {
    try_files $uri /images/default.gif;
}

location = /images/default.gif {
    expires 30s;
}
```

Последний параметр может также указывать на именованный `location`, может также быть кодом:

```
location / {
    try_files $uri $uri/index.html $uri.html =404;
}
```

В следующем примере директива `try_files`

```
location / {
    try_files $uri $uri/ @drupal;
}
```

аналогична директивам

```
location / {
    error_page 404 = @drupal;
    log_not_found off;
}
```

А здесь

```
location ~ /\.php$ {
    try_files $uri @drupal;

    fastcgi_pass ...;

    fastcgi_param SCRIPT_FILENAME /path/to$fastcgi_script_name;

# ...
}
```

*try\_files* проверяет существование PHP-файла, прежде чем передать запрос FastCGI-серверу.

```
location / {
    try_files /system/maintenance.html
            $uri $uri/index.html $uri.html
            @mongrel;
}

location @mongrel {
    proxy_pass http://mongrel;
}
```

```
location / {
    try_files $uri $uri/ @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
}
```

```
location / {
    try_files $uri $uri/ @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 определенного `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 размер;</code> |
| 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 <i>размер</i>;</code> |
| По умолчанию | <code>types_hash_max_size 1024;</code>          |
| Контекст     | <code>http, server, location</code>             |

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     | <code>http, server</code>                     |

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 <i>размер</i>;</code> |
| По умолчанию | <code>variables_hash_bucket_size 64;</code>            |
| Контекст     | <code>http</code>                                      |

Задаёт размер корзины в хэш-таблице переменных. Подробнее настройка хэш-таблиц обсуждается отдельно.

### variables\_hash\_max\_size

|              |                                                     |
|--------------|-----------------------------------------------------|
| Синтаксис    | <code>variables_hash_max_size <i>размер</i>;</code> |
| По умолчанию | <code>variables_hash_max_size 1024;</code>          |
| Контекст     | <code>http</code>                                   |

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

`$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

`$limit_rate`

установка этой переменной позволяет ограничивать скорость передачи ответа, см. *limit\_rate*

`$msec`

текущее время в секундах с точностью до миллисекунд

`$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 (включая строку запроса, заголовок и тело запроса)

`$request_method`

метод запроса, обычно `GET` или `POST`

`$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)

`$scheme`

схема запроса, "http" или "https"

`$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

`$time_iso8601`

локальное время в формате по стандарту ISO 8601

`$time_local`

локальное время в Common Log Format

`$tcpinfo_rtt`, `$tcpinfo_rttvar`, `$tcpinfo_snd_cwnd`, `$tcpinfo_rcv_space`

информация о клиентском TCP-соединении; доступна на системах, поддерживающих параметр сокета TCP\_INFO

`$uri`

Текущий URI запроса в *нормализованном* виде.

Значение `$uri` может изменяться в процессе обработки запроса, например, при внутренних перенаправлениях или при использовании индексных файлов.

### 3.3.3 Поточковый 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.

## 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:

|                      |                                                                                                                                                                                                                                                                                                                         |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>delete</code>  | удаляет описанную сеть                                                                                                                                                                                                                                                                                                  |
| <code>default</code> | значение переменной, если адрес клиента не соответствует ни одному из заданных адресов. При задании адресов в формате CIDR вместо <code>default</code> можно использовать " <code>0.0.0.0/0</code> " и " <code>::/0</code> ". Если параметр <code>default</code> не указан, значением по умолчанию будет пустая строка. |
| <code>include</code> | включает файл с адресами и значениями. Включений может быть несколько.                                                                                                                                                                                                                                                  |
| <code>ranges</code>  | указывает, что адреса задаются в виде диапазонов. Этот параметр должен быть первым. Для ускорения загрузки гео-базы нужно располагать адреса в порядке возрастания.                                                                                                                                                     |

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;
}
```

## GeoIP

Создает переменные, значения которых зависят от IP-адреса клиента, используя готовые базы данных [MaxMind](#).

При использовании баз данных с поддержкой IPv6 IPv4-адреса ищутся отображенными на IPv6.

При сборке из исходного кода модуль необходимо включить с помощью параметра сборки `--with-stream_geoip_module`.

### Важно

Для этого модуля нужна библиотека [MaxMind GeoIP](#).

## Пример конфигурации

```
stream {
  geoip_country      GeoIP.dat;
  geoip_city         GeoLiteCity.dat;

  map $geoip_city_continent_code $nearest_server {
    default          example.com;
    EU               eu.example.com;
    NA               na.example.com;
    AS               as.example.com;
  }
# ...
}
```

## Директивы

### geoip\_country

|              |                             |
|--------------|-----------------------------|
| Синтаксис    | geoip_country <i>файл</i> ; |
| По умолчанию | —                           |
| Контекст     | stream                      |

Задаёт базу данных для определения страны в зависимости от значения IP-адреса клиента. При использовании этой базы данных доступны следующие переменные:

|                   |                                                                   |
|-------------------|-------------------------------------------------------------------|
| \$geoip_country_c | двухбуквенный код страны, например, "RU", "US".                   |
| \$geoip_country_c | трехбуквенный код страны, например, "RUS", "USA".                 |
| \$geoip_country_n | название страны, например, "Russian Federation", "United States". |

### geoip\_city

|              |                          |
|--------------|--------------------------|
| Синтаксис    | geoip_city <i>файл</i> ; |
| По умолчанию | —                        |
| Контекст     | stream                   |

Задаёт базу данных для определения страны, региона и города в зависимости от значения IP-адреса клиента. При использовании этой базы данных доступны следующие переменные:

|                   |                                                                                                                                               |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| \$geoip_city_cont | двухбуквенный код континента, например, "EU", "NA".                                                                                           |
| \$geoip_city_coun | двухбуквенный код страны, например, "RU", "US".                                                                                               |
| \$geoip_city_coun | трехбуквенный код страны, например, "RUS", "USA".                                                                                             |
| \$geoip_city_coun | название страны, например, "Russian Federation", "United States".                                                                             |
| \$geoip_dma_code  | DMA-код региона в США (также известный как "код агломерации"), согласно <a href="#">геотаргетингу Google AdWords API</a> .                    |
| \$geoip_latitude  | широта.                                                                                                                                       |
| \$geoip_longitude | долгота.                                                                                                                                      |
| \$geoip_region    | двухсимвольный код региона страны (область, край, штат, провинция, федеральная земля и тому подобное), например, "48", "DC".                  |
| \$geoip_region_na | название региона страны (область, край, штат, провинция, федеральная земля и тому подобное), например, "Moscow City", "District of Columbia". |
| \$geoip_city      | название города, например, "Moscow", "Washington".                                                                                            |
| \$geoip_postal_co | почтовый индекс.                                                                                                                              |

## geoip\_org

|              |                         |
|--------------|-------------------------|
| Синтаксис    | geoip_org <i>файл</i> ; |
| По умолчанию | —                       |
| Контекст     | stream                  |

Задаёт базу данных для определения названия организации в зависимости от значения IP-адреса клиента. При использовании этой базы данных доступна следующая переменная:

|                          |                                                                |
|--------------------------|----------------------------------------------------------------|
| <code>\$geoip_org</code> | название организации, например, "The University of Melbourne". |
|--------------------------|----------------------------------------------------------------|

## JS

Позволяет задавать обработчики на `njs` — подмножестве языка JavaScript.

В наших репозиториях модуль собран динамически и доступен отдельным пакетом `angie-module-njs` или `angie-pro-module-njs`; подключить его можно с помощью директивы `load_module`.

### 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

|              |                                                                 |
|--------------|-----------------------------------------------------------------|
| Синтаксис    | <code>js_access</code> <i>функция</i>   <i>модуль.функция</i> ; |
| По умолчанию | —                                                               |
| Контекст     | stream, server                                                  |

Задаёт функцию `цjs`, 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:

|                |                             |
|----------------|-----------------------------|
| <code>s</code> | 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\_fetch\_buffer\_size

|              |                                                   |
|--------------|---------------------------------------------------|
| Синтаксис    | <code>js_fetch_buffer_size</code> <i>размер</i> ; |
| По умолчанию | <code>js_fetch_buffer_size</code> 16k;            |
| Контекст     | stream, server                                    |

Задаёт размер буфера, который будет использоваться для чтения и записи для *Fetch API*.

### js\_fetch\_ciphers

|              |                                                 |
|--------------|-------------------------------------------------|
| Синтаксис    | <code>js_fetch_ciphers</code> <i>шифры</i> ;    |
| По умолчанию | <code>js_fetch_ciphers</code> 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

|              |                                                   |
|--------------|---------------------------------------------------|
| Синтаксис    | js_fetch_max_response_buffer_size <i>размер</i> ; |
| По умолчанию | js_fetch_max_response_buffer_size 1m;             |
| Контекст     | stream, server                                    |

Задаёт максимальный размер ответа, полученного при помощи [Fetch API](#).

### js\_fetch\_protocols

|              |                                                           |
|--------------|-----------------------------------------------------------|
| Синтаксис    | js_fetch_protocols [TLSv1] [TLSv1.1] [TLSv1.2] [TLSv1.3]; |
| По умолчанию | js_fetch_protocols TLSv1 TLSv1.1 TLSv1.2;                 |
| Контекст     | stream, server                                            |

Разрешает указанные протоколы для HTTPS-соединений при помощи [Fetch API](#).

### js\_fetch\_timeout

|              |                                 |
|--------------|---------------------------------|
| Синтаксис    | js_fetch_timeout <i>время</i> ; |
| По умолчанию | js_fetch_timeout 60s;           |
| Контекст     | 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

|              |                                            |
|--------------|--------------------------------------------|
| Синтаксис    | js_fetch_trusted_certificate <i>файл</i> ; |
| По умолчанию | —                                          |
| Контекст     | stream, server                             |

Задаёт файл с доверенными сертификатами CA в формате PEM, используемыми при проверке HTTPS-сертификата при помощи [Fetch API](#).

### js\_fetch\_verify

|              |                                        |
|--------------|----------------------------------------|
| Синтаксис    | <code>js_fetch_verify on   off;</code> |
| По умолчанию | <code>js_fetch_verify on;</code>       |
| Контекст     | <code>stream, server</code>            |

Разрешает или запрещает проверку сертификата HTTPS-сервера при помощи [Fetch API](#).

### js\_fetch\_verify\_depth

|              |                                                  |
|--------------|--------------------------------------------------|
| Синтаксис    | <code>js_fetch_verify_depth <i>число</i>;</code> |
| По умолчанию | <code>js_fetch_verify_depth 100;</code>          |
| Контекст     | <code>stream, server</code>                      |

Устанавливает глубину проверки в цепочке HTTPS-сертификатов при помощи [Fetch API](#).

### js\_filter

|              |                                                                |
|--------------|----------------------------------------------------------------|
| Синтаксис    | <code>js_filter <i>функция</i>   <i>модуль.функция</i>;</code> |
| По умолчанию | —                                                              |
| Контекст     | <code>stream, server</code>                                    |

Задаёт фильтр данных. Можно ссылаться на функцию модуля.

Функция фильтра вызывается однократно при первом достижении сессией *content-фазы*. Функция фильтра вызывается со следующими аргументами:

|          |                                      |
|----------|--------------------------------------|
| <b>s</b> | объект <a href="#">stream-сессии</a> |
|----------|--------------------------------------|

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

|              |                                                                 |
|--------------|-----------------------------------------------------------------|
| Синтаксис    | <code>js_import модуль.js   имя_экспорта from модуль.js;</code> |
| По умолчанию | —                                                               |
| Контекст     | stream, server                                                  |

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 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

|              |                            |
|--------------|----------------------------|
| Синтаксис    | <code>js_path путь;</code> |
| По умолчанию | —                          |
| Контекст     | stream, server             |

Задаёт дополнительный путь для модулей pjs.

## js\_preload\_object

|              |                                                               |
|--------------|---------------------------------------------------------------|
| Синтаксис    | <code>js_preload_object имя.json   имя from файл.json;</code> |
| 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 pjs. 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    | <code>js_preread <i>функция</i>   <i>модуль.функция</i>;</code> |
| По умолчанию | —                                                               |
| Контекст     | stream, server                                                  |

Задаёт функцию `njs`, которая будет вызываться в *preread-фазе*. Можно сослаться на функцию модуля.

Функция вызывается однократно при первом достижении сессией *preread-фазы*. Функция вызывается со следующими аргументами:

|          |                             |
|----------|-----------------------------|
| <b>s</b> | 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

|              |                                                                          |
|--------------|--------------------------------------------------------------------------|
| Синтаксис    | <code>js_set <i>\$переменная функция</i>   <i>модуль.функция</i>;</code> |
| По умолчанию | —                                                                        |
| Контекст     | stream, server                                                           |

Задаёт функцию `njs` для указанной переменной. Можно сослаться 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.

### 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

|              |                                                                                                                |
|--------------|----------------------------------------------------------------------------------------------------------------|
| Синтаксис    | <code>js_shared_dict_zone zone=<i>имя:размер</i> [timeout=<i>время</i>] [type=string   number] [evict];</code> |
| По умолчанию | —                                                                                                              |
| Контекст     | 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 ( <code>number</code> ), по умолчанию в качестве ключа и значения используется строка ( <code>string</code> ) |
| 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                                                                                             |

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;
```

```
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    | <code>js_var \$переменная [значение];</code> |
| 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`.

### 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    | <code>limit_conn_dry_run on   off;</code> |
| По умолчанию | <code>limit_conn_dry_run off;</code>      |
| Контекст     | <code>stream, server</code>               |

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>stream, server</code>                                     |

Задаёт желаемый уровень записи в лог случаев ограничения числа соединений.

### limit\_conn\_zone

|              |                                                                         |
|--------------|-------------------------------------------------------------------------|
| Синтаксис    | <code>limit_conn_zone <i>ключ</i> zone = <i>название:размер</i>;</code> |
| По умолчанию | —                                                                       |
| Контекст     | <code>stream</code>                                                     |

Задаёт параметры зоны разделяемой памяти, которая хранит состояние для разных значений ключа. Состояние в частности содержит текущее число соединений. В качестве ключа может использоваться текст, переменные и их комбинации. 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><br><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.

#### Осторожно

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;
```

**Важно**

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 такимилогами;
- 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 *"\d"*, *"\t"*, *"\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:

|                       |                                                                                                                                                                                               |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>max</code>      | задаёт максимальное число дескрипторов в кэше; при переполнении кэша наименее востребованные (LRU) дескрипторы закрываются                                                                    |
| <code>inactive</code> | задаёт время, после которого кэшированный дескриптор закрывается, если к нему не было обращений в течение этого времени; по умолчанию 10 секунд                                               |
| <code>min_uses</code> | задаёт минимальное число использований файла в течение времени, заданного параметром <code>inactive</code> , после которого дескриптор файла будет оставаться открытым в кэше; по умолчанию 1 |
| <code>valid</code>    | задаёт, через какое время нужно проверять, что файл ещё существует под тем же именем; по умолчанию 60 секунд                                                                                  |
| <code>off</code>      | запрещает кэш                                                                                                                                                                                 |

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 $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                                        |

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 выделения, 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:

|                                      |                                                                                                                                                                                                           |
|--------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>default</code> <i>значение</i> | задает результирующее значение, если исходное значение не совпадает ни с одним из перечисленных. Если параметр <code>default</code> не указан, результирующим значением по умолчанию будет пустая строка. |
| <code>hostnames</code>               | 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;
```

|                                  |                                                              |
|----------------------------------|--------------------------------------------------------------|
| <code>include</code> <i>файл</i> | включает файл со значениями. Включений может быть несколько. |
| <code>volatile</code>            | 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 <i>размер</i>;</code> |
| 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    | <code>map_hash_max_size <i>размер</i>;</code> |
| По умолчанию | <code>map_hash_max_size 2048;</code>          |
| Контекст     | <code>stream</code>                           |

Задаёт максимальный размер хэш-таблиц для переменных *map*. Подробнее настройка хэш-таблиц обсуждается отдельно.

## 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.

При сборке из исходного кода модуль необходимо включить с помощью параметра сборки `--with-stream_mqtt_preread_module`. В пакетах и образах из наших репозиториях модуль включен в сборку.

### 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 модуль *HTTP*, *поточковый* или *почтовый* модули.

### Пример конфигурации

После того, как модуль `stream` завершит обработку SSL/TLS, он передает соединение в модуль `http`:

```
http {
    server {
        listen 8000;

        location / {
            root html;
        }
    }
}

stream {
    server {
        listen 12345 ssl;
    }
}
```

```

    ssl_certificate    domain.crt;
    ssl_certificate_key domain.key;

    pass 127.0.0.1:8000;
}
}

```

## 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;
```

## Прoxy

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 допустимо использование переменных. Специальное значение `off` отменяет действие унаследованной с предыдущего уровня конфигурации директивы `proxy_bind`, позволяя системе самостоятельно выбирать локальный IP-адрес.

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.

#### ❗ Важно

Необходимо настроить таблицу маршрутизации ядра для перехвата сетевого трафика с проксируемого сервера.

### proxy\_buffer\_size

|              |                                               |
|--------------|-----------------------------------------------|
| Синтаксис    | <code>proxy_buffer_size <i>размер</i>;</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 <i>время</i> ; |
| По умолчанию | proxy_connect_timeout 60s;           |
| Контекст     | stream, server                       |

Задаёт таймаут для установления соединения с проксируемым сервером.

### proxy\_connection\_drop

|              |                                                |
|--------------|------------------------------------------------|
| Синтаксис    | proxy_connection_drop <i>время</i>   on   off; |
| По умолчанию | proxy_connection_drop off;                     |
| Контекст     | http, server, location                         |

Настраивает завершение всех соединений с проксируемым сервером, если он был удален из группы или помечен как постоянно недоступный в результате процесса *resolve* 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\_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 <i>время</i> ; |
| 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 <i>число</i> ; |
| 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*.

Адрес можно также задать с помощью переменных:

```
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\_requests

|              |                                    |
|--------------|------------------------------------|
| Синтаксис    | <code>proxy_requests число;</code> |
| По умолчанию | <code>proxy_requests 0;</code>     |
| Контекст     | stream, server                     |

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 *тайм-аута*.

### proxy\_responses

|              |                                     |
|--------------|-------------------------------------|
| Синтаксис    | <code>proxy_responses число;</code> |
| По умолчанию | —                                   |
| Контекст     | stream, server                      |

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     | stream, server                                |

Конфигурирует поведение "TCP keealive" 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\_certificate

|              |                                                 |
|--------------|-------------------------------------------------|
| Синтаксис    | <code>proxy_ssl_certificate файл [файл];</code> |
| По умолчанию | —                                               |
| Контекст     | stream, server                                  |

Задаёт файл с сертификатом в формате PEM для аутентификации на проксируемом сервере. В имени файла можно использовать переменные.

Добавлено в версии 1.2.0.

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\_certificate\_key

|              |                                                        |
|--------------|--------------------------------------------------------|
| Синтаксис    | proxy_ssl_certificate_key <i>файл</i> [ <i>файл</i> ]; |
| По умолчанию | —                                                      |
| Контекст     | stream, server                                         |

Задаёт файл с секретным ключом в формате PEM для аутентификации на проксируемом сервере. В имени файла можно использовать переменные.

Добавлено в версии 1.2.0.

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    | proxy_ssl_ciphers <i>шифры</i> ; |
| По умолчанию | proxy_ssl_ciphers DEFAULT;       |
| 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`.

### proxy\_ssl\_conf\_command

|              |                                              |
|--------------|----------------------------------------------|
| Синтаксис    | proxy_ssl_conf_command <i>имя значение</i> ; |
| По умолчанию | —                                            |
| Контекст     | stream, server                               |

Задаёт произвольные конфигурационные команды OpenSSL при установлении соединения с проксируемым сервером.

#### Важно

Директива поддерживается при использовании OpenSSL 1.0.2 и выше.

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*.

#### Осторожно

Следует учитывать, что изменение настроек OpenSSL напрямую может привести к неожиданному поведению.

### proxy\_ssl\_crl

|              |                             |
|--------------|-----------------------------|
| Синтаксис    | proxy_ssl_crl <i>файл</i> ; |
| По умолчанию | —                           |
| Контекст     | stream, server              |

Указывает файл с отозванными сертификатами (CRL) в формате PEM, используемыми при *проверке* сертификата проксируемого сервера.

### proxy\_ssl\_name

|              |                                            |
|--------------|--------------------------------------------|
| Синтаксис    | proxy_ssl_name <i>имя</i> ;                |
| По умолчанию | proxy_ssl_name <i>хост из proxy_pass</i> ; |
| 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

Добавлено в версии 1.2.0.

|              |                          |
|--------------|--------------------------|
| Синтаксис    | proxy_ssl_ntls on   off; |
| По умолчанию | proxy_ssl_ntls off;      |
| Контекст     | stream, server           |

Включает клиентскую поддержку NTLS при использовании TLS библиотеки `TongSuo`.

```
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;
}
```

### Важно

Angie необходимо собрать с использованием параметра конфигурации `--with-ntls`, с соответствующей SSL библиотекой с поддержкой NTLS

```
./configure --with-openssl=../Tongsuo-8.3.0 \
            --with-openssl-opt=enable-ntls \
            --with-ntls
```

## 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                                                             |

Изменено в версии 1.2.0: Параметр `TLSv1.3` добавлен к используемым по умолчанию.

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 CA 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\_timeout

|              |                              |
|--------------|------------------------------|
| Синтаксис    | proxy_timeout <i>время</i> ; |
| 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.

### upstream\_probe\_timeout (PRO)

Добавлено в версии 1.4.0: PRO

|              |                                       |
|--------------|---------------------------------------|
| Синтаксис    | upstream_probe_timeout <i>время</i> ; |
| По умолчанию | upstream_probe_timeout 50s;           |
| Контекст     | server                                |

Задаёт максимальное время бездействия установленного с сервером соединения для проверок, настроенных с помощью директивы *upstream\_probe (PRO)*; при превышении этого предела соединение будет закрыто.

### 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:

```
proxy_upload_rate $rate;
```

```
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.

При сборке из исходного кода модуль необходимо включить с помощью параметра сборки `--with-stream_rdp_preread_module`. В пакетах и образах из наших репозиториях модуль включен в сборку.

### 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*.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-stream_realip_module`. В пакетах и образах из наших репозиториях модуль включен в сборку.

### 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 <i>адрес</i>   <i>CIDR</i>   <i>unix</i>::;</code> |
| По умолчанию | —                                                                         |
| Контекст     | stream, server                                                            |

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 текст, переменные и их комбинации.

## Set

Позволяет устанавливать значение переменной.

### Пример конфигурации

```
server {
    listen 12345;
    set    $true 1;
}
```

## Директивы

### set

|              |                                         |
|--------------|-----------------------------------------|
| Синтаксис    | <code>set \$переменная значение;</code> |
| По умолчанию | —                                       |
| Контекст     | server                                  |

Устанавливает значение указанной переменной. В качестве значения можно использовать текст, переменные и их комбинации.

## Split Clients

Модуль генерирует переменные для A/B-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.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-stream_ssl_module`.

В пакетах и образах из наших репозиториях модуль включен в сборку.

### 📌 Важно

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 протокол ...;</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    | <code>ssl_certificate <i>файл</i>;</code> |
| 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.

**Важно**

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.

Неадекватное использование подобного синтаксиса может быть небезопасно, например данные секретного ключа могут попасть в *лог ошибок*.

### ssl\_certificate\_key

|              |                                        |
|--------------|----------------------------------------|
| Синтаксис    | <code>ssl_certificate_key файл;</code> |
| По умолчанию | —                                      |
| Контекст     | stream, server                         |

Указывает файл с секретным ключом в формате PEM для данного виртуального сервера.

#### Важно

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 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 шифры;</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`.

### ssl\_client\_certificate

|              |                                           |
|--------------|-------------------------------------------|
| Синтаксис    | <code>ssl_client_certificate файл;</code> |
| По умолчанию | —                                         |
| Контекст     | stream, server                            |

Задаёт 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 имя значение;</code> |
| По умолчанию | —                                           |
| Контекст     | stream, server                              |

Задаёт произвольные конфигурационные команды OpenSSL.

### Важно

Директива поддерживается при использовании OpenSSL 1.0.2 и выше.

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`.

### Осторожно

Изменение настроек OpenSSL напрямую может привести к неожиданному поведению.

## ssl\_crl

|              |                            |
|--------------|----------------------------|
| Синтаксис    | <code>ssl_crl файл;</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 файл;</code> |
| По умолчанию | —                              |
| Контекст     | stream, server                 |

Указывает файл с параметрами для DHE-шифров.

### Осторожно

По умолчанию параметры не заданы, и соответственно DHE-шифры не будут использоваться.

## ssl\_ecdh\_curve

|              |                                             |
|--------------|---------------------------------------------|
| Синтаксис    | <code>ssl_ecdh_curve</code> <i>кривая</i> ; |
| По умолчанию | <code>ssl_ecdh_curve auto</code> ;          |
| Контекст     | <code>stream, server</code>                 |

Задаёт кривую для ECDHE-шифров.

### Важно

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.

### Важно

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\_handshake\_timeout

|              |                                                   |
|--------------|---------------------------------------------------|
| Синтаксис    | <code>ssl_handshake_timeout</code> <i>время</i> ; |
| По умолчанию | <code>ssl_handshake_timeout 60s</code> ;          |
| Контекст     | <code>stream, server</code>                       |

Задаёт таймаут для завершения операции SSL-рукопожатия.

## ssl\_ocsp

|              |                                         |
|--------------|-----------------------------------------|
| Синтаксис    | <code>ssl_ocsp on   off   leaf</code> ; |
| По умолчанию | <code>ssl_ocsp off</code> ;             |
| Контекст     | <code>http, server</code>               |

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     | <code>http, server</code>                              |

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 | <code>—</code>                       |
| Контекст     | <code>http, server</code>            |

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\_ntls

Добавлено в версии 1.2.0.

|              |                                 |
|--------------|---------------------------------|
| Синтаксис    | <code>ssl_ntls on   off;</code> |
| По умолчанию | <code>ssl_ntls off;</code>      |
| Контекст     | <code>stream, server</code>     |

Включает серверную поддержку NTLS при использовании TLS библиотеки `TongSuo`

```
listen ... ssl;
ssl_ntls on;
```

#### Важно

Angie необходимо собрать с использованием параметра конфигурации `--with-ntls`, с соответствующей SSL библиотекой с поддержкой NTLS

```
./configure --with-openssl=../Tongsuo-8.3.0 \  

--with-openssl-opt=enable-ntls \  

--with-ntls
```

### ssl\_password\_file

|              |                                 |
|--------------|---------------------------------|
| Синтаксис    | 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.

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    | <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>stream, server</code>                                                       |

Изменено в версии 1.2.0: Параметр `TLSv1.3` добавлен к используемым по умолчанию.

Разрешает указанные протоколы.

### Важно

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>stream, server</code>                                                              |

Задаёт тип и размеры кэшей для хранения параметров сессий. Тип кэша может быть следующим:

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>off</code>     | жёсткое запрещение использования кэша сессий: Angie явно сообщает клиенту, что сессии не могут использоваться повторно.                                                                                                                                                                                                                                                                                                                                                          |
| <code>none</code>    | мягкое запрещение использования кэша сессий: Angie сообщает клиенту, что сессии могут использоваться повторно, но на самом деле не хранит параметры сессии в кэше.                                                                                                                                                                                                                                                                                                               |
| <code>builtin</code> | 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.                                                                                                                                                                                                                                              |
| <code>shared</code>  | 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 файл;</code> |
| По умолчанию | —                                         |
| Контекст     | stream, 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     | stream, 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     | stream, 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     | <code>http, server</code>           |

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`.

### Внимание

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     | <code>http, server</code>            |

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     | <code>http, server</code>                |

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 сертификата 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     | stream, 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     | stream, server                                                       |

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.

|                             |                                                                                                                                                                                                                              |
|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>optional</code>       | запрашивает клиентский сертификат, и если сертификат был предоставлен, проверяет его                                                                                                                                         |
| <code>optional_no_ca</code> | 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`

### Важно

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_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
```

#### **Важно**

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
```

#### **Важно**

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_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 Preread

Позволяет извлекать информацию из сообщения `ClientHello` без терминации TLS, например имя сервера, запрошенное через `SNI`, или протоколы, указанные в `ALPN`.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-stream-ssl_preread_module`. В пакетах и образах из наших репозиториях модуль включен в сборку.

## 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;
}
```

## 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_protocol`

Максимальная версия протокола SSL, поддерживаемая клиентом.

`$ssl_preread_server_name`

Имя сервера, запрошенное через SNI.

`$ssl_preread_alpn_protocols`

Список протоколов, переданный клиентом через ALPN. Значения разделяются запятыми.

## 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

### upstream

|              |                                   |
|--------------|-----------------------------------|
| Синтаксис    | <code>upstream имя { ... }</code> |
| По умолчанию | —                                 |
| Контекст     | stream                            |

Описывает группу серверов. Серверы могут слушать на разных портах. Кроме того, можно одновременно использовать серверы, слушающие на TCP- и UNIX-сокетах.

Пример:

```
upstream backend {
    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.

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.

## 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 обязательного порта, или в виде пути UNIX-сокета, который указывается после префикса `unix:.` 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 <i>зоне</i> 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.

|                          |                            |
|--------------------------|----------------------------|
| <code>max_fails=1</code> | число попыток по умолчанию |
| <code>max_fails=0</code> | отключает учет попыток     |

`fail_timeout=время` — задаёт:

`fail_timeout=время` — задаёт 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.

|                     |                                                                                                                                                                                                                               |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>backup</code> | помечает сервер как запасной. На него будут передаваться запросы в случае, если не работают основные серверы.                                                                                                                 |
| <code>down</code>   | помечает сервер как постоянно недоступный.                                                                                                                                                                                    |
| <code>drain</code>  | 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> . |

### Осторожно

Параметр `backup` нельзя использовать совместно с методами балансировки нагрузки `hash` и `random`.

Параметры `down` и `drain` взаимно исключающие.

Добавлено в версии 1.3.0.

|                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>resolve</code>     | Позволяет отслеживать изменения списка IP-адресов, соответствующего доменному имени, и обновлять его без перезагрузки конфигурации. При этом группа должна находиться в <i>зоне разделяемой памяти</i> ; также должен быть определен <i>преобразователь имен в адреса</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <code>service=имя</code> | <p>Включает преобразование SRV-записей DNS и задает имя сервиса. При указании этого параметра необходимо также задать параметр <code>resolve</code>, не указывая порт сервера при имени хоста.</p> <p>Если в имени службы нет точек, формируется имя по стандарту RFC: к имени службы добавляется префикс <code>_</code>, затем через точку добавляется <code>_tcp</code>. Так, имя службы <code>http</code> даст в результате <code>_http._tcp</code>.</p> <p>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.</p> <ul style="list-style-type: none"> <li>• SRV-записи с наивысшим приоритетом (те, которые имеют минимальное значение приоритета) разрешаются как основные серверы, а прочие записи становятся запасными серверами. Если <code>backup</code> установлено с <code>server</code>, SRV-записи с наивысшим приоритетом разрешаются как запасные серверы, а прочие записи игнорируются.</li> <li>• Вес аналогичен параметру <code>weight</code> директивы <code>server</code>. Если вес задан как в самой директиве, так и в SRV-записи, используется вес, установленный в директиве.</li> </ul> |

В этом примере выполняется поиск записи `_http._tcp.backend.example.com`:

```
server backend.example.com service=http resolve;
```

Добавлено в версии 1.4.0.



`slow_start=время` задает время восстановления веса сервера, возвращающегося к работе при балансировке нагрузки методом *round-robin* или *least\_conn*.  
Если параметр задан и сервер после сбоя снова считается работающим с точки зрения *max\_fails* и *upstream\_probe (PRO)*, 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 (PRO)

Добавлено в версии 1.4.0: PRO

|              |                          |
|--------------|--------------------------|
| Синтаксис    | <code>state файл;</code> |
| По умолчанию | —                        |
| Контекст     | <code>upstream</code>    |

Указывает *файл*, где постоянно хранится список серверов апстрима. При установке из наших пакетов для хранения таких файлов специально создается каталог `/var/lib/angie/state/` (`/var/db/angie/state/` во FreeBSD) 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 `s_server`. Содержимое файла изменяется при любом изменении серверов в разделе `/config/stream/upstreams/` через API конфигурации. Файл считывается при запуске Angie или перезагрузке конфигурации.

#### Осторожно

Чтобы использовать директиву `state` в блоке `upstream`, в нем не должно быть директив `server`, но нужна зона разделяемой памяти (*zone*).

## 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.

## feedback (PRO)

Добавлено в версии 1.7.0: PRO

|              |                                                                                          |
|--------------|------------------------------------------------------------------------------------------|
| Синтаксис    | <code>feedback переменная [inverse] [factor=число] [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:

|                   |                                                                                                                                                                                                                                                                                                                                                                           |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>переменная</b> | 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 <code>inverse</code> и <code>factor</code> .                                             |
| <b>inverse</b>    | Если параметр задан, значение обратной связи интерпретируется наоборот: более низкие значения указывают на лучшую производительность.                                                                                                                                                                                                                                     |
| <b>factor</b>     | Коэффициент, по которому значение обратной связи учитывается при расчёте среднего. Допустимы целые числа от 0 до 99. По умолчанию — 90. Среднее рассчитывается по формуле <i>экспоненциального сглаживания</i> . 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. |
| <b>account</b>    | Указывает условную переменную, которая контролирует, как соединения учитываются при расчёте. Среднее значение обновляется с учетом 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 может использоваться текст, переменные и их комбинации. Пример использования:

```
hash $remote_addr;
```

Метод совместим с библиотекой Perl `Cache::Memcached`.

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` равным 160.

## least\_conn

|              |                          |
|--------------|--------------------------|
| Синтаксис    | <code>least_conn;</code> |
| По умолчанию | —                        |
| Контекст     | upstream                 |

Задаёт для группы метод балансировки нагрузки, при котором соединение передаётся серверу с наименьшим числом активных соединений, с учётом весов серверов. Если подходит сразу несколько серверов, они выбираются циклически (в режиме round-robin) с учётом их весов.

## least\_time (PRO)

|              |                                                                                                        |
|--------------|--------------------------------------------------------------------------------------------------------|
| 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.

|                         |                                                                    |
|-------------------------|--------------------------------------------------------------------|
| <code>connect</code>    | Директива учитывает среднее время установки соединения.            |
| <code>first_byte</code> | Директива использует среднее время получения первого байта ответа. |
| <code>last_byte</code>  | Директива использует среднее время получения полного ответа.       |

Добавлено в версии 1.7.0: PRO

|                      |                                                                                                                                                                                              |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>factor</code>  | Выполняет ту же функцию, что и <i>response_time_factor (PRO)</i> , и переопределяет его, если параметр задан.                                                                                |
| <code>account</code> | Указывает условную переменную, которая контролирует, какие соединения учитываются при расчёте. Среднее значение обновляется, только если условная переменная соединения не равна "" или "0". |

### Примечание

По умолчанию *активные проверки* не включаются в расчёт; 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.

## random

|              |                            |
|--------------|----------------------------|
| Синтаксис    | <code>random [two];</code> |
| По умолчанию | —                          |
| Контекст     | upstream                   |

Задаёт для группы метод балансировки нагрузки, при котором соединение передаётся случайно выбранному серверу, с учётом 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 передаётся на сервер с наименьшим количеством активных соединений.

## response\_time\_factor (PRO)

|              |                                          |
|--------------|------------------------------------------|
| Синтаксис    | <code>response_time_factor число;</code> |
| По умолчанию | <code>response_time_factor 90;</code>    |
| Контекст     | upstream                                 |

Задаёт для метода балансировки нагрузки `least_time (PRO)` 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 анстрима* в API.

### Примечание

При подсчете учитываются только успешные ответы; что считать неуспешным ответом, определяют директивы `proxy_next_upstream`.

## sticky

Добавлено в версии 1.6.0: Angie

Добавлено в версии 1.6.0: Angie PRO

|              |                                                                                                                                                           |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Синтаксис    | <code>sticky route \$variable...;</code><br><code>sticky learn zone=zone create=\$create_var1... lookup=\$lookup_var1... [connect] [timeout=time];</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`.

### ⚠ Внимание

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 переменные для маршрутизации. Чтобы выбрать сервер, куда направляется входящее соединение, используется первая непустая переменная; 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 {
        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` (PRO)

В этом режиме для привязки клиента к конкретному проксируемому серверу используется динамически генерируемый ключ; он более гибок, так как назначает серверы на ходу, хранит сеансы в зоне общей 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 общей 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.

Последующие соединения от клиентов, желающих использовать сессию, должны содержать ее идентификатор, причем так, чтобы он попал в непустую переменную, указанную с `lookup`; тогда его значение будет сопоставлено с сессиями в общей 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 получения заголовков ответа от проксируемого сервера. Без него сессия создается только после завершения обработки со.

В примере Angie создает и ищет сессии, используя переменную `$rdp_cookie`:

```
stream {
    upstream backend {
        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;
    }

    server {
        listen 127.0.0.1:3389;

        ssl_preread on;

        proxy_pass backend;
    }
}
```

### sticky\_strict

Добавлено в версии 1.6.0: Angie

Добавлено в версии 1.6.0: Angie PRO

|              |                                      |
|--------------|--------------------------------------|
| 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.

## sticky\_secret

Добавлено в версии 1.6.0: Angie

Добавлено в версии 1.6.0: Angie PRO

|              |                                    |
|--------------|------------------------------------|
| Синтаксис    | <code>sticky_secret строка;</code> |
| По умолчанию | —                                  |
| Контекст     | upstream                           |

Добавляет *строку* как соль в функцию MD5-хэширования для директивы *sticky* в режиме *route*. *Строка* может содержать переменные, например *\$remote\_addr*:

```
upstream backend {
    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
```

## Встроенные переменные

Модуль `stream_upstream` поддерживает следующие встроенные переменные:

### `$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_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.

|                   |                                                                                   |
|-------------------|-----------------------------------------------------------------------------------|
| <code>""</code>   | Соединение направлено в группу серверов, где привязка не используется.            |
| <code>NEW</code>  | Соединение не содержит информации о привязке к серверу.                           |
| <code>HIT</code>  | Соединение с привязкой направлено на желаемый сервер.                             |
| <code>MISS</code> | 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\n\n";
}

```

## Директивы

### upstream\_probe (PRO)

Добавлено в версии 1.4.0: PRO

|              |                                                                                                                                                                                                                                                                     |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Синтаксис    | <code>upstream_probe <i>имя</i> [port=<i>число</i>] [interval=<i>время</i>] [test=<i>условие</i>] [essential [persistent]] [fails=<i>число</i>] [passes=<i>число</i>] [max_response=<i>размер</i>] [mode=always   idle   onfail] [udp] [send=<i>строка</i>];</code> |
| По умолчанию | —                                                                                                                                                                                                                                                                   |
| Контекст     | 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:

|                     |                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>имя</b>          | Обязательное имя проверки.                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <b>port</b>         | Альтернативный порт для запроса.                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>interval</b>     | Интервал между проверками. По умолчанию — 5s.                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>test</b>         | Проверяемое при запросе условие; задается строкой из переменных. Если результат подстановки всех переменных — "" или "0", проверка не пройдена.                                                                                                                                                                                                                                                                                            |
| <b>essential</b>    | Если параметр задан, то изначально состояние сервера подлежит уточнению и клиентские запросы не передаются ему, пока проверка не будет пройдена.                                                                                                                                                                                                                                                                                           |
| <b>persistent</b>   | Установка этого параметра требует сначала включить <b>essential</b> ; серверы с <b>persistent</b> , работавшие до перезагрузки конфигурации, начинают получать запросы без необходимости сначала пройти эту проверку.                                                                                                                                                                                                                      |
| <b>fails</b>        | Число последовательных неуспешных запросов, при котором проверка считает сервер неработающим. По умолчанию — 1.                                                                                                                                                                                                                                                                                                                            |
| <b>passes</b>       | Число последовательных успешных запросов, при котором проверка считает сервер работающим. По умолчанию — 1.                                                                                                                                                                                                                                                                                                                                |
| <b>max_response</b> | Максимальный объем памяти для ответа. Если задано нулевое значение, ожидание ответа отключается. По умолчанию — 256k.                                                                                                                                                                                                                                                                                                                      |
| <b>mode</b>         | Режим проверки в зависимости от работоспособности серверов: <ul style="list-style-type: none"> <li>• <b>always</b> — серверы проверяются независимо от состояния;</li> <li>• <b>idle</b> — проверяются неработающие серверы, а также серверы, где с последнего клиентского запроса прошло время <b>interval</b>.</li> <li>• <b>onfail</b> — проверяются серверы только в неработающем состоянии.</li> </ul> По умолчанию — <b>always</b> . |
| <b>udp</b>          | Если параметр указан, используется протокол UDP. По умолчанию для проверок используется TCP.                                                                                                                                                                                                                                                                                                                                               |
| <b>send</b>         | Отправляемые для проверки данные; это может быть строка с префиксом <b>data:</b> или имя файла с данными (задается абсолютно или относительно каталога <code>/usr/local/angie/</code> ).                                                                                                                                                                                                                                                   |

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\n\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*.

### Встроенные переменные

Модуль `stream_upstream` поддерживает следующие встроенные переменные:

#### `$upstream_probe` (PRO)

Имя активной сейчас проверки *upstream\_probe*.

#### `$upstream_probe_response` (PRO)

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.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-stream`. В пакетах и образах из наших репозиториях модуль включен в сборку.

### 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

### listen

|              |                                                                                                                                                                                                                                                                                |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Синтаксис    | <code>listen адрес[:порт] [ssl] [udp] [proxy_protocol] [setfib=число] [fastopen=число] [backlog=число] [rcvbuf=размер] [sndbuf=размер] [accept_filter=фильтр] [deferred] [bind] [ipv6only=on   off] [reuseport] [so_keepalive=on off][keepidle]:[keepintvl]:[keepcnt]];</code> |
| По умолчанию | —                                                                                                                                                                                                                                                                              |
| Контекст     | server                                                                                                                                                                                                                                                                         |

Задаёт *адрес* и *порт* для сокета, на котором сервер будет принимать соединения. Можно указать только *порт*. 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;

```

UNIX-сокеты задаются префиксом `unix`:

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

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

```
listen 127.0.0.1:12345-12399;
listen 12345-12399;
```

### ⓘ Важно

Разные серверы должны слушать на разных парах *адрес:порт*.

|                             |                                                                                                                                                                                     |
|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <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> . |
| <code>proxy_protocol</code> | указывает на то, что все соединения, принимаемые на данном порту, должны использовать протокол PROXY.                                                                               |

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 <b>ограничивает</b> максимальную длину очереди соединений, которые еще не завершили процесс трехстороннего рукопожатия. |

### ⚠ Осторожно

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 <i>backlog</i> в вызове <i>listen()</i> , который ограничивает максимальный размер очереди ожидающих приема соединений. По умолчанию <i>backlog</i> устанавливается равным <i>-1</i> для FreeBSD, DragonFly BSD и macOS, и <i>511</i> для других платформ.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <code>rcvbuf=размер</code>     | задает размер буфера приема (параметр <i>SO_RCVBUF</i> ) для слушающего сокета.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <code>sndbuf=размер</code>     | задает размер буфера передачи (параметр <i>SO_SNDBUF</i> ) для слушающего сокета.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <code>accept_filter=фи</code>  | задает имя принимающего фильтра (параметр <i>SO_ACCEPTFILTER</i> ) для слушающего сокета, который фильтрует входящие соединения перед их передачей в <i>accept()</i> . Работает только на FreeBSD и NetBSD 5.0+. Допустимые значения: <i>dataready</i> и <i>httpready</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <code>deferred</code>          | указывает использовать отложенный <i>accept()</i> (параметр <i>TCP_DEFER_ACCEPT</i> ) на Linux.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <code>bind</code>              | 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>setfib</i> , <i>fastopen</i> , <i>backlog</i> , <i>rcvbuf</i> , <i>sndbuf</i> , <i>accept_filter</i> , <i>deferred</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> . |
| <code>ipv6only=on   off</code> | 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.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <code>reuseport</code>         | указывает, что нужно создавать отдельный слушающий сокет для каждого рабочего процесса (через параметр сокета <i>SO_REUSEPORT</i> для Linux 3.9+ и DragonFly BSD или <i>SO_REUSEPORT_LB</i> 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+.                                                                                                                                                                                                                                                                                                                                                                                                                                              |

### Осторожно

Ненадлежащее использование параметра *reuseport* может быть небезопасно.

`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 <i>SO_KEEPALIVE</i>                                |
| <code>off</code> | 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.

### preread\_buffer\_size

|              |                                                  |
|--------------|--------------------------------------------------|
| Синтаксис    | <code>preread_buffer_size</code> <i>размер</i> ; |
| По умолчанию | <code>preread_buffer_size</code> 16k;            |
| Контекст     | stream, server                                   |

Задаёт размер буфера *предварительного чтения*.

### preread\_timeout

|              |                                             |
|--------------|---------------------------------------------|
| Синтаксис    | <code>preread_timeout</code> <i>время</i> ; |
| По умолчанию | <code>preread_timeout</code> 30s;           |
| Контекст     | stream, server                              |

Задаёт время фазы *предварительного чтения*.

### proxy\_protocol\_timeout

|              |                                                    |
|--------------|----------------------------------------------------|
| Синтаксис    | <code>proxy_protocol_timeout</code> <i>время</i> ; |
| По умолчанию | <code>proxy_protocol_timeout</code> 30s;           |
| 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</code> <i>адрес</i> ... [ <code>valid=время</code> ] [ <code>ipv4=on</code>   <code>off</code> ] [ <code>ipv6=on</code>   <code>off</code> ] [ <code>status_zone=зона</code> ]; |
| По умолчанию | —                                                                                                                                                                                              |
| 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.

По умолчанию Angie кэширует ответы, используя значение TTL из ответа. Необязательный параметр `valid` позволяет это переопределить:

|                    |                                                                                  |
|--------------------|----------------------------------------------------------------------------------|
| <code>valid</code> | <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.

|                       |                              |
|-----------------------|------------------------------|
| <code>ipv4=off</code> | запрещает поиск IPv4-адресов |
| <code>ipv6=off</code> | запрещает поиск IPv6-адресов |

|                          |                                                                                                                                                      |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>status_zone</code> | <i>необязательный</i> параметр; включает сбор метрик запросов и ответов DNS-сервера ( <code>/status/resolvers/&lt;зона&gt;</code> ) в указанной зоне |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|

#### Совет

Для предотвращения DNS-спуфинга рекомендуется использовать DNS-серверы в защищенной доверенной локальной сети.

#### Подсказка

При запуске в Docker используйте соответствующий внутренний адрес DNS-сервера, например 127.0.0.11.

### resolver\_timeout

|              |                                              |
|--------------|----------------------------------------------|
| Синтаксис    | <code>resolver_timeout</code> <i>время</i> ; |
| По умолчанию | <code>resolver_timeout 30s</code> ;          |
| Контекст     | stream, server, upstream                     |

Задаёт таймаут для преобразования имени в адрес, например:

```
resolver_timeout 5s;
```

## server

|              |                             |
|--------------|-----------------------------|
| Синтаксис    | <code>server { ... }</code> |
| По умолчанию | —                           |
| Контекст     | stream                      |

Задаёт конфигурацию для сервера.

## 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.

Имена серверов могут включать звездочку (\*), заменяющую первую или последнюю часть имени:

```
server {
    server_name example.com *.example.com www.example.*;
}
```

Такие имена называются шаблонными именами.

Первые два примера, приведенные выше, можно объединить в один:

```
server {
    server_name .example.com;
}
```

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 захваты, 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)

### **Внимание**

Для TLS-соединений используйте модуль `SSL Preread`.

### `server_names_hash_bucket_size`

|              |                                                           |
|--------------|-----------------------------------------------------------|
| Синтаксис    | <code>server_names_hash_bucket_size <i>размер</i>;</code> |
| По умолчанию | <code>server_names_hash_bucket_size 32 64 128;</code>     |
| 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 <i>размер</i>;</code> |
| По умолчанию | <code>server_names_hash_max_size 512;</code>           |
| 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.

## 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.

## variables\_hash\_bucket\_size

|              |                                                        |
|--------------|--------------------------------------------------------|
| Синтаксис    | <code>variables_hash_bucket_size <i>размер</i>;</code> |
| По умолчанию | <code>variables_hash_bucket_size 64;</code>            |
| Контекст     | stream                                                 |

Задаёт размер корзины в хэш-таблице переменных. Подробнее настройка хэш-таблиц обсуждается отдельно.

## variables\_hash\_max\_size

|              |                                                     |
|--------------|-----------------------------------------------------|
| Синтаксис    | <code>variables_hash_max_size <i>размер</i>;</code> |
| По умолчанию | <code>variables_hash_max_size 1024;</code>          |
| Контекст     | stream                                              |

Задаёт максимальный размер хэш-таблиц переменных. Подробнее настройка хэш-таблиц обсуждается отдельно.

## Встроенные переменные

Модуль *stream core* 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

`$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 <i>определенных адресов клиентов</i> |
| 500 | внутренняя ошибка сервера                                                                  |
| 502 | плохой шлюз, например если невозможно выбрать сервер группы или сервер недоступен          |
| 503 | сервис недоступен, например при ограничении по <i>числу соединений</i>                     |



`$time_iso8601`

локальное время в формате по стандарту ISO 8601

`$time_local`

локальное время в Common Log Format

### 3.3.4 Почтовый модуль

#### Auth HTTP

##### 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 заголовок значение;</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    | auth_http_timeout <i>время</i> ; |
| По умолчанию | auth_http_timeout 60s;           |
| 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
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
```

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*.

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

### Директивы

#### imap\_auth

|              |                                          |
|--------------|------------------------------------------|
| Синтаксис    | <code>imap_auth <i>метод</i> ...;</code> |
| По умолчанию | <code>imap_auth plain;</code>            |
| Контекст     | mail, server                             |

Задаёт разрешенные методы аутентификации IMAP-клиентов. Поддерживаемые методы:

|                       |                                                                                                      |
|-----------------------|------------------------------------------------------------------------------------------------------|
| <code>plain</code>    | <code>LOGIN, AUTH=PLAIN</code>                                                                       |
| <code>login</code>    | <code>AUTH=LOGIN</code>                                                                              |
| <code>cram-md5</code> | <code>AUTH=CRAM-MD5</code> . Для работы этого метода пароль должен храниться в незашифрованном виде. |
| <code>external</code> | <code>AUTH=EXTERNAL</code>                                                                           |

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    | <code>imap_capabilities <i>расширение</i> ...;</code>   |
| По умолчанию | <code>imap_capabilities IMAP4 IMAP4rev1 UIDPLUS;</code> |
| 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 <i>размер</i> ; |
| 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

### Директивы

#### pop3\_auth

|              |                             |
|--------------|-----------------------------|
| Синтаксис    | pop3_auth <i>метод</i> ...; |
| По умолчанию | ipop3_auth plain;           |
| Контекст     | mail, server                |

Задаёт разрешенные 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                                                                          |

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

### Директивы

#### proxy\_buffer

|              |                                   |
|--------------|-----------------------------------|
| Синтаксис    | <code>proxy_buffer размер;</code> |
| По умолчанию | <code>proxy_buffer 4k 8k;</code>  |
| 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    | <code>proxy_pass_error_message on   off;</code> |
| По умолчанию | <code>proxy_pass_error_message off;</code>      |
| 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    | <code>proxy_protocol on   off;</code> |
| По умолчанию | <code>proxy_protocol off;</code>      |
| Контекст     | mail, server                          |

Включает протокол PROXY для соединений с бэкендом.

### proxy\_smtp\_auth

|              |                                        |
|--------------|----------------------------------------|
| Синтаксис    | <code>proxy_smtp_auth on   off;</code> |
| По умолчанию | <code>proxy_smtp_auth off;</code>      |
| 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\_timeout

|              |                                   |
|--------------|-----------------------------------|
| Синтаксис    | <code>proxy_timeout время;</code> |
| По умолчанию | <code>proxy_timeout 24h;</code>   |
| 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    | <code>xclient on   off;</code> |
| По умолчанию | <code>xclient on;</code>       |
| 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 адрес   CIDR   unix::;</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

### Директивы

#### smtp\_auth

|              |                                     |
|--------------|-------------------------------------|
| Синтаксис    | <code>smtp_auth метод ...;</code>   |
| По умолчанию | <code>smtp_auth plain login;</code> |
| Контекст     | mail, server                        |

Задаёт разрешенные методы аутентификации 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                                                                          |
| 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    | <code>smtp_capabilities расширение ...;</code> |
| 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    | <code>smtp_client_buffer размер;</code> |
| По умолчанию | <code>smtp_client_buffer 4k 8k;</code>  |
| 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    | <code>smtp_greeting_delay размер;</code> |
| По умолчанию | <code>smtp_greeting_delay 0;</code>      |
| Контекст     | mail, server                             |

Позволяет задать задержку перед отправкой SMTP-приветствия, чтобы отклонить клиентов, не дожидаящихся приветствия до начала отправки SMTP-команд.

## SSL

Обеспечивает работу почтового прокси-сервера по протоколу SSL/TLS.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-mail_ssl_module`. В пакетах и образах из наших репозиториях модуль включен в сборку.

#### **Важно**

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\_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 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`.

## ssl\_client\_certificate

|              |                                                  |
|--------------|--------------------------------------------------|
| Синтаксис    | <code>ssl_client_certificate <i>файл</i>;</code> |
| По умолчанию | —                                                |
| Контекст     | mail, server                                     |

Указывает файл с доверенными сертификатами CA 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    | <code>ssl_conf_command <i>имя значение</i>;</code> |
| По умолчанию | —                                                  |
| Контекст     | mail, server                                       |

Задаёт произвольные *конфигурационные команды* OpenSSL.

### Важно

Директива поддерживается при использовании OpenSSL 1.0.2 и выше.

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`.

### Осторожно

Изменение настроек OpenSSL напрямую может привести к неожиданному поведению.

## ssl\_crl

|              |                                   |
|--------------|-----------------------------------|
| Синтаксис    | <code>ssl_crl <i>файл</i>;</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.

#### **Осторожно**

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.

#### **Важно**

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.

#### **Важно**

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\_password\_file

|              |                                      |
|--------------|--------------------------------------|
| Синтаксис    | <code>ssl_password_file файл;</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    | ssl_prefer_server_ciphers on   off; |
| По умолчанию | ssl_prefer_server_ciphers off;      |
| Контекст     | mail, 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     | mail, server                                                         |

Изменено в версии 1.2.0: Параметр TLSv1.3 добавлен к используемым по умолчанию.

Разрешает указанные протоколы.

#### **Важно**

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     | mail, server                                                                             |

Задаёт тип и размеры кэшей для хранения параметров сессий. Тип кэша может быть следующим:

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>off</code>     | жёсткое запрещение использования кэша сессий: Angie явно сообщает клиенту, что сессии не могут использоваться повторно.                                                                                                                                                                                                                                                                                                                                                          |
| <code>none</code>    | мягкое запрещение использования кэша сессий: Angie сообщает клиенту, что сессии могут использоваться повторно, но на самом деле не хранит параметры сессии в кэше.                                                                                                                                                                                                                                                                                                               |
| <code>builtin</code> | 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.                                                                                                                                                                                                                                              |
| <code>shared</code>  | 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 файл;</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 <i>время</i>;</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.



|                             |                                                                                                                                                                                                                                                                                                                      |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>optional</code>       | запрашивает клиентский сертификат, и если сертификат был предоставлен, проверяет его                                                                                                                                                                                                                                 |
| <code>optional_no_ca</code> | 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, <i>посылаемых</i> на сервер аутентификации. |

### ssl\_verify\_depth

|              |                                      |
|--------------|--------------------------------------|
| Синтаксис    | <code>ssl_verify_depth число;</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                           |

|                   |                                                                          |
|-------------------|--------------------------------------------------------------------------|
| <code>on</code>   | разрешить использование команд STLS для POP3 и STARTTLS для IMAP и SMTP; |
| <code>off</code>  | запретить использование команд STLS и STARTTLS;                          |
| <code>only</code> | требовать предварительного перехода на TLS.                              |

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.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-mail`. В пакетах и образах из наших репозиториев модуль 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 8BITMIME 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

|              |                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Синтаксис    | <code>listen</code> <i>адрес[:порт]</i> [ <code>ssl</code> ] [ <code>proxy_protocol</code> ] [ <code>backlog=число</code> ]<br>[ <code>rcvbuf=размер</code> ] [ <code>sndbuf=размер</code> ] [ <code>bind</code> ] [ <code>ipv6only=on   off</code> ] [ <code>reuseport</code> ]<br>[ <code>so_keepalive=on off</code> ][ <code>keepidle</code> ]:[ <code>keepintvl</code> ]:[ <code>keepcnt</code> ]; |
| По умолчанию | —                                                                                                                                                                                                                                                                                                                                                                                                      |
| Контекст     | server                                                                                                                                                                                                                                                                                                                                                                                                 |

Задаёт *адрес* и *порт* для сокета, на котором сервер будет принимать соединения. Можно указать только *порт*. 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;
```

### Важно

Разные серверы должны слушать на разных парах *адрес:порт*.

|                             |                                                                                                                                                                                                                                      |
|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ssl</code>            | указывает на то, что все соединения, принимаемые на данном слушающем сокете, должны работать в режиме SSL.                                                                                                                           |
| <code>proxy_protocol</code> | 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.

|                                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|--------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <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>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>backlog</code> , <code>rcvbuf</code> , <code>sndbuf</code> , <code>ipv6only</code> , <code>reuseport</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.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

`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.

Разные серверы должны слушать на разных парах *адрес:порт*.

## mail

|              |                           |
|--------------|---------------------------|
| Синтаксис    | <code>mail { ... }</code> |
| По умолчанию | —                         |
| Контекст     | main                      |

Предоставляет контекст конфигурационного файла, в котором указываются директивы почтового сервера.

## max\_commands

Добавлено в версии 1.7.0.

|              |                                  |
|--------------|----------------------------------|
| Синтаксис    | <code>max_commands число;</code> |
| По умолчанию | <code>max_commands 1000;</code>  |
| Контекст     | mail, server                     |

Задаёт максимальное количество команд, отправляемых во время аутентификации, для усиления защиты от DoS-атак.

### max\_errors

|              |                                |
|--------------|--------------------------------|
| Синтаксис    | <code>max_errors число;</code> |
| По умолчанию | <code>max_errors 5;</code>     |
| Контекст     | mail, server                   |

Задаёт число ошибок протокола, по достижении которого соединение закрывается.

### protocol

|              |                                           |
|--------------|-------------------------------------------|
| Синтаксис    | <code>protocol imap   pop3   smtp;</code> |
| По умолчанию | —                                         |
| Контекст     | 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
```

При сборке из исходного кода поддержку ненужных протоколов можно отключить с помощью параметров сборки `--without-mail_imap_module`, `--without-mail_pop3_module` и `--without-mail_smtp_module`.

### resolver

|              |                                                                                                   |
|--------------|---------------------------------------------------------------------------------------------------|
| Синтаксис    | <code>resolver адрес ... [valid=время] [ipv4=on   off] [ipv6=on   off] [status_zone=зона];</code> |
| По умолчанию | <code>resolver off;</code>                                                                        |
| Контекст     | 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;
```

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.

По умолчанию Angie кэширует ответы, используя значение TTL из ответа. 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.

|                          |                                                                                                            |
|--------------------------|------------------------------------------------------------------------------------------------------------|
| <code>ipv4=off</code>    | запрещает поиск IPv4-адресов                                                                               |
| <code>ipv6=off</code>    | запрещает поиск IPv6-адресов                                                                               |
| <code>status_zone</code> | <i>необязательный</i> 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.

#### Подсказка

При запуске в Docker используйте соответствующий внутренний адрес DNS-сервера, например 127.0.0.11.

### resolver\_timeout

|              |                                              |
|--------------|----------------------------------------------|
| Синтаксис    | <code>resolver_timeout</code> <i>время</i> ; |
| По умолчанию | <code>resolver_timeout 30s</code> ;          |
| Контекст     | mail, server                                 |

Задаёт таймаут для преобразования имени в адрес, например:

```
resolver_timeout 5s;
```

### server

|              |                             |
|--------------|-----------------------------|
| Синтаксис    | <code>server { ... }</code> |
| По умолчанию | —                           |
| Контекст     | mail                        |

Задаёт конфигурацию для сервера.

## server\_name

|              |                                    |
|--------------|------------------------------------|
| Синтаксис    | <code>server_name имя;</code>      |
| По умолчанию | <code>server_name hostname;</code> |
| Контекст     | 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

|              |                             |
|--------------|-----------------------------|
| Синтаксис    | <code>timeout время;</code> |
| По умолчанию | <code>timeout 60s;</code>   |
| Контекст     | mail, server                |

Задаёт таймаут, который используется до начала проксирования на бэкенд.

### 3.3.5 Модуль Google PerfTools

Включает поддержку профилирования рабочих процессов Angie при помощи Google Performance Tools. Модуль предназначен для разработчиков Angie и позволяет им анализировать и оптимизировать производительность сервера, предоставляя подробную информацию об использовании памяти, загрузке процессора и других метриках производительности.

При сборке из исходного кода модуль не собирается по умолчанию; его необходимо включить с помощью параметра сборки `--with-google_perftools_module`.

#### Важно

Для этого модуля нужна библиотека `gperftools`.

#### Пример конфигурации

```
google_perftools_profiles /var/log/angie/perftools;
```

Профили будут сохраняться в файлах вида `/var/log/angie/perftools.<PID рабочего процесса>`.

## Директивы

### google\_perftools\_profiles

|              |                                                              |
|--------------|--------------------------------------------------------------|
| Синтаксис    | <code>google_perftools_profiles</code> <i>префикс файла;</i> |
| По умолчанию | —                                                            |
| Контекст     | <code>main</code>                                            |

Задаёт префикс имени файла, где будет храниться информация о профилировании рабочего процесса Angie. Идентификатор рабочего процесса добавляется в конце имени через точку, например: `/var/log/angie/perftools.1234`.

## 3.3.6 Модуль WASM

### WAMR

Модуль обеспечивает интеграцию с [WebAssembly Micro Runtime](#) для выполнения WASM-кода, добавляя ряд директив, специфичных для этой среды выполнения, в контекст `wasm_modules`.

В наших репозиториях модуль собран динамически и доступен отдельным пакетом `angie-module-wamr`.

### Пример конфигурации

```
wasm_modules {
    wamr_heap_size 16k;
    wamr_stack_size 16k;
    load fft_transform.wasm id=fft;
}
```

## Директивы

### wamr\_heap\_size

|              |                                            |
|--------------|--------------------------------------------|
| Синтаксис    | <code>wamr_heap_size</code> <i>размер;</i> |
| По умолчанию | <code>wamr_heap_size 8k;</code>            |
| Контекст     | <code>wasm_modules</code>                  |

Устанавливает размер кучи для отдельного экземпляра модуля.



### wamr\_global\_heap\_size

|              |                                            |
|--------------|--------------------------------------------|
| Синтаксис    | <code>wamr_global_heap_size размер;</code> |
| По умолчанию | <code>wamr_global_heap_size 1m;</code>     |
| Контекст     | <code>wasm_modules</code>                  |

Устанавливает размер кучи для всей среды выполнения WAMR.

### wamr\_stack\_size

|              |                                      |
|--------------|--------------------------------------|
| Синтаксис    | <code>wamr_stack_size размер;</code> |
| По умолчанию | <code>wamr_stack_size 8k;</code>     |
| Контекст     | <code>wasm_modules</code>            |

Устанавливает размер стека для отдельного экземпляра модуля.

## Wasmtime

Модуль обеспечивает интеграцию со средой выполнения [Wasmtime](#) для выполнения WASM-кода, добавляя ряд директив, специфичных для этой среды, в контекст `wasm_modules`.

В наших репозиториях модуль собран динамически и доступен отдельным пакетом `angie-module-wasmtime`.

### Пример конфигурации

```
wasm_modules {
    wasmtime_stack_size 8k;

    wasmtime_enable_wasi on;

    load fft_transform.wasm id=fft;
}
```

## Директивы

### wasmtime\_enable\_wasi

|              |                                             |
|--------------|---------------------------------------------|
| Синтаксис    | <code>wasmtime_enable_wasi on   off;</code> |
| По умолчанию | <code>wasmtime_enable_wasi on;</code>       |
| Контекст     | <code>wasm_modules</code>                   |

Включает или отключает использование [API WebAssembly System Interface](#), предоставляющих базовый POSIX-подобный функционал для WASM-модулей, запускаемых в Angie.

**i Примечание**

API, специфичные для Angie, можно разрешить явно с помощью директивы *load*.

### wasmtime\_stack\_size

|              |                                                 |
|--------------|-------------------------------------------------|
| Синтаксис    | <code>wasmtime_stack_size <i>размер</i>;</code> |
| По умолчанию | <code>wasmtime_stack_size 8k;</code>            |
| Контекст     | <code>wasm_modules</code>                       |

Устанавливает для значения `max_wasm_stack` заданный размер, тем самым ограничивая максимальный объем стека, доступного для выполнения WASM-кода.

Основной модуль, реализующий базовую функциональность WASM в Angie: он включает поддержку загрузки альтернативных сред выполнения и модулей WASM, а также настройку их функций и ограничений.

Другие модули в этом разделе расширяют данную функциональность, позволяя гибко настраивать и оптимизировать возможности WASM для различных сценариев и требований.

В наших репозиториях модуль собран динамически и доступен отдельным пакетом `angie-module-wasm`.

### Пример конфигурации

```
# Эти директивы загружают основную функциональность
load_module modules/nginx_wasm_module.so;
load_module modules/nginx_wasm_core_module.so;

load_module modules/nginx_wasmtime_module.so;

# Доступны здесь: https://git.angie.software/web-server/angie-wasm
load_module modules/nginx_http_wasm_host_module.so;
load_module modules/nginx_http_wasm_content_module.so;
load_module modules/nginx_http_wasm_vars_module.so;

events {
}

wasm_modules {
    #use wasmtime;

    load ngx_http_handler.wasm id=handler;
    load ngx_http_vars.wasm id=vars type=reactor;
}

http {
    wasm_var vars "ngx:wasi/var-utils#sum-entry" $rvar $arg_a $arg_b $arg_c $arg_d;
```

```
server {
    listen *:8080;

    location / {
        return 200 "sum('$arg_a', '$arg_b', '$arg_c', '$arg_d')=$rvar\n";
    }

    location /wasm {
        client_max_body_size 20M;
        wasm_content handler "ngx:wasi/http-handler-entry#handle-request";
    }
}
```

## Директивы

### load

|              |                                                                                                             |
|--------------|-------------------------------------------------------------------------------------------------------------|
| Синтаксис    | <code>load file id=идентификатор [fs=путь_хоста:путь_гостя]... [api=api]... [type=command   reactor]</code> |
| По умолчанию | —                                                                                                           |
| Контекст     | wasm_modules                                                                                                |

Загружает модуль из *файла* на диске и назначает ему уникальный *идентификатор* (обязательный параметр). Во время загрузки происходит проверка того, что модуль можно инстанцировать.

Директива поддерживает следующие параметры:

|             |                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>fs</b>   | Позволяет гостю получить доступ к каталогу на хосте. Параметр может быть указан несколько раз для разных каталогов.                                                                                                                                                                                                                                                                                                           |
| <b>api</b>  | Явно ограничивает список API, разрешенных для модуля, перечисляя их. Если модуль пытается использовать недоступные API (не указанные здесь), возвращается ошибка "API не найден".<br>По умолчанию модулю доступны все API.                                                                                                                                                                                                    |
| <b>type</b> | Управляет жизненным циклом загруженного модуля. <ul style="list-style-type: none"> <li>В режиме <b>command</b> машина выполняется один раз, и ее состояние уничтожается после выполнения.</li> <li>В режиме <b>reactor</b> машина фактически работает бесконечно, позволяя выполнять код многократно. Это требует внимательного управления памятью: если ресурсы не освобождаются, могут возникнуть утечки памяти.</li> </ul> |

## wasm\_modules

|              |                       |
|--------------|-----------------------|
| Синтаксис    | wasm_modules { ... }; |
| По умолчанию | —                     |
| Контекст     | main                  |

Директива верхнего уровня, которая предоставляет контекст файла конфигурации, в котором должны указываться директивы WASM. Она может содержать команды для загрузки модулей WASM и настройки параметров, специфичных для той или иной среды выполнения.

### 3.3.7 Основной модуль

|                        |                                                                     |
|------------------------|---------------------------------------------------------------------|
| <i>Основной модуль</i> | Управление служебными файлами, процессами и другими модулями Angie. |
|------------------------|---------------------------------------------------------------------|

### 3.3.8 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 SSL-сертификатов по протоколу ACME.                        |
| <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 <b>User-Agent</b> .                        |
| <i>Charset</i>           | Настройка и преобразование кодировки ответа.                                        |
| <i>DAV</i>               | Управление файлами на сервере по протоколу WebDAV.                                  |
| <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>GeoIP</i>             | Получение данных об IP-адресах на основе геолокации по базам MaxMind GeoIP.         |
| <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.                                             |
| <i>Image Filter</i>      | Преобразование изображений.                                                         |
| <i>Index</i>             | Настройка индексных файлов, обслуживающих запросы с косой чертой в конце (/).       |
| <i>JS</i>                | Обработчики для расширения функциональности путем задания дополнительной логики     |
| <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 predefined пар "ключ-значение".                 |
| <i>Memcached</i>         | Получение ответов от Memcached-сервера.                                             |
| <i>Mirror</i>            | Зеркалирование запросов на другие серверы.                                          |
| <i>MP4</i>               | Псевдо-стриминг файлов в формате MP4.                                               |
| <i>Perl</i>              | Обработчики для расширения функциональности путем задания дополнительной логики     |

Таблица 1 – продолжение с предыдущей с

|                              |                                                                                             |
|------------------------------|---------------------------------------------------------------------------------------------|
| <i>Prometheus</i>            | Метрики сервера в формате, совместимом с Prometheus, для мониторинга и сбора статистики.    |
| <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 <i>Referer</i> .                                               |
| <i>Rewrite</i>               | Модификация URI запроса, перенаправления, установка переменных и выбор конфигурации.        |
| <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 кэширования. |
| <i>Split Clients</i>         | Создание переменных для A/B-тестирования, канареечных релизов, шардинга и других сценариев. |
| <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>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 сессии.     |
| <i>uWSGI</i>                 | Проксирование запроса к uWSGI-серверу.                                                      |
| <i>XSLT</i>                  | Преобразование XML-документов с помощью языка XSLT.                                         |

### 3.3.9 Потокные модули

|                      |                                                                                                                                               |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Stream</i>        | Основная функциональность потокного сервера для балансировки протоколов TCP и UDP на уровне L4.                                               |
| <i>Access</i>        | Контроль доступа на основе IP-адресов и диапазонов CIDR.                                                                                      |
| <i>Geo</i>           | Преобразование IP-адресов в заданные значения переменных.                                                                                     |
| <i>GeoIP</i>         | Получение данных об IP-адресах на основе геолокации по базам MaxMind GeoIP.                                                                   |
| <i>JS</i>            | Обработчики для расширения функциональности путем задания дополнительной логики на <i>njs</i> , подмножестве языка JavaScript.                |
| <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 predefined пар "ключ-значение".                                                                           |
| <i>MQTT</i>          | Чтение идентификатора клиента и имени пользователя из соединения по протоколу MQTT до момента принятия решения о балансировке.                |
| <i>Preread</i>       | 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</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 <i>ClientHello</i> без терминации SSL/TLS и до момента принятия решения о балансировке.                    |
| <i>Upstream</i>      | Настройка групп проксируемых серверов для балансировки нагрузки.                                                                              |
| <i>Upstream</i>      | Настройка активных проверок работоспособности для групп проксируемых серверов.                                                                |

### 3.3.10 Почтовые модули

|                  |                                                                                                                         |
|------------------|-------------------------------------------------------------------------------------------------------------------------|
| <i>Mail</i>      | Основная функциональность почтового прокси-сервера.                                                                     |
| <i>Auth HTTP</i> | Аутентификация пользователей и выбор сервера для последующего проксирования с помощью HTTP-запросов к внешнему серверу. |
| <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.                                                                                |

### 3.3.11 Модуль Google PerfTools

*Google Perf*: Отвечает за интеграцию с библиотекой Google Performance Tools для профилирования и анализа производительности приложений.

### 3.3.12 Модули WASM

|                 |                                                                         |
|-----------------|-------------------------------------------------------------------------|
| <i>WASM</i>     | Основная функциональность WASM, позволяющая запускать WASM-код в Angie. |
| <i>WAMR</i>     | Интеграция с WebAssembly Micro Runtime.                                 |
| <i>Wasmtime</i> | Интеграция со средой выполнения Wasmtime.                               |

## ГЛАВА 4

---

### Права на интеллектуальную собственность

---

Документация на программный продукт Angie PRO является интеллектуальной собственностью ООО «Веб-Сервер», документация создана в результате изменения (переработки) документации на программный продукт Nginx.

---

## Алфавитный указатель

---

### A

absolute\_redirect (*http*), 316  
 accept\_mutex (*core*), 17  
 accept\_mutex\_delay (*core*), 17  
 access\_log (*http*), 134  
 access\_log (*stream*), 374  
 acme (*http*), 28  
 acme\_client (*http*), 29  
 acme\_client\_path (*http*), 30  
 acme\_dns\_port (*http*), 30  
 acme\_hook (*http*), 31  
 add\_after\_body (*http*), 33  
 add\_before\_body (*http*), 33  
 add\_header (*http*), 115  
 add\_trailer (*http*), 115  
 addition\_types (*http*), 33  
 aio (*http*), 316  
 aio\_write (*http*), 317  
 alias (*http*), 317  
 allow (*http*), 28  
 allow (*stream*), 360  
 ancient\_browser (*http*), 70  
 ancient\_browser\_value (*http*), 70  
 api (*http*), 34  
 api\_config\_files (*http*), 35  
 auth\_basic (*http*), 66  
 auth\_basic\_user\_file (*http*), 66  
 auth\_delay (*http*), 318  
 auth\_http, 439  
 auth\_http\_header, 439  
 auth\_http\_pass\_client\_cert, 439  
 auth\_http\_timeout, 439  
 auth\_request (*http*), 67  
 auth\_request\_set (*http*), 67  
 auto\_redirect (*http*), 318  
 autoindex (*http*), 68  
 autoindex\_exact\_size (*http*), 68  
 autoindex\_format (*http*), 68  
 autoindex\_localtime (*http*), 68

### B

bind\_conn (*http*), 261  
 break (*http*), 209

### C

charset (*http*), 71  
 charset\_map (*http*), 72  
 charset\_types (*http*), 72  
 chunked\_transfer\_encoding (*http*), 318  
 client\_body\_buffer\_size (*http*), 319  
 client\_body\_in\_file\_only (*http*), 319  
 client\_body\_in\_single\_buffer (*http*), 319  
 client\_body\_temp\_path (*http*), 319  
 client\_body\_timeout (*http*), 320  
 client\_header\_buffer\_size (*http*), 320  
 client\_header\_timeout (*http*), 320  
 client\_max\_body\_size (*http*), 321  
 connection\_pool\_size (*http*), 321  
 create\_full\_put\_path (*http*), 74

### D

daemon (*core*), 17  
 dav\_access (*http*), 74  
 dav\_methods (*http*), 74  
 debug\_connection (*core*), 18  
 debug\_points (*core*), 18  
 default\_type (*http*), 321  
 deny (*http*), 28  
 deny (*stream*), 360  
 directio (*http*), 321  
 directio\_alignment (*http*), 322  
 disable\_symlinks (*http*), 322

### E

empty\_gif (*http*), 76  
 env (*core*), 18  
 error\_log (*core*), 19  
 error\_page (*http*), 323  
 etag (*http*), 324  
 events (*core*), 20  
 expires (*http*), 115

### F

fastcgi\_bind (*http*), 76  
 fastcgi\_buffer\_size (*http*), 77  
 fastcgi\_buffering (*http*), 77  
 fastcgi\_buffers (*http*), 77



- fastcgi\_busy\_buffers\_size (*http*), 78
  - fastcgi\_cache (*http*), 78
  - fastcgi\_cache\_background\_update (*http*), 78
  - fastcgi\_cache\_bypass (*http*), 78
  - fastcgi\_cache\_key (*http*), 79
  - fastcgi\_cache\_lock (*http*), 79
  - fastcgi\_cache\_lock\_age (*http*), 79
  - fastcgi\_cache\_lock\_timeout (*http*), 79
  - fastcgi\_cache\_max\_range\_offset (*http*), 80
  - fastcgi\_cache\_methods (*http*), 80
  - fastcgi\_cache\_min\_uses (*http*), 80
  - fastcgi\_cache\_path (*http*), 80
  - fastcgi\_cache\_revalidate (*http*), 82
  - fastcgi\_cache\_use\_stale (*http*), 82
  - fastcgi\_cache\_valid (*http*), 83
  - fastcgi\_catch\_stderr (*http*), 83
  - fastcgi\_connect\_timeout (*http*), 84
  - fastcgi\_connection\_drop (*http*), 84
  - fastcgi\_force\_ranges (*http*), 84
  - fastcgi\_hide\_header (*http*), 85
  - fastcgi\_ignore\_client\_abort (*http*), 85
  - fastcgi\_ignore\_headers (*http*), 85
  - fastcgi\_index (*http*), 86
  - fastcgi\_intercept\_errors (*http*), 86
  - fastcgi\_keep\_conn (*http*), 86
  - fastcgi\_limit\_rate (*http*), 86
  - fastcgi\_max\_temp\_file\_size (*http*), 87
  - fastcgi\_next\_upstream (*http*), 87
  - fastcgi\_next\_upstream\_timeout (*http*), 88
  - fastcgi\_next\_upstream\_tries (*http*), 88
  - fastcgi\_no\_cache (*http*), 89
  - fastcgi\_param (*http*), 89
  - fastcgi\_pass (*http*), 90
  - fastcgi\_pass\_header (*http*), 90
  - fastcgi\_pass\_request\_body (*http*), 90
  - fastcgi\_pass\_request\_headers (*http*), 90
  - fastcgi\_read\_timeout (*http*), 91
  - fastcgi\_request\_buffering (*http*), 91
  - fastcgi\_send\_lowat (*http*), 91
  - fastcgi\_send\_timeout (*http*), 92
  - fastcgi\_socket\_keepalive (*http*), 92
  - fastcgi\_split\_path\_info (*http*), 92
  - fastcgi\_store (*http*), 92
  - fastcgi\_store\_access (*http*), 93
  - fastcgi\_temp\_file\_write\_size (*http*), 94
  - fastcgi\_temp\_path (*http*), 94
  - feedback (*http*), 262
  - feedback (*stream*), 416
  - flv (*http*), 96
- ## G
- geo (*http*), 96
  - geo (*stream*), 361
  - geoip\_city (*http*), 99
  - geoip\_city (*stream*), 363
  - geoip\_country (*http*), 98
  - geoip\_country (*stream*), 363
  - geoip\_org (*http*), 99
  - geoip\_org (*stream*), 363
  - geoip\_proxy (*http*), 99
  - geoip\_proxy\_recursive (*http*), 100
  - google\_perftools\_profiles, 462
  - grpc\_bind (*http*), 101
  - grpc\_buffer\_size (*http*), 101
  - grpc\_connect\_timeout (*http*), 101
  - grpc\_connection\_drop (*http*), 101
  - grpc\_hide\_header (*http*), 102
  - grpc\_ignore\_headers (*http*), 102
  - grpc\_intercept\_errors (*http*), 102
  - grpc\_next\_upstream (*http*), 103
  - grpc\_next\_upstream\_timeout (*http*), 104
  - grpc\_next\_upstream\_tries (*http*), 104
  - grpc\_pass (*http*), 104
  - grpc\_pass\_header (*http*), 105
  - grpc\_read\_timeout (*http*), 105
  - grpc\_send\_timeout (*http*), 105
  - grpc\_set\_header (*http*), 105
  - grpc\_socket\_keepalive (*http*), 106
  - grpc\_ssl\_certificate (*http*), 106
  - grpc\_ssl\_certificate\_key (*http*), 106
  - grpc\_ssl\_ciphers (*http*), 107
  - grpc\_ssl\_conf\_command (*http*), 107
  - grpc\_ssl\_crl (*http*), 107
  - grpc\_ssl\_name (*http*), 108
  - grpc\_ssl\_password\_file (*http*), 108
  - grpc\_ssl\_protocols (*http*), 108
  - grpc\_ssl\_server\_name (*http*), 108
  - grpc\_ssl\_session\_reuse (*http*), 109
  - grpc\_ssl\_trusted\_certificate (*http*), 109
  - grpc\_ssl\_verify (*http*), 109
  - grpc\_ssl\_verify\_depth (*http*), 109
  - gunzip (*http*), 110
  - gunzip\_buffers (*http*), 110
  - gzip (*http*), 111
  - gzip\_buffers (*http*), 111
  - gzip\_comp\_level (*http*), 111
  - gzip\_disable (*http*), 112
  - gzip\_http\_version (*http*), 112
  - gzip\_min\_length (*http*), 112
  - gzip\_proxied (*http*), 112
  - gzip\_static (*http*), 114
  - gzip\_types (*http*), 113
  - gzip\_vary (*http*), 113
- ## H
- hash (*http*), 263
  - hash (*stream*), 417
  - http (*http*), 324
  - http2 (*http*), 307
  - http2\_body\_preload\_size (*http*), 307
  - http2\_chunk\_size (*http*), 307
  - http2\_max\_concurrent\_pushes (*http*), 307
  - http2\_max\_concurrent\_streams (*http*), 307
  - http2\_push (*http*), 308
  - http2\_push\_preload (*http*), 308
  - http2\_recv\_buffer\_size (*http*), 308

http3 (*http*), 310  
 http3\_hq (*http*), 310  
 http3\_max\_concurrent\_streams (*http*), 310  
 http3\_max\_table\_capacity (*http*), 310  
 http3\_stream\_buffer\_size (*http*), 311

## I

if (*http*), 210  
 if\_modified\_since (*http*), 325  
 ignore\_invalid\_headers (*http*), 325  
 image\_filter (*http*), 117  
 image\_filter\_buffer (*http*), 118  
 image\_filter\_interlace (*http*), 118  
 image\_filter\_jpeg\_quality (*http*), 118  
 image\_filter\_sharpen (*http*), 119  
 image\_filter\_transparency (*http*), 119  
 image\_filter\_webp\_quality (*http*), 119  
 imap\_auth, 442  
 imap\_capabilities, 442  
 imap\_client\_buffer, 443  
 include (*core*), 20  
 index (*http*), 120  
 internal (*http*), 325  
 ip\_hash (*http*), 264

## J

js\_access (*stream*), 366  
 js\_body\_filter (*http*), 122  
 js\_content (*http*), 123  
 js\_fetch\_buffer\_size (*http*), 123  
 js\_fetch\_buffer\_size (*stream*), 366  
 js\_fetch\_ciphers (*http*), 123  
 js\_fetch\_ciphers (*stream*), 366  
 js\_fetch\_max\_response\_buffer\_size (*http*), 124  
 js\_fetch\_max\_response\_buffer\_size (*stream*), 366  
 js\_fetch\_protocols (*http*), 124  
 js\_fetch\_protocols (*stream*), 367  
 js\_fetch\_timeout (*http*), 124  
 js\_fetch\_timeout (*stream*), 367  
 js\_fetch\_trusted\_certificate (*http*), 124  
 js\_fetch\_trusted\_certificate (*stream*), 367  
 js\_fetch\_verify (*http*), 125  
 js\_fetch\_verify (*stream*), 367  
 js\_fetch\_verify\_depth (*http*), 125  
 js\_fetch\_verify\_depth (*stream*), 368  
 js\_filter (*stream*), 368  
 js\_header\_filter (*http*), 125  
 js\_import (*http*), 125  
 js\_import (*stream*), 368  
 js\_path (*http*), 126  
 js\_path (*stream*), 369  
 js\_preload\_object (*http*), 126  
 js\_preload\_object (*stream*), 369  
 js\_preread (*stream*), 369  
 js\_set (*http*), 126  
 js\_set (*stream*), 370  
 js\_shared\_dict\_zone (*http*), 127

js\_shared\_dict\_zone (*stream*), 370  
 js\_var (*http*), 128  
 js\_var (*stream*), 371

## K

keepalive (*http*), 264  
 keepalive\_disable (*http*), 326  
 keepalive\_requests (*http*), 266, 326  
 keepalive\_time (*http*), 266, 327  
 keepalive\_timeout (*http*), 267, 327

## L

large\_client\_header\_buffers (*http*), 327  
 least\_conn (*http*), 267  
 least\_conn (*stream*), 417  
 least\_time (*http*), 267  
 least\_time (*stream*), 418  
 limit\_conn (*http*), 129  
 limit\_conn (*stream*), 372  
 limit\_conn\_dry\_run (*http*), 130  
 limit\_conn\_dry\_run (*stream*), 373  
 limit\_conn\_log\_level (*http*), 130  
 limit\_conn\_log\_level (*stream*), 373  
 limit\_conn\_status (*http*), 130  
 limit\_conn\_zone (*http*), 130  
 limit\_conn\_zone (*stream*), 373  
 limit\_except (*http*), 327  
 limit\_rate (*http*), 328  
 limit\_rate\_after (*http*), 329  
 limit\_req (*http*), 131  
 limit\_req\_dry\_run (*http*), 132  
 limit\_req\_log\_level (*http*), 132  
 limit\_req\_status (*http*), 133  
 limit\_req\_zone (*http*), 133  
 lingering\_close (*http*), 329  
 lingering\_time (*http*), 329  
 lingering\_timeout (*http*), 330  
 listen, 456  
 listen (*http*), 330  
 listen (*stream*), 427  
 load, 465  
 load\_module (*core*), 20  
 location (*http*), 333  
 lock\_file (*core*), 21  
 log\_format (*http*), 135  
 log\_format (*stream*), 375  
 log\_not\_found (*http*), 335  
 log\_subrequest (*http*), 335

## M

mail, 458  
 map (*http*), 137  
 map (*stream*), 376  
 map\_hash\_bucket\_size (*http*), 138  
 map\_hash\_bucket\_size (*stream*), 378  
 map\_hash\_max\_size (*http*), 139  
 map\_hash\_max\_size (*stream*), 378  
 master\_process (*core*), 21

max\_commands, 458  
max\_errors, 458  
max\_headers (*http*), 336  
max\_ranges (*http*), 336  
memcached\_bind (*http*), 139  
memcached\_buffer\_size (*http*), 140  
memcached\_connect\_timeout (*http*), 140  
memcached\_gzip\_flag (*http*), 140  
memcached\_next\_upstream (*http*), 140  
memcached\_next\_upstream\_timeout (*http*), 141  
memcached\_next\_upstream\_tries (*http*), 141  
memcached\_pass (*http*), 142  
memcached\_read\_timeout (*http*), 142  
memcached\_send\_timeout (*http*), 142  
memcached\_socket\_keepalive (*http*), 143  
merge\_slashes (*http*), 336  
min\_delete\_depth (*http*), 75  
mirror (*http*), 144  
mirror\_request\_body (*http*), 144  
modern\_browser (*http*), 70  
modern\_browser\_value (*http*), 71  
mp4 (*http*), 146  
mp4\_buffer\_size (*http*), 146  
mp4\_limit\_rate (*http*), 146  
mp4\_limit\_rate\_after (*http*), 147  
mp4\_max\_buffer\_size (*http*), 146  
mp4\_start\_key\_frame (*http*), 147  
mqtt\_preread (*stream*), 379  
msie\_padding (*http*), 337  
msie\_refresh (*http*), 337  
multi\_accept (*core*), 21

## O

open\_file\_cache (*http*), 337  
open\_file\_cache\_errors (*http*), 338  
open\_file\_cache\_min\_uses (*http*), 338  
open\_file\_cache\_valid (*http*), 338  
open\_log\_file\_cache (*http*), 136  
open\_log\_file\_cache (*stream*), 375  
output\_buffers (*http*), 338  
override\_charset (*http*), 73

## P

pass (*stream*), 380  
pcre\_jit (*core*), 22  
perl (*http*), 149  
perl\_modules (*http*), 149  
perl\_require (*http*), 149  
perl\_set (*http*), 150  
pid (*core*), 22  
pop3\_auth, 443  
pop3\_capabilities, 443  
port\_in\_redirect (*http*), 339  
postpone\_output (*http*), 339  
preread\_buffer\_size (*stream*), 430  
preread\_timeout (*stream*), 430  
prometheus (*http*), 171  
prometheus\_template (*http*), 171

protocol, 459  
proxy\_bind (*http*), 174  
proxy\_bind (*stream*), 381  
proxy\_buffer, 444  
proxy\_buffer\_size (*http*), 174  
proxy\_buffer\_size (*stream*), 381  
proxy\_buffering (*http*), 175  
proxy\_buffers (*http*), 175  
proxy\_busy\_buffers\_size (*http*), 175  
proxy\_cache (*http*), 176  
proxy\_cache\_background\_update (*http*), 177  
proxy\_cache\_bypass (*http*), 177  
proxy\_cache\_convert\_head (*http*), 177  
proxy\_cache\_key (*http*), 177  
proxy\_cache\_lock (*http*), 178  
proxy\_cache\_lock\_age (*http*), 178  
proxy\_cache\_lock\_timeout (*http*), 178  
proxy\_cache\_max\_range\_offset (*http*), 178  
proxy\_cache\_methods (*http*), 179  
proxy\_cache\_min\_uses (*http*), 179  
proxy\_cache\_path (*http*), 179  
proxy\_cache\_revalidate (*http*), 181  
proxy\_cache\_use\_stale (*http*), 181  
proxy\_cache\_valid (*http*), 181  
proxy\_connect\_timeout (*http*), 182  
proxy\_connect\_timeout (*stream*), 381  
proxy\_connection\_drop (*http*), 183  
proxy\_connection\_drop (*stream*), 382  
proxy\_cookie\_domain (*http*), 183  
proxy\_cookie\_flags (*http*), 184  
proxy\_cookie\_path (*http*), 184  
proxy\_download\_rate (*stream*), 382  
proxy\_force\_ranges (*http*), 185  
proxy\_half\_close (*stream*), 383  
proxy\_headers\_hash\_bucket\_size (*http*), 185  
proxy\_headers\_hash\_max\_size (*http*), 185  
proxy\_hide\_header (*http*), 186  
proxy\_http3\_hq (*http*), 186  
proxy\_http3\_max\_concurrent\_streams (*http*),  
186  
proxy\_http3\_max\_table\_capacity (*http*), 187  
proxy\_http3\_stream\_buffer\_size (*http*), 187  
proxy\_http\_version (*http*), 186  
proxy\_ignore\_client\_abort (*http*), 187  
proxy\_ignore\_headers (*http*), 187  
proxy\_intercept\_errors (*http*), 188  
proxy\_limit\_rate (*http*), 188  
proxy\_max\_temp\_file\_size (*http*), 188  
proxy\_method (*http*), 189  
proxy\_next\_upstream (*http*), 189  
proxy\_next\_upstream (*stream*), 383  
proxy\_next\_upstream\_timeout (*http*), 190  
proxy\_next\_upstream\_timeout (*stream*), 383  
proxy\_next\_upstream\_tries (*http*), 190  
proxy\_next\_upstream\_tries (*stream*), 383  
proxy\_no\_cache (*http*), 191  
proxy\_pass (*http*), 191  
proxy\_pass (*stream*), 384

proxy\_pass\_error\_message, 444  
 proxy\_pass\_header (*http*), 192  
 proxy\_pass\_request\_body (*http*), 192  
 proxy\_pass\_request\_headers (*http*), 193  
 proxy\_pass\_trailers (*http*), 193  
 proxy\_protocol, 444  
 proxy\_protocol (*stream*), 384  
 proxy\_protocol\_timeout (*stream*), 430  
 proxy\_quic\_active\_connection\_id\_limit (*http*), 194  
 proxy\_quic\_gso (*http*), 194  
 proxy\_quic\_host\_key (*http*), 194  
 proxy\_read\_timeout (*http*), 194  
 proxy\_redirect (*http*), 194  
 proxy\_request\_buffering (*http*), 196  
 proxy\_requests (*stream*), 384  
 proxy\_responses (*stream*), 384  
 proxy\_send\_lowat (*http*), 196  
 proxy\_send\_timeout (*http*), 197  
 proxy\_set\_body (*http*), 197  
 proxy\_set\_header (*http*), 197  
 proxy\_smtp\_auth, 444  
 proxy\_socket\_keepalive (*http*), 198  
 proxy\_socket\_keepalive (*stream*), 385  
 proxy\_ssl (*stream*), 385  
 proxy\_ssl\_certificate (*http*), 198  
 proxy\_ssl\_certificate (*stream*), 385  
 proxy\_ssl\_certificate\_key (*http*), 198  
 proxy\_ssl\_certificate\_key (*stream*), 386  
 proxy\_ssl\_ciphers (*http*), 199  
 proxy\_ssl\_ciphers (*stream*), 386  
 proxy\_ssl\_conf\_command (*http*), 199  
 proxy\_ssl\_conf\_command (*stream*), 386  
 proxy\_ssl\_crl (*http*), 200  
 proxy\_ssl\_crl (*stream*), 387  
 proxy\_ssl\_name (*http*), 200  
 proxy\_ssl\_name (*stream*), 387  
 proxy\_ssl\_ntls (*http*), 200  
 proxy\_ssl\_ntls (*stream*), 387  
 proxy\_ssl\_password\_file (*http*), 201  
 proxy\_ssl\_password\_file (*stream*), 388  
 proxy\_ssl\_protocols (*http*), 201  
 proxy\_ssl\_protocols (*stream*), 388  
 proxy\_ssl\_server\_name (*http*), 201  
 proxy\_ssl\_server\_name (*stream*), 388  
 proxy\_ssl\_session\_reuse (*http*), 202  
 proxy\_ssl\_session\_reuse (*stream*), 389  
 proxy\_ssl\_trusted\_certificate (*http*), 202  
 proxy\_ssl\_trusted\_certificate (*stream*), 389  
 proxy\_ssl\_verify (*http*), 202  
 proxy\_ssl\_verify (*stream*), 389  
 proxy\_ssl\_verify\_depth (*http*), 202  
 proxy\_ssl\_verify\_depth (*stream*), 389  
 proxy\_store (*http*), 203  
 proxy\_store\_access (*http*), 204  
 proxy\_temp\_file\_write\_size (*http*), 204  
 proxy\_temp\_path (*http*), 204  
 proxy\_timeout, 445

proxy\_timeout (*stream*), 390  
 proxy\_upload\_rate (*stream*), 390

## Q

queue (*http*), 268  
 quic\_active\_connection\_id\_limit (*http*), 311  
 quic\_bpf (*http*), 311  
 quic\_gso (*http*), 311  
 quic\_host\_key (*http*), 312  
 quic\_retry (*http*), 312

## R

random (*http*), 268  
 random (*stream*), 418  
 random\_index (*http*), 206  
 rdp\_preread (*stream*), 392  
 read\_ahead (*http*), 339  
 real\_ip\_header (*http*), 206  
 real\_ip\_recursive (*http*), 207  
 recursive\_error\_pages (*http*), 339  
 referer\_hash\_bucket\_size (*http*), 208  
 referer\_hash\_max\_size (*http*), 208  
 request\_pool\_size (*http*), 340  
 reset\_timedout\_connection (*http*), 340  
 resolver, 459  
 resolver (*http*), 340  
 resolver (*stream*), 430  
 resolver\_timeout, 460  
 resolver\_timeout (*http*), 341  
 resolver\_timeout (*stream*), 431  
 response\_time\_factor (*http*), 269  
 response\_time\_factor (*stream*), 419  
 return (*http*), 211  
 return (*stream*), 393  
 rewrite (*http*), 211  
 rewrite\_log (*http*), 212  
 root (*http*), 341

## S

satisfy (*http*), 342  
 scgi\_bind (*http*), 214  
 scgi\_buffer\_size (*http*), 214  
 scgi\_buffering (*http*), 215  
 scgi\_buffers (*http*), 215  
 scgi\_busy\_buffers\_size (*http*), 215  
 scgi\_cache (*http*), 216  
 scgi\_cache\_background\_update (*http*), 216  
 scgi\_cache\_bypass (*http*), 216  
 scgi\_cache\_key (*http*), 217  
 scgi\_cache\_lock (*http*), 217  
 scgi\_cache\_lock\_age (*http*), 217  
 scgi\_cache\_lock\_timeout (*http*), 217  
 scgi\_cache\_max\_range\_offset (*http*), 218  
 scgi\_cache\_methods (*http*), 218  
 scgi\_cache\_min\_uses (*http*), 218  
 scgi\_cache\_path (*http*), 218  
 scgi\_cache\_revalidate (*http*), 220  
 scgi\_cache\_use\_stale (*http*), 220

scgi\_cache\_valid (*http*), 221  
 scgi\_connect\_timeout (*http*), 222  
 scgi\_connection\_drop (*http*), 222  
 scgi\_force\_ranges (*http*), 222  
 scgi\_hide\_header (*http*), 222  
 scgi\_ignore\_client\_abort (*http*), 223  
 scgi\_ignore\_headers (*http*), 223  
 scgi\_intercept\_errors (*http*), 223  
 scgi\_limit\_rate (*http*), 224  
 scgi\_max\_temp\_file\_size (*http*), 224  
 scgi\_next\_upstream (*http*), 224  
 scgi\_next\_upstream\_timeout (*http*), 225  
 scgi\_next\_upstream\_tries (*http*), 226  
 scgi\_no\_cache (*http*), 226  
 scgi\_param (*http*), 226  
 scgi\_pass (*http*), 227  
 scgi\_pass\_header (*http*), 227  
 scgi\_pass\_request\_body (*http*), 227  
 scgi\_pass\_request\_headers (*http*), 228  
 scgi\_read\_timeout (*http*), 228  
 scgi\_request\_buffering (*http*), 228  
 scgi\_send\_timeout (*http*), 228  
 scgi\_socket\_keepalive (*http*), 229  
 scgi\_store (*http*), 229  
 scgi\_store\_access (*http*), 230  
 scgi\_temp\_file\_write\_size (*http*), 230  
 scgi\_temp\_path (*http*), 230  
 secure\_link (*http*), 231  
 secure\_link\_md5 (*http*), 232  
 secure\_link\_secret (*http*), 232  
 send\_lowat (*http*), 342  
 send\_timeout (*http*), 342  
 sendfile (*http*), 343  
 sendfile\_max\_chunk (*http*), 343  
 server, 460  
 server (*http*), 269, 343  
 server (*stream*), 413, 431  
 server\_name, 460  
 server\_name (*http*), 344  
 server\_name (*stream*), 432  
 server\_name\_in\_redirect (*http*), 346  
 server\_names\_hash\_bucket\_size (*http*), 346  
 server\_names\_hash\_bucket\_size (*stream*), 433  
 server\_names\_hash\_max\_size (*http*), 346  
 server\_names\_hash\_max\_size (*stream*), 433  
 server\_tokens (*http*), 346  
 set (*http*), 212  
 set (*stream*), 394  
 set\_real\_ip\_from, 446  
 set\_real\_ip\_from (*http*), 206  
 set\_real\_ip\_from (*stream*), 393  
 slice (*http*), 234  
 smtp\_auth, 446  
 smtp\_capabilities, 446  
 smtp\_client\_buffer, 447  
 smtp\_greeting\_delay, 447  
 source\_charset (*http*), 73  
 split\_clients (*http*), 235  
 split\_clients (*stream*), 395  
 ssi (*http*), 236  
 ssi\_last\_modified (*http*), 236  
 ssi\_min\_file\_chunk (*http*), 236  
 ssi\_silent\_errors (*http*), 236  
 ssi\_types (*http*), 237  
 ssi\_value\_length (*http*), 237  
 ssl\_alpn (*stream*), 396  
 ssl\_buffer\_size (*http*), 242  
 ssl\_certificate, 448  
 ssl\_certificate (*http*), 242  
 ssl\_certificate (*stream*), 396  
 ssl\_certificate\_key, 449  
 ssl\_certificate\_key (*http*), 243  
 ssl\_certificate\_key (*stream*), 397  
 ssl\_ciphers, 449  
 ssl\_ciphers (*http*), 244  
 ssl\_ciphers (*stream*), 398  
 ssl\_client\_certificate, 449  
 ssl\_client\_certificate (*http*), 244  
 ssl\_client\_certificate (*stream*), 398  
 ssl\_conf\_command, 450  
 ssl\_conf\_command (*http*), 245  
 ssl\_conf\_command (*stream*), 398  
 ssl\_crl, 450  
 ssl\_crl (*http*), 245  
 ssl\_crl (*stream*), 399  
 ssl\_dhparam, 450  
 ssl\_dhparam (*http*), 245  
 ssl\_dhparam (*stream*), 399  
 ssl\_early\_data (*http*), 246  
 ssl\_ecdh\_curve, 451  
 ssl\_ecdh\_curve (*http*), 246  
 ssl\_ecdh\_curve (*stream*), 399  
 ssl\_engine (*core*), 22  
 ssl\_handshake\_timeout (*stream*), 400  
 ssl\_ntls (*http*), 247  
 ssl\_ntls (*stream*), 401  
 ssl\_ocsp (*http*), 247  
 ssl\_ocsp (*stream*), 400  
 ssl\_ocsp\_cache (*http*), 248  
 ssl\_ocsp\_cache (*stream*), 401  
 ssl\_ocsp\_responder (*http*), 248  
 ssl\_ocsp\_responder (*stream*), 401  
 ssl\_password\_file, 451  
 ssl\_password\_file (*http*), 248  
 ssl\_password\_file (*stream*), 402  
 ssl\_prefer\_server\_ciphers, 452  
 ssl\_prefer\_server\_ciphers (*http*), 249  
 ssl\_prefer\_server\_ciphers (*stream*), 402  
 ssl\_preread (*stream*), 411  
 ssl\_protocols, 452  
 ssl\_protocols (*http*), 249  
 ssl\_protocols (*stream*), 402  
 ssl\_reject\_handshake (*http*), 249  
 ssl\_session\_cache, 452  
 ssl\_session\_cache (*http*), 250  
 ssl\_session\_cache (*stream*), 403

ssl\_session\_ticket\_key, 453  
 ssl\_session\_ticket\_key (*http*), 250  
 ssl\_session\_ticket\_key (*stream*), 403  
 ssl\_session\_tickets, 454  
 ssl\_session\_tickets (*http*), 251  
 ssl\_session\_tickets (*stream*), 404  
 ssl\_session\_timeout, 454  
 ssl\_session\_timeout (*http*), 251  
 ssl\_session\_timeout (*stream*), 404  
 ssl\_stapling (*http*), 251  
 ssl\_stapling (*stream*), 404  
 ssl\_stapling\_file (*http*), 252  
 ssl\_stapling\_file (*stream*), 405  
 ssl\_stapling\_responder (*http*), 252  
 ssl\_stapling\_responder (*stream*), 405  
 ssl\_stapling\_verify (*http*), 252  
 ssl\_stapling\_verify (*stream*), 405  
 ssl\_trusted\_certificate, 454  
 ssl\_trusted\_certificate (*http*), 253  
 ssl\_trusted\_certificate (*stream*), 406  
 ssl\_verify\_client, 454  
 ssl\_verify\_client (*http*), 253  
 ssl\_verify\_client (*stream*), 406  
 ssl\_verify\_depth, 455  
 ssl\_verify\_depth (*http*), 253  
 ssl\_verify\_depth (*stream*), 406  
 starttls, 455  
 state (*http*), 272  
 state (*stream*) ((*stream upstream module*)), 415  
 status\_zone (*http*), 347  
 status\_zone (*stream*), 433  
 sticky (*http*), 272  
 sticky (*stream*), 419  
 sticky\_secret (*http*), 276  
 sticky\_secret (*stream*), 421  
 sticky\_strict (*http*), 276  
 sticky\_strict (*stream*), 421  
 stream (*stream*), 434  
 stub\_status (*http*), 258  
 sub\_filter (*http*), 259  
 sub\_filter\_last\_modified (*http*), 260  
 sub\_filter\_once (*http*), 260  
 sub\_filter\_types (*http*), 260  
 subrequest\_output\_buffer\_size (*http*), 348

## T

tcp\_nodelay (*http*), 348  
 tcp\_nodelay (*stream*), 435  
 tcp\_nopush (*http*), 348  
 thread\_pool (*core*), 23  
 timeout, 461  
 timer\_resolution (*core*), 23  
 try\_files (*http*), 349  
 types (*http*), 351  
 types\_hash\_bucket\_size (*http*), 351  
 types\_hash\_max\_size (*http*), 351

## U

underscores\_in\_headers (*http*), 352  
 uninitialized\_variable\_warn (*http*), 212  
 upstream (*http*), 276  
 upstream (*stream*), 412  
 upstream\_probe (*http*), 281  
 upstream\_probe (*stream*), 424  
 upstream\_probe\_timeout (*stream*), 390  
 use (*core*), 23  
 user (*core*), 24  
 userid (*http*), 283  
 userid\_domain (*http*), 283  
 userid\_expires (*http*), 284  
 userid\_flags (*http*), 284  
 userid\_mark (*http*), 284  
 userid\_name (*http*), 284  
 userid\_p3p (*http*), 285  
 userid\_path (*http*), 285  
 userid\_service (*http*), 285  
 uwsgi\_bind (*http*), 286  
 uwsgi\_buffer\_size (*http*), 287  
 uwsgi\_buffering (*http*), 287  
 uwsgi\_buffers (*http*), 287  
 uwsgi\_busy\_buffers\_size (*http*), 288  
 uwsgi\_cache (*http*), 288  
 uwsgi\_cache\_background\_update (*http*), 288  
 uwsgi\_cache\_bypass (*http*), 288  
 uwsgi\_cache\_key (*http*), 289  
 uwsgi\_cache\_lock (*http*), 289  
 uwsgi\_cache\_lock\_age (*http*), 289  
 uwsgi\_cache\_lock\_timeout (*http*), 289  
 uwsgi\_cache\_max\_range\_offset (*http*), 290  
 uwsgi\_cache\_methods (*http*), 290  
 uwsgi\_cache\_min\_uses (*http*), 290  
 uwsgi\_cache\_path (*http*), 290  
 uwsgi\_cache\_revalidate (*http*), 292  
 uwsgi\_cache\_use\_stale (*http*), 292  
 uwsgi\_cache\_valid (*http*), 293  
 uwsgi\_connect\_timeout (*http*), 294  
 uwsgi\_connection\_drop (*http*), 294  
 uwsgi\_force\_ranges (*http*), 294  
 uwsgi\_hide\_header (*http*), 294  
 uwsgi\_ignore\_client\_abort (*http*), 295  
 uwsgi\_ignore\_headers (*http*), 295  
 uwsgi\_intercept\_errors (*http*), 295  
 uwsgi\_limit\_rate (*http*), 296  
 uwsgi\_max\_temp\_file\_size (*http*), 296  
 uwsgi\_modifier1 (*http*), 296  
 uwsgi\_modifier2 (*http*), 297  
 uwsgi\_next\_upstream (*http*), 297  
 uwsgi\_next\_upstream\_timeout (*http*), 298  
 uwsgi\_next\_upstream\_tries (*http*), 298  
 uwsgi\_no\_cache (*http*), 298  
 uwsgi\_pass (*http*), 299  
 uwsgi\_pass\_header (*http*), 299  
 uwsgi\_pass\_request\_body (*http*), 299  
 uwsgi\_pass\_request\_headers (*http*), 299  
 uwsgi\_read\_timeout (*http*), 300

uwsgi\_request\_buffering (*http*), 300  
uwsgi\_send\_timeout (*http*), 300  
uwsgi\_socket\_keepalive (*http*), 300  
uwsgi\_ssl\_certificate (*http*), 301  
uwsgi\_ssl\_certificate\_key (*http*), 301  
uwsgi\_ssl\_ciphers (*http*), 301  
uwsgi\_ssl\_conf\_command (*http*), 301  
uwsgi\_ssl\_crl (*http*), 302  
uwsgi\_ssl\_name (*http*), 302  
uwsgi\_ssl\_password\_file (*http*), 302  
uwsgi\_ssl\_protocols (*http*), 303  
uwsgi\_ssl\_server\_name (*http*), 303  
uwsgi\_ssl\_session\_reuse (*http*), 303  
uwsgi\_ssl\_trusted\_certificate (*http*), 303  
uwsgi\_ssl\_verify (*http*), 304  
uwsgi\_ssl\_verify\_depth (*http*), 304  
uwsgi\_store (*http*), 304  
uwsgi\_store\_access (*http*), 305  
uwsgi\_temp\_file\_write\_size (*http*), 305  
uwsgi\_temp\_path (*http*), 305

## V

valid\_referers (*http*), 208  
variables\_hash\_bucket\_size (*http*), 352  
variables\_hash\_bucket\_size (*stream*), 435  
variables\_hash\_max\_size (*http*), 352  
variables\_hash\_max\_size (*stream*), 435

## W

wamr\_global\_heap\_size, 462  
wamr\_heap\_size, 462  
wamr\_stack\_size, 463  
wasm\_modules, 465  
wasmtime\_enable\_wasi, 463  
wasmtime\_stack\_size, 464  
worker\_aio\_requests (*core*), 24  
worker\_connections (*core*), 24  
worker\_cpu\_affinity (*core*), 24  
worker\_priority (*core*), 25  
worker\_processes (*core*), 26  
worker\_rlimit\_core (*core*), 26  
worker\_rlimit\_nofile (*core*), 26  
worker\_shutdown\_timeout (*core*), 26  
working\_directory (*core*), 27

## X

xclient, 445  
xml\_entities (*http*), 313  
xslt\_last\_modified (*http*), 314  
xslt\_param (*http*), 314  
xslt\_string\_param (*http*), 314  
xslt\_stylesheet (*http*), 314  
xslt\_types (*http*), 315

## Z

zone (*http*), 277  
zone (*stream*), 415