



Руководство по эксплуатации

версия 0.7.3

ООО «Веб-Сервер»

дек. 29, 2025

Оглавление

1	Аннотация	1
1.1	Общие сведения	1
1.2	Системные требования	2
2	Настройка	3
2.1	Аргументы командной строки	3
2.1.1	-angie-configmaps <строка>	4
2.1.2	-angie-dashboard	4
2.1.3	-angie-dashboard-port <int>	4
2.1.4	-angie-dashboard-allow-cidrs <строка>	4
2.1.5	-angie-debug	4
2.1.6	-angie-reload-timeout <значение>	4
2.1.7	-angie-status	4
2.1.8	-angie-status-allow-cidrs <строка>	4
2.1.9	-angie-status-port <int>	5
2.1.10	-angie-status-prometheus <bool>	5
2.1.11	-angie-status-prometheus-allow-cidrs	5
2.1.12	-angie-status-prometheus-path <строка>	5
2.1.13	-angie-status-prometheus-port <int>	5
2.1.14	-default-server-tls-secret <строка>	5
2.1.15	-disable-ipv6	5
2.1.16	-enable-cert-manager	6
2.1.17	-enable-custom-resources	6
2.1.18	-enable-external-dns	6
2.1.19	-enable-jwt	6
2.1.20	-enable-leader-election	6
2.1.21	-enable-oide	6
2.1.22	-enable-prometheus-metrics	6
2.1.23	-enable-service-insight	6
2.1.24	-enable-snippets	7
2.1.25	-enable-tls-passthrough	7
2.1.26	-external-service <строка>	7
2.1.27	-global-configuration <строка>	7
2.1.28	-health-status	7
2.1.29	-health-status-uri <строка>	7
2.1.30	-ingress-class <строка>	7
2.1.31	-ingresslink <строка>	8
2.1.32	-ingress-template-path <строка>	8
2.1.33	-leader-election-lock-name <строка>	8
2.1.34	-main-template-path <строка>	8
2.1.35	-prometheus-metrics-listen-port <int>	8

2.1.36	-prometheus-tls-secret <строка>	8
2.1.37	-proxy <строка>	8
2.1.38	-ready-status	9
2.1.39	-ready-status-port	9
2.1.40	-report-ingress-status	9
2.1.41	-service-insight-listen-port <int>	9
2.1.42	-service-insight-tls-secret <строка>	9
2.1.43	-tls-passthrough-port <int>	9
2.1.44	-transportserver-template-path <строка>	9
2.1.45	-v <значение>	10
2.1.46	-version	10
2.1.47	-virtualserver-template-path <строка>	10
2.1.48	-vmodule <значение>	10
2.1.49	-watch-namespace <строка>	10
2.1.50	-watch-namespace-label <строка>	10
2.1.51	-watch-secret-namespace <строка>	10
2.1.52	-wildcard-tls-secret <строка>	10
2.2	Настройка ANIC	11
2.2.1	Параметры Ingress Controller	11
2.2.2	Общие параметры	11
2.2.3	Параметры ведения журнала	13
2.2.4	Управление URI и заголовками в запросах	13
2.2.5	Авторизация и SSL/TLS	13
2.2.6	Протоколы	14
2.2.7	Апстримы	14
2.2.8	Настраиваемые шаблоны	14
2.3	ConfigMap	14
2.3.1	Использование ConfigMap	15
2.3.2	ConfigMap и аннотации Ingress	15
2.3.3	Переопределение ConfigMap для конкретного ресурса Ingress с помощью аннотации	15
2.3.4	ConfigMap и ресурсы VirtualServer, VirtualServerRoute	16
2.3.5	Краткое описание ключей ConfigMap	16
2.4	GlobalConfiguration	23
2.4.1	Предварительные требования	23
2.4.2	Спецификация GlobalConfiguration	23
2.4.3	Использование GlobalConfiguration	24
2.5	Policy	26
2.5.1	Предварительные требования	26
2.5.2	Спецификация Policy	26
2.6	TransportServer	41
2.6.1	Предварительные требования	41
2.6.2	Спецификация TransportServer	41
2.6.3	Использование TransportServer	46
2.7	VirtualServer, VirtualServerRoute	49
2.7.1	Спецификация VirtualServer	49
2.7.2	Спецификация VirtualServerRoute	58
2.7.3	Общие части VirtualServer и VirtualServerRoute	61
2.7.4	Использование VirtualServer и VirtualServerRoute	75
2.7.5	Настройка с помощью ConfigMap	78
2.8	Расширенная конфигурация с помощью аннотаций	78
2.8.1	Использование аннотаций	79
2.8.2	Валидация	79
2.8.3	Сводка аннотаций	80
3	Журналы и мониторинг	89
3.1	Просмотр журналов	89
3.1.1	Журнал процесса Ingress Controller	89

3.1.2	Журналы Angie	90
3.2	Просмотр состояния сервера	90
3.2.1	Доступ к Stub Status	90
3.3	Просмотр состояния ресурсов	91
3.3.1	Ресурсы Ingress	91
3.3.2	Ресурсы VirtualServer и VirtualServerRoute	91
3.3.3	Ресурсы Policy	93
3.3.4	Ресурсы TransportServer	93
4	Типовые задачи	95
4.1	Создание кастомных страниц ошибок	95
4.1.1	Пересборка образа ANIC с кастомной страницей	95
4.1.2	Использование ConfigMap без пересборки образа	96
4.2	Сопоставление путей с помощью регулярных выражений	97
4.2.1	Пример настройки ресурса Ingress	97
4.2.2	Пример настройки ресурса Mergeable Ingress	99
5	Примеры для пользовательских ресурсов	105
5.1	Базовая конфигурация	106
5.1.1	Предварительные действия	106
5.1.2	Настройка базовой конфигурации	106
5.2	Настройка базовой аутентификации	109
5.2.1	Предварительные действия	109
5.2.2	Настройка базовой аутентификации	109
5.3	Базовая балансировка TCP- и UDP-трафика	112
5.3.1	Предварительные действия	112
5.3.2	Балансировка TCP/UDP-трафика	113
5.4	Контроль доступа	117
5.4.1	Предварительные действия	117
5.4.2	Настройка контроля доступа	117
5.5	Конфигурация для нескольких пространств имен	120
5.5.1	Предварительные действия	120
5.5.2	Настройка конфигурации для нескольких пространств имен	120
5.6	Ограничение скорости запросов	125
5.6.1	Предварительные действия	125
5.6.2	Развертывание веб-приложения	125
5.7	Поддержка переписывания (rewrites)	127
5.7.1	Пример с префиксом в пути	127
5.7.2	Пример с регулярными выражениями	128
5.8	Распределение трафика	128
5.8.1	Предварительные действия	129
5.8.2	Настройка распределения трафика	129
5.9	Расширенная маршрутизация	131
5.9.1	Предварительные действия	132
5.9.2	Настройка расширенной маршрутизации	132
5.10	Сохранение сессий	136
5.10.1	Синтаксис	137
5.10.2	Пример	137
5.11	Cert-manager	138
5.11.1	Развертывание cert-manager и самоподписанного центра сертификации	138
5.11.2	Запуск примера	139
5.12	gRPC	141
5.12.1	Предварительная настройка	141
5.12.2	Пример	142
5.13	Ingress MTLS	142
5.13.1	Предварительные действия	142
5.13.2	Настройка Ingress MTLS	143
5.14	JWKS	145

5.14.1	Предварительные действия	145
5.14.2	Настройка JWKS	146
5.15	JWT	151
5.15.1	Предварительные действия	151
5.15.2	Настройка JWT	151
5.16	OIDC	153
5.16.1	Настройка аутентификации через OpenID Connect	154
5.16.2	Полный пример конфигурации	155
5.16.3	Пример включения переменных <code>map</code> в зависимости от входного значения	156
5.17	TLS Passthrough	157
5.17.1	О Secure App	157
5.17.2	Предварительные действия	157
5.17.3	Настройка TLS Passthrough	157
6	Общие примеры	161
6.1	Пользовательские шаблоны	161
6.1.1	Пример	161
6.1.2	Диагностика ошибок	162
6.2	Пользовательский формат журнала	163
6.3	Протокол PROXY	163
6.3.1	Пример конфигурации	164
6.4	Wildcard-сертификат	164
6.4.1	Пример	164
6.5	Пример <code>default-server-secret</code>	165
7	Известные проблемы и решения	166
7.1	Ошибка <code>"proxy_busy_buffers_size" must be less than the size of all "proxy_buffers" minus one buffer</code>	166
8	Права на интеллектуальную собственность	168

ГЛАВА 1

Аннотация

Angie Ingress Controller (ANIC) — приложение, которое запускается в кластере и управляет балансировщиком нагрузки.

ANIC использует в своей работе [Angie PRO](#) — эффективный, мощный и масштабируемый веб-сервер, который позволяет балансировать нагрузку между серверами как по протоколам TCP/UDP, так и по HTTP.

Примечание

Angie PRO внесен в Единый реестр российских программ для электронных вычислительных машин и баз данных (запись № 17604).

1.1 Общие сведения

Angie Ingress Controller (ANIC) - это решение для управления трафиком контейнеризированных приложений в Kubernetes.

ANIC разворачивается и работает в кластере, управляя функциями Ingress с возможностью настройки правил обработки трафика. Продукт базируется на Angie PRO, что позволяет строить безопасные масштабируемые высокопроизводительные окружения, используя российское решение с профессиональными сервисами миграции и технической поддержки на русском языке.

ANIC использует широкий набор функций Ingress:

- *Балансировка нагрузки TCP, UDP, TLS, HTTP, gRPC*: Гибкое распределение трафика и его плавного переноса при обновлениях приложений
- *Терминирование сессий TLS*: Подтверждения подлинности сервисов и защиты онлайн-транзакций
- *Настройки гибкого логирования*: Управление современными динамическими приложениями
- *Расширенная маршрутизация трафика*: Разделение трафика и расширенная маршрутизация на основе содержимого
- *Ограничение поступающего трафика*: По различным критериям для защиты приложений от DDoS

- *Модификация ответов на запросы:* На уровне балансировщика HTTP

1.2 Системные требования

Список поддерживаемых ОС и архитектур:

ОС	Версии	Архитектуры
Alpine Linux	3.21	x86_64, arm64
Alt Linux	10	x86_64, arm64
Debian	11	x86_64, arm64

ГЛАВА 2

Настройка

Настройка ANIC включает настройку параметров через ConfigMap и аннотации, управление маршрутизацией Ingress-ресурсов и настройку авторизации и SSL для безопасного трафика.

Аргументы командной строки

Настройка ANIC

ConfigMap

GlobalConfiguration

Policy

TransportServer

VirtualServer, VirtualServerRoute

Расширенная конфигурация с помощью аннотаций

2.1 Аргументы командной строки

ANIC поддерживает ряд аргументов командной строки. Способ указания этих аргументов зависит от того, как вы устанавливаете ANIC:

- Если вы используете *манифесты Kubernetes* (Deployment или DaemonSet) для установки ANIC, измените эти манифесты соответствующим образом, чтобы задать аргументы командной строки. См. документацию по установке с манифестами.
- Если вы используете *Helm* для установки ANIC, измените параметры диаграммы Helm, соответствующие аргументам командной строки. См. документацию по установке с помощью Helm.

Ниже в алфавитном порядке перечислены доступные аргументы командной строки:

2.1.1 -angie-configmaps <строка>

Ресурс ConfigMap для настройки конфигурации Angie. Если ConfigMap задан, но ANIC не может получить его из API Kubernetes, то ANIC не запустится.

Формат: <пространство имен>/<имя>

2.1.2 -angie-dashboard

Включает Angie Dashboard (status endpoint для получения информации о состоянии Angie).

2.1.3 -angie-dashboard-port <int>

Задаёт порт, на котором доступен Angie Dashboard (по умолчанию 8082).

Формат: [1024 - 65535]

2.1.4 -angie-dashboard-allow-cidrs <строка>

Добавляет блоки IP/CIDR в список разрешений для Angie Dashboard. Несколько IP или CIDR разделяются запятыми.

2.1.5 -angie-debug

Включает отладку для Angie. Использует бинарник angie-debug. Требуется 'error-log-level: debug' в ConfigMap.

2.1.6 -angie-reload-timeout <значение>

Время ожидания в миллисекундах, в течение которого ANIC будет ожидать успешной перезагрузки Angie после изменения конфигурации или при начальном запуске.

Значение по умолчанию - 60000.

2.1.7 -angie-status

Включает Angie stub_status.

По умолчанию true.

2.1.8 -angie-status-allow-cidrs <строка>

Добавляет блоки IP/CIDR в список разрешений для Angie stub_status.

Несколько IP или CIDR разделяются запятыми. (По умолчанию 127.0.0.1, ::1)

2.1.9 -angie-status-port <int>

Задаёт порт, на котором доступен Angie stub_status.

Формат: [1024 - 65535] (по умолчанию 8080)

2.1.10 -angie-status-prometheus <bool>

Включает или отключает выдачу статистики Angie в формате Prometheus.

Формат: false или true (по умолчанию true)

2.1.11 -angie-status-prometheus-allow-cidrs

Добавляет блоки IP/CIDR в список разрешений для статистики Angie в формате Prometheus.

Несколько IP или CIDR разделяются запятыми. (По умолчанию 127.0.0.1,:::1)

2.1.12 -angie-status-prometheus-path <строка>

Позволяет менять путь для публикации статистики Angie в формате Prometheus.

По умолчанию используется /p8s.

2.1.13 -angie-status-prometheus-port <int>

Задаёт порт, на котором доступна статистика Angie в формате Prometheus.

Формат: [1024 - 65535] (по умолчанию 8083)

2.1.14 -default-server-tls-secret <строка>

Секрет с сертификатом TLS и ключом для TLS-терминации на сервере по умолчанию.

- Если значение не задано, используются сертификат и ключ в файле /etc/angie/secrets/default.
- Если /etc/angie/secrets/default не существует, ANIC настроит в Angie отклонение TLS-подключений к серверу по умолчанию.
- Если секрет установлен, но ANIC не может получить его из API Kubernetes, или же не установлен, и ANIC не удастся прочитать файл /etc/angie/secrets/default, то ANIC не запустится.

Формат: <пространство имен>/<имя>

2.1.15 -disable-ipv6

Явно отключает прослушиватели IPV6 для узлов, которые не поддерживают стек IPV6.

По умолчанию false.

2.1.16 -enable-cert-manager

Включает автоматическое управление сертификатами x509 для ресурсов VirtualServer с помощью cert-manager (cert-manager.io).

Требует *-enable-custom-resources*.

2.1.17 -enable-custom-resources

Включает пользовательские ресурсы.

По умолчанию *true*.

2.1.18 -enable-external-dns

Включает интеграцию с ExternalDNS для настройки общедоступных записей DNS у ресурсов VirtualServer с использованием ExternalDNS.

Требует наличия *-enable-custom-resources*.

2.1.19 -enable-jwt

Включает функцию аутентификации JWT в ресурсах Policy.

По умолчанию *false*.

2.1.20 -enable-leader-election

Позволяет выбирать лидера, чтобы избежать ситуации, когда несколько реплик контроллера общаются о статусе ресурсов Ingress, VirtualServer и VirtualServerRoute; сообщать о статусе будет только одна реплика. По умолчанию *true*.

См. флаг *-report-ingress-status*.

2.1.21 -enable-oidc

Включает функцию аутентификации по OpenID Connect в ресурсах Policy.

По умолчанию *false*.

2.1.22 -enable-prometheus-metrics

Позволяет публиковать метрики Angie в формате Prometheus.

2.1.23 -enable-service-insight

Публикует конечную точку Service Insight для ANIC.

2.1.24 -enable-snippets

Включает пользовательские фрагменты конфигурации Angie в ресурсах Ingress, VirtualServer, VirtualServerRoute и TransportServer.

По умолчанию `false`.

2.1.25 -enable-tls-passthrough

Включает сквозную передачу данных по протоколу TLS на порту 443.

Требует наличия `-enable-custom-resources`.

2.1.26 -external-service <строка>

Указывает имя сервиса с типом LoadBalancer, через который поды ANIC делаются доступными извне. Внешний адрес сервиса используется для отчетов о состоянии ресурсов Ingress, VirtualServer и VirtualServerRoute.

Только для ресурсов Ingress: требует наличия `-report-ingress-status`.

2.1.27 -global-configuration <строка>

Ресурс GlobalConfiguration для глобальной настройки ANIC.

Формат: <пространство имен>/<имя>

Требует наличия `-enable-custom-resources`.

2.1.28 -health-status

Добавляет местоположение `/angie-health` к серверу по умолчанию. Местоположение отвечает кодом статуса 200 на любой запрос.

Это полезно для внешней проверки работоспособности ANIC.

2.1.29 -health-status-uri <строка>

Задаёт URI местоположения проверки работоспособности на сервере по умолчанию. Требует наличия `-health-status`.

По умолчанию `/angie-health`.

2.1.30 -ingress-class <строка>

Класс ANIC.

Должен быть развернут соответствующий ресурс IngressClass с именем, равным классу. В противном случае ANIC не запустится. ANIC обрабатывает только те ресурсы, которые принадлежат его классу, т. е. имеют ресурс поля `ingressClassName`, равный классу.

ANIC обрабатывает все ресурсы, у которых нет поля `ingressClassName`.

По умолчанию `angie`.

2.1.31 -ingresslink <строка>

Указывает имя ресурса IngressLink, через который предоставляется доступ к подам ANIC через систему BIG-IP. IP-адрес системы BIG-IP используется для отчетов о состоянии ресурсов Ingress, VirtualServer и VirtualServerRoute.

Только для ресурсов Ingress: требует наличия *-report-ingress-status*.

2.1.32 -ingress-template-path <строка>

Путь к шаблону конфигурации Ingress Angie для ресурса Ingress. По умолчанию для Angie используется `angie.ingress.tpl`.

2.1.33 -leader-election-lock-name <строка>

Указывает в том же пространстве имен, где находится контроллер, имя ConfigMap, используемое для блокировки при выборе лидера.

Требует наличия *-enable-leader-election*.

2.1.34 -main-template-path <строка>

Путь к основному шаблону конфигурации Angie.

- По умолчанию для Angie используется `angie.ingress.tpl`.

2.1.35 -prometheus-metrics-listen-port <int>

Задаёт порт, на котором публикуются метрики Prometheus.

Формат: [1024 - 65535] (по умолчанию 9113)

2.1.36 -prometheus-tls-secret <строка>

Секрет с сертификатом TLS и ключом для TLS-терминации конечной точки метрик Prometheus.

- Если аргумент не задан, конечная точка Prometheus не будет использовать TLS-соединение.
- Если аргумент задан, но ANIC не может получить секрет из API Kubernetes, то ANIC не запустится.

2.1.37 -proxy <строка>

Задаёт использование прокси-сервера для подключения к API Kubernetes, запускаемого командой "kubectl proxy". **Только в целях тестирования.**

ANIC не запускает Angie и не записывает на диск никакие сгенерированные файлы конфигурации Angie.

2.1.38 -ready-status

Включает конечную точку готовности `/angie-ready`. Конечная точка возвращает код успеха, когда Angie загрузил всю конфигурацию после запуска.

По умолчанию `true`.

2.1.39 -ready-status-port

HTTP-порт для конечной точки готовности.

Формат: [1024 - 65535] (по умолчанию 8081)

2.1.40 -report-ingress-status

Обновляет поле адреса в статусе ресурсов Ingress.

Требуется флаг `-external-service` или `-ingresslink`, либо ключ `external-status-address` в ConfigMap.

2.1.41 -service-insight-listen-port <int>

Задаёт порт, на котором публикуется Service Insight.

Формат: [1024 - 65535] (по умолчанию 9114)

2.1.42 -service-insight-tls-secret <строка>

Секрет с сертификатом TLS и ключом для TLS-терминации конечной точки Service Insight.

- Если аргумент не задан, конечная точка Service Insight не будет использовать TLS-соединение.
- Если аргумент задан, но ANIC не может получить секрет из API Kubernetes, то ANIC не запустится.

Формат: <пространство имен>/<имя>

2.1.43 -tls-passthrough-port <int>

Задаёт порт для сквозной передачи данных по протоколу TLS. Формат: [1024 - 65535] (по умолчанию 443)

Требуется включить `-enable-custom-resources`.

2.1.44 -transportserver-template-path <строка>

Путь к шаблону конфигурации TransportServer Angie для ресурса TransportServer.

- По умолчанию для Angie используется `angie.transportserver.tpl`.

2.1.45 -v <значение>

Уровень детализации записи логов. Значение по умолчанию — 1, при этом значении записывается минимальное количество логов. Значение 3 полезно для устранения неполадок.

2.1.46 -version

Выводит версию, хэш git-коммита и дату сборки, затем завершает работу.

2.1.47 -virtualserver-template-path <строка>

Путь к шаблону конфигурации VirtualServer Angie для ресурса VirtualServer.

- По умолчанию для Angie используется `angie.ingress.tpl`.

2.1.48 -vmodule <значение>

Разделенный запятыми список параметров `pattern=N` для ведения журнала с фильтрацией файлов.

2.1.49 -watch-namespace <строка>

Разделенный запятыми список пространств имен, за ресурсами которых должен следить ANIC. По умолчанию ANIC отслеживает все пространства имен. Нельзя использовать вместе с `"watch-namespace-label"`.

2.1.50 -watch-namespace-label <строка>

Настраивает в ANIC просмотр только пространств имен с меткой `foo=bar`. По умолчанию ANIC отслеживает все пространства имен. Нельзя использовать вместе с `"watch-namespace"`.

2.1.51 -watch-secret-namespace <строка>

Разделенный запятыми список пространств имен, за которыми ANIC должен следить на предмет наличия секретов. Если этот параметр не настроен, ANIC отслеживает одни и те же пространства имен для всех ресурсов. См. также `"watch-namespace"` и `"watch-namespace-label"`.

2.1.52 -wildcard-tls-secret <строка>

Секрет с сертификатом TLS и ключом для TLS-терминации каждого узла Ingress или VirtualServer, для которого включена TLS-терминация, но секрет не указан.

- Если аргумент не задан, для таких узлов Ingress и VirtualServer Angie прервет любую попытку установить TLS-соединение.
- Если аргумент задан, но ANIC не может получить секрет из API Kubernetes, то ANIC не запустится.

Формат: <пространство имен>/<имя>

2.2 Настройка ANIC

Здесь приведены параметры настройки ANIC. ANIC настраивается путем изменения параметров ConfigMap и Annotation.

Список 1: Пример ConfigMap

```
$ kubectl apply -f - <<EOF
kind: ConfigMap
apiVersion: v1
metadata:
  name: angie-config
  namespace: angie-ingress
data:
  proxy-connect-timeout: "10s"
  proxy-read-timeout: "10s"
  client-max-body-size: "2m"
EOF
```

2.2.1 Параметры Ingress Controller

external-status- Задаёт адрес, который выводится в статусе Ingress ресурса. Имеет приоритет над аргументом командной строки **-external-service**.

2.2.2 Общие параметры

Примечание

Для всех параметров типа **boolean** допустимы пары значений **true/false**, **t/f**, **on/off** и **1/0**. Регистр не имеет значения.

Параметр	Описание	Умолчание
proxy-connect-timeout	Задаёт значение proxy_connect_timeout и grpc_connect_timeout.	60s
proxy-read-timeout	Задаёт значение proxy_read_timeout и grpc_read_timeout	60s
proxy-send-timeout	Задаёт значение proxy_send_timeout и grpc_send_timeout	60s
client-max-body-size	Задаёт значение client_max_body_size	1m
proxy-buffering	Включает или отключает буферизацию ответов от проксируемого сервера	True
proxy-buffers	Задаёт значение proxy_buffers	Зависит от платформы
proxy-buffer-size	Задаёт значение proxy_buffer_size и grpc_buffer_size	Зависит от платформы
proxy-max-temp-file-size	Задаёт значение proxy_max_temp_file_size	1024m
set-real-ip-from	Задаёт значение set_real_ip_from	Нет
real-ip-header	Задаёт значение real_ip_header	X-Real-IP
real-ip-recursive	Включает или отключает real_ip_recursive	False

продолжается на следующей странице

Таблица 1 – продолжение с предыдущей страницы

Параметр	Описание	Умолчение
default-server-return	Настраивает ответ в сервере по умолчанию, который перехватывает клиентский запрос, если для запроса не был определен ресурс Ingress или VirtualServer . Можно установить фиксированный ответ или перенаправление запроса.	Страница с ошибкой HTTP 404
server-tokens	Включает или отключает <code>server_tokens</code>	True
worker-processes	Задаёт значение <code>worker_processes</code>	auto
worker-rlimit-nofile	Задаёт значение <code>worker_rlimit_nofile</code>	Нет
worker-connections	Задаёт значение <code>worker_connections</code>	1024
worker-cpu-affinity	Задаёт значение <code>worker_cpu_affinity</code>	Нет
worker-shutdown-timeout	Задаёт значение <code>worker_shutdown_timeout</code>	Нет
server-names-hash-bucket-size	Задаёт значение <code>server_names_hash_bucket_size</code>	256
server-names-hash-max-size	Задаёт значение <code>server_names_hash_max_size</code>	1024
map-hash-bucket-size	Задаёт значение <code>map_hash_bucket_size</code>	256
map-hash-max-size	Задаёт значение <code>map_hash_max_size</code>	2048
resolver-addresses	Задаёт значение DNS resolver	Нет
resolver-ipv6	Разрешает или запрещает поиск IPv6-адресов	True
resolver-valid	Позволяет переопределить срок кэширования DNS-записей	Нет
resolver-timeout	Задаёт значение <code>resolver_timeout</code>	30s
keepalive-timeout	Задаёт значение <code>keepalive_timeout</code>	65s
keepalive-requests	Задаёт значение <code>keepalive_requests</code>	100
variables-hash-bucket-size	Задаёт значение <code>variables_hash_bucket_size</code>	256
variables-hash-max-size	Задаёт значение <code>variables_hash_max_size</code>	1024

2.2.3 Параметры ведения журнала

Параметр	Описание	Умолчение
<code>error-log-level</code>	Определяет глобальное значение уровня <code>error_log</code> и может принимать одно из следующих значений: <i>debug</i> , <i>info</i> , <i>notice</i> , <i>warn</i> , <i>error</i> , <i>crit</i> , <i>alert</i> или <i>emerg</i>	<code>notice</code>
<code>access-log-off</code>	Отключает <code>access_log</code>	<code>False</code>
<code>default-server-access-log-off</code>	Отключает <code>access_log</code> для сервиса по умолчанию	<code>False</code>
<code>log-format</code>	Задаёт общий формат журнала. Для удобства можно использовать несколько строк, разделённых <code>\n</code> . В этом случае каждый перевод строки будет заменён на пробел. Все символы <code>'</code> должны быть экранированы	Нет
<code>log-format-escaping</code>	Позволяет задать экранирование символов <code>json</code> или <code>default</code> в переменных; по умолчанию используется <code>default</code> . Значение <code>none</code> отключает экранирование	<code>default</code>
<code>stream-log-format</code>	Задаёт формат журнала <code>stream</code> для сквозного трафика TCP, UDP и TLS. Для удобства можно использовать несколько строк, разделённых <code>\n</code> . В этом случае каждый перевод строки будет заменён на пробел. Все символы <code>'</code> должны быть экранированы	Нет
<code>stream-log-format-escaping</code>	Позволяет задать экранирование символов <code>json</code> или <code>default</code> в переменных; по умолчанию используется <code>default</code> . Значение <code>none</code> отключает экранирование	<code>default</code>

2.2.4 Управление URI и заголовками в запросах

<code>proxy-hide-headers</code>	Значение одного <code>proxy_hide_header</code> или нескольких
<code>proxy-pass-headers</code>	Значение одного <code>proxy_pass_header</code> или нескольких

2.2.5 Авторизация и SSL/TLS

Параметр	Описание	Умолчение
<code>redirect-to-https</code>	Задаёт правило 301 redirect в зависимости от заголовка <code>http_x_forwarded_proto</code>	<code>False</code>
<code>ssl-redirect</code>	Задаёт правило 301 redirect для всего входящего HTTP-трафика, чтобы перевести запросы в HTTPS	<code>True</code>
<code>ssl-protocols</code>	Задаёт значение <code>ssl_protocols</code>	TLSv1 TLSv1.1 TLSv1.2
<code>ssl-prefer-server-ciphers</code>	Включает или отключает <code>ssl_prefer_server_ciphers</code>	<code>False</code>
<code>ssl-ciphers</code>	Задаёт значение <code>ssl_ciphers</code>	HIGH:!aNULL:!MD5
<code>ssl-dhparam-file</code>	Указывает файл с параметрами для DHE-шифров	Нет

2.2.6 Протоколы

Параметр	Описание	Умолчение
http2	Включает поддержку протокола HTTP/2	False
proxy-protocol	Указывает, что все соединения, принимаемые на данном слушающем сокете, должны использовать протокол PROXY	False

2.2.7 Апстримы

Параметр	Описание	Умолчение
max-fails	Задаёт значение <code>max_fails</code> для сервера	1
upstream-zone-size	Задаёт имя и размер зоны разделяемой памяти	Нет
fail-timeout	Задаёт значение <code>fail_timeout</code> для сервера	10s
keepalive	Задействует кэш соединений для группы серверов апстрима	Нет

2.2.8 Настраиваемые шаблоны

main-snippets	Вставляет собственный фрагмент конфигурации в контекст main
http-snippets	Вставляет собственный фрагмент конфигурации в контекст http
location-snippets	Вставляет собственный фрагмент конфигурации в контекст location
server-snippets	Вставляет собственный фрагмент конфигурации в контекст server
stream-snippets	Вставляет собственный фрагмент конфигурации в контекст main
main-template	Определяет основной шаблон для основных настроек Angie. По умолчанию шаблон считывается из файла в контейнере
ingress-template	Определяет шаблон настроек для ресурса Ingress. По умолчанию шаблон считывается из файла в контейнере
virtualserver-template	Определяет шаблон настроек для ресурса <i>VirtualServer</i> . По умолчанию шаблон считывается из файла в контейнере

2.3 ConfigMap

ConfigMap позволяет настраивать поведение Angie. Например, можно задать количество рабочих процессов или настроить формат журнала доступа.

2.3.1 Использование ConfigMap

1. Наши инструкции по установке с манифестами развертывают пустой ConfigMap, в то время как манифесты установки по умолчанию указывают ее в аргументах командной строки ANIC. Однако, если вы настроили манифесты, чтобы использовать ConfigMap, обязательно укажите ресурс ConfigMap для использования с помощью *аргументов командной строки* ANIC.
2. Создайте файл ConfigMap с именем `angie-config.yaml` и установите значения, которые имеют смысл для вашей среды:

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: angie-config
  namespace: angie-ingress
data:
  proxy-connect-timeout: 10s
  proxy-read-timeout: 10s
  client-max-body-size: 2m
```

См. в разделе *Краткое описание ключей ConfigMap* сведения о доступных ключах ConfigMap (таких как `proxy-connect-timeout` в этом примере).

3. Создайте новый (или обновите существующий) ресурс ConfigMap:

```
kubectl apply -f angie-config.yaml
```

Конфигурация Angie будет обновлена.

2.3.2 ConfigMap и аннотации Ingress

Аннотации позволяют настраивать расширенные функции Angie и менять поведение Angie.

ConfigMap применяется глобально, то есть влияет на каждый ресурс Ingress. Напротив, аннотации всегда применяются только к своему ресурсу Ingress. Аннотации позволяют переопределять некоторые ключи ConfigMap. Например, в `angie.software/proxy-connect-timeout` аннотации переопределяют ключ конфигурации `proxy-connect-timeout`.

2.3.3 Переопределение ConfigMap для конкретного ресурса Ingress с помощью аннотации

Вы можете применять разные конфигурации ConfigMap к Ingress-ресурсам в зависимости от того, какое пространство имен указано в конфигурации. Аннотация `angie.software/configmap` позволяет задать конкретный ConfigMap для настройки ресурса Ingress. Заданный ConfigMap будет иметь приоритет над глобальным. В случае, если глобальный и заданный ConfigMap совпадают, применится заданный.

Чтобы применить конкретный ConfigMap к ресурсу Ingress:

1. Создайте ConfigMap с указанием нужного пространства имен.

Например:

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: echoserver-new-config
  namespace: echoserver-new
data:
  server-snippets: |
```



```
location /echoserver-new-snippet {
    return 302 /echo-test-2;
}
```

- Укажите аннотацию `angie.software/configmap` в ресурсе Ingress, к которому нужно применить этот ConfigMap.

Например:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    angie.software/configmap: "echoserver-new/echoserver-new-config"
  name: echoserver-new
  namespace: echoserver-new
spec:
  ingressClassName: angie
  rules:
  - host: test-new.example.com
    http:
      paths:
      - backend:
          service:
            name: echoserver-new
            port:
              number: 8077
        pathType: ImplementationSpecific
```

В этом примере аннотация `angie.software/configmap` указывает на использование конфигурации из ConfigMap `echoserver-new-config`. Это означает, что директивы, описанные в `server-snippets` из этого ConfigMap, будут применяться к запросам, обрабатываемым этим Ingress.

См. также документацию по *расширенной конфигурации с помощью аннотаций*.

2.3.4 ConfigMap и ресурсы VirtualServer, VirtualServerRoute

ConfigMap влияет на все ресурсы VirtualServer и VirtualServerRoute. Однако поля этих ресурсов позволяют переопределять некоторые ключи ConfigMap. Например, поле `connect-timeout` сервера апстрима имеет приоритет над ключом ConfigMap `proxy-connect-timeout`.

См. документацию по *ресурсам VirtualServer и VirtualServerRoute*.

2.3.5 Краткое описание ключей ConfigMap

Примечание

Для всех параметров типа `boolean` допустимы пары значений `true / false`, `t / f`, `on / off` и `1 / 0`. Регистр не имеет значения.

ANIC (не связанные с конфигурацией Angie)

Ключ ConfigMa	Описание	По умол- чанию	При- мер
<code>external</code>	Задаёт адрес, который будет отображаться в статусе ресурсов Ingress. Требуется аргумент командной строки <code>-report-status</code> . Имеет приоритет над аргументом <code>-external-service</code> .	Н/Д	<i>От- чет о состо- янии Ingress</i>

Общая настройка

Ключ ConfigMap	Описание	По умолчанию	Пример
<code>proxy-connect-timeout</code>	Задаёт значение директивы <code>proxy_connect_timeout</code> .	60s	
<code>proxy-read-timeout</code>	Задаёт значение директивы <code>proxy_read_timeout</code> .	60s	
<code>proxy-send-timeout</code>	Задаёт значение директивы <code>proxy_send_timeout</code> .	60s	
<code>client-max-body-size</code>	Задаёт значение директивы <code>client_max_body_size</code> .	1m	
<code>proxy-buffer-size</code>	Включает или отключает буферизацию ответов от проксируемого сервера.	True	
<code>proxy-buffer-size</code>	Задаёт значение директивы <code>proxy_buffers</code> .	Зависит от платформы.	
<code>proxy-buffer-size</code>	Задаёт значение директив <code>proxy_buffer_size</code> и <code>grpc_buffer_size</code> .	Зависит от платформы.	
<code>proxy-max-temp-file-size</code>	Задаёт значение директивы <code>proxy_max_temp_file_size</code> .	1024m	
<code>set-real-ip-from</code>	Задаёт значение директивы <code>set_real_ip_from</code> .	Н/Д	
<code>real-ip-header</code>	Задаёт значение директивы <code>real_ip_header</code> .	X-Real-IP	
<code>real-ip-recursive</code>	Включает или отключает директиву <code>real_ip_recursive</code> .	False	
<code>default-server-return</code>	Настраивает директиву <code>return</code> на сервере по умолчанию, которая обрабатывает клиентский запрос, если ни один из узлов ресурсов Ingress или VirtualServer не совпадает. Значение по умолчанию настраивает в Angie возврат страницы с ошибкой 404. Вы можете настроить фиксированный ответ или перенаправление. Например, значение <code>default-server-return: 302 https://mysite.ru:samp;</code> перенаправит клиент на <code>https://mysite.ru</code> .	404	
<code>server-tokens</code>	Включает или отключает директиву <code>server_tokens</code> .	True	
<code>worker-processes</code>	Задаёт значение директивы <code>worker_processes</code> .	auto	
<code>worker-rlimit-nofile</code>	Задаёт значение директивы <code>worker_rlimit_nofile</code> .	Н/Д	
<code>worker-connections</code>	Задаёт значение директивы <code>worker_connections</code> .	1024	
<code>worker-cpu-affinity</code>	Задаёт значение директивы <code>worker_cpu_affinity</code> .	Н/Д	
<code>worker-shutdown-timeout</code>	Задаёт значение директивы <code>worker_shutdown_timeout</code> .	Н/Д	
<code>server-names-hash-bucket-size</code>	Задаёт значение директивы <code>server_names_hash_bucket_size</code> .	256	
<code>server-names-hash-max-size</code>	Задаёт значение директивы <code>server_names_hash_max_size</code> .	1024	
<code>map-hash-bucket-size</code>	Задаёт значение директивы <code>map_hash_bucket_size</code> .	256	
<code>map-hash-max-size</code>	Задаёт значение директивы <code>map_hash_max_size</code> .	2048	
<code>resolver</code>	Задаёт значение адресов resolver.	Н/Д	
<code>resolver</code>	Включает разрешение IPv6 в распознавателе.	True	
<code>resolver-timeout</code>	Задаёт значение директивы <code>resolver_timeout</code> для разрешения имен.	30s	
<code>keepalive-timeout</code>	Задаёт значение директивы <code>keepalive_timeout</code> .	65s	
<code>keepalive-requests</code>	Задаёт значение директивы <code>keepalive_requests</code> .	100	
<code>variables-hash-bucket-size</code>	Задаёт значение директивы <code>variables_hash_bucket_size</code> .	256	
<code>variables-hash-max-size</code>	Задаёт значение директивы <code>variables_hash_max_size</code> .	1024	


Ведение журнала

Ключ ConfigMap	Описание	По умолчанию	Пример
<code>error-log</code>	Задаёт глобальный уровень журнала ошибок для Angie.	<code>notice</code>	
<code>access-log</code>	Отключает журнал доступа.	<code>False</code>	
<code>default-log</code>	Отключает журнал доступа для сервера по умолчанию. Если журнал доступа отключен глобально (<code>access-log-off: "True"</code>), то журнал доступа к серверу по умолчанию всегда отключен.	<code>False</code>	
<code>log-format</code>	Задаёт настраиваемый формат журнала для HTTP- и HTTPS-трафика. Для удобства можно определить формат журнала в нескольких строках (строки разделяются символом <code>\n</code>). В этом случае ANIC заменит каждый символ <code>\n</code> символом пробела. Все символы <code>'</code> должны быть экранированы.		
<code>log-format-escape</code>	Задаёт экранирующие символы для переменных формата журнала. Поддерживаемые значения: <code>json</code> (экранирование JSON), <code>default</code> (экранирование по умолчанию), <code>none</code> (отключает экранирование).	<code>default</code>	
<code>stream-log-format</code>	Задаёт настраиваемый формат журнала <code><s_log_format></code> для сквозного трафика TCP, UDP и TLS. Для удобства можно определить формат журнала в нескольких строках (строки разделяются символом <code>\n</code>). В этом случае ANIC заменит каждый символ <code>\n</code> символом пробела. Все символы <code>'</code> должны быть экранированы.		
<code>stream-log-format-escape</code>	Задаёт экранирующие символы для переменных формата журнала потока. Поддерживаемые значения: <code>json</code> (экранирование JSON), <code>default</code> (экранирование по умолчанию), <code>none</code> (отключает экранирование).	<code>default</code>	

Манипулирование URI и заголовками запроса

Ключ ConfigMap	Описание	По умолчанию	Пример
<code>proxy-hide-headers</code>	Задаёт значение одной директивы <code>proxy_hide_header</code> или нескольких.	Н/Д	<code>"angie. software/ proxy-hide-headers": "header-a, header-b"</code>
<code>proxy-pass-headers</code>	Задаёт значение одной директивы <code>proxy_pass_header</code> или нескольких.	Н/Д	<code>"angie. software/ proxy-pass-headers": "header-a, header-b"</code>

Аутентификация, SSL, TLS

Ключ ConfigMa	Описание	По умолчанию	Пример
<code>redirect</code>	Задаёт правило перенаправления 301 на основе значения заголовка <code>http_x_forwarded_proto</code> в серверном блоке, требуя, чтобы входящий трафик шел по протоколу HTTPS. Полезно при терминеции SSL в системе балансировки нагрузки перед ANIC.	False	
<code>ssl-redirect</code>	Задаёт безусловное правило перенаправления 301 для всего входящего HTTP-трафика, требуя, чтобы входящий трафик шел по протоколу HTTPS.	True	
<code>hsts</code>	Включает режим HTTP Strict Transport Security (HSTS): заголовок HSTS добавляется к ответам от проксируемых серверов. Директива <code>preload</code> будет включена в заголовок.	False	
<code>hsts-max</code>	Задаёт значение директивы <code>max-age</code> заголовка HSTS.	2592000 (1 месяц)	
<code>hsts-include</code>	Добавляет директиву <code>includeSubDomains</code> в заголовок HSTS.	False	
<code>hsts-headers</code>	Включает HSTS на основе значения заголовка запроса <code>http_x_forwarded_proto</code> . Следует использовать только в том случае, если в балансировщике нагрузки (прокси-сервере) перед ANIC настроена терминеция TLS.	False	
<div>  Примечание </div> <p>Чтобы управлять перенаправлением с HTTP на HTTPS, настройте аннотацию <code>angie.software/redirect-to-https</code>.</p>			
<code>ssl-protocols</code>	Задаёт значение директивы <code>ssl_protocols</code> .	TLSv1 TLSv1.1 1 TLSv1.2	
<code>ssl-prefer_server_ciphers</code>	Включает или отключает директиву <code>ssl_prefer_server_ciphers</code> .	On	
<code>ssl_ciphers</code>	Задаёт значение директивы <code>ssl_ciphers</code> .	HIGH:!aNULL:!MD5	
<code>ssl_dhparam</code>	Задаёт содержимое файла <code>dhparam</code> . Контроллер создаст файл и установит значение директивы <code>ssl_dhparam</code> с указанием пути к файлу.	Н/Д	

Прослушиватели

Ключ ConfigMa	Описание	По умолчанию	Пример
<code>http2</code>	Включает HTTP/2 на серверах с включенным SSL.	False	
<code>proxy-protocol</code>	Включает прокси-протокол для входящих соединений.	False	

Бэкенд-сервисы (апстримы)

Ключ ConfigMap	Описание	По умол- чанию	При- мер
<code>lb-method</code>	Задаёт метод балансировки нагрузки. Чтобы использовать циклический метод, укажите <code>"round_robin"</code> .	<code>"random two least_conn"</code>	
<code>max-fail</code>	Задаёт значение параметра <code>max_fails</code> директивы <code>u_server</code> .	<code>1</code>	
<code>upstream</code>	Задаёт размер зоны разделяемой памяти для апстримов.		
<code>fail-time</code>	Задаёт значение параметра <code>fail_timeout</code> директивы <code>u_server</code> .	<code>10s</code>	
<code>keepalive</code>	Задаёт значение директивы <code>u_keepalive</code> . Обратите внимание: если значение больше 0, в сгенерированную конфигурацию добавляется <code>proxy_set_header Connection ""</code> ;	<code>0</code>	

Фрагменты и пользовательские шаблоны

Ключ ConfigMap	Описание	По умолчанию	Пример
<code>main-sni</code>	Задаёт пользовательский фрагмент в основном контексте.	Н/Д	
<code>http-sni</code>	Задаёт пользовательский фрагмент в контексте http.	Н/Д	
<code>location</code>	Задаёт пользовательский фрагмент в контексте location.	Н/Д	
<code>server-ssl</code>	Задаёт пользовательский фрагмент в контексте server.	Н/Д	
<code>stream-ssl</code>	Задаёт пользовательский фрагмент в контексте stream.	Н/Д	
<code>main-template</code>	Задаёт основной шаблон конфигурации Angie.	По умолчанию считывается из файла в контейнере.	
<code>ingress-template</code>	Задаёт шаблон конфигурации Angie для ресурса Ingress.	По умолчанию считывается из файла в контейнере.	
<code>virtual-server-template</code>	Задаёт шаблон конфигурации Angie для ресурса VirtualServer.	По умолчанию считывается из файла в контейнере.	

2.4 GlobalConfiguration

Ресурс GlobalConfiguration позволяет вам определить глобальные параметры конфигурации ANIC. Он реализован как [пользовательский ресурс](#).

Ресурс поддерживает настройку прослушивателей для балансировки нагрузки TCP и UDP. Прослушиватели требуются [ресурсам TransportServer](#).

2.4.1 Предварительные требования

При установке ANIC с манифестами необходимо указать ссылку на ресурс GlobalConfiguration в аргументе командной строки `-global-configuration`. Для ANIC требуется только один ресурс GlobalConfiguration.

2.4.2 Спецификация GlobalConfiguration

Ресурс GlobalConfiguration определяет глобальные параметры конфигурации ANIC. Ниже приведен пример:

```
apiVersion: k8s.angie.software/v1alpha1
kind: GlobalConfiguration
metadata:
  name: angie-configuration
  namespace: angie-ingress
spec:
  listeners:
  - name: dns-udp
    port: 5353
    protocol: UDP
  - name: dns-tcp
    port: 5353
    protocol: TCP
```

Поле	Описание	Тип	Обязательно
<code>listeners</code>	Список прослушивателей.	<code>listener[]</code>	Нет

Прослушиватель

Прослушиватель определяет комбинацию протокола и порта, которые Angie будет использовать при приеме трафика для [TransportServer](#):

```
name: dns-tcp
port: 5353
protocol: TCP
```


Поле	Описание	Тип	Обязательно
<code>name</code>	Имя прослушивателя. Это должна быть допустимая метка DNS, как определено в RFC 1035. Например, допустимы значения <code>hello</code> и <code>listener-123</code> . Имя должно быть уникальным среди всех прослушивателей. Имя <code>tls-passthrough</code> зарезервировано для встроенного прослушивателя TLS Passthrough и не может быть использовано.	<code>string</code>	Да
<code>port</code>	Порт прослушивателя. Порт должен находиться в диапазоне 1..65535 со следующими исключениями: 80, 443, порт [статуса](/angie-ingress-controller/logging-and-monitoring/status-page). Комбинация порта и протокола должна быть уникальна среди всех прослушивателей.	<code>int</code>	Да
<code>protocol</code>	Протокол прослушивателя. Поддерживаемые значения: TCP и UDP.	<code>string</code>	Да

2.4.3 Использование GlobalConfiguration

Вы можете использовать обычные команды `kubectl` для работы с ресурсом `GlobalConfiguration`.

Например, следующая команда создает ресурс `GlobalConfiguration`, определенный в `global-configuration.yaml` с именем `angie-configuration`:

```
$ kubectl apply -f global-configuration.yaml

globalconfiguration.k8s.angie.software/angie-configuration created
```

Предполагая, что пространство имен ресурса называется `angie-ingress`, вы можете получить ресурс, запустив:

```
$ kubectl get globalconfiguration angie-configuration -n angie-ingress

NAME                      AGE
angie-configuration       13s
```

В `kubectl get` и подобных командах также можно использовать короткое имя `gc` вместо `globalconfiguration`.

Валидация

Для ресурса `GlobalConfiguration` доступны два типа валидации:

- *Структурная валидация* с помощью `kubectl` и сервера Kubernetes API.
- *Всесторонняя валидация* с помощью ANIC.

Структурная валидация

Пользовательское определение ресурса для `GlobalConfiguration` включает структурную схему OpenAPI, которая описывает тип каждого поля ресурса.

Если вы попытаетесь создать (или обновить) ресурс с нарушением структурной схемы (например, используете строковое значение для поля порта прослушивателя), `kubectl` и сервер Kubernetes API отклонят такой ресурс:

- Пример проверки `kubectl`:


```
$ kubectl apply -f global-configuration.yaml

error: error validating "global-configuration.yaml": error validating
data: ValidationError(GlobalConfiguration.spec.listeners[0].port):
invalid type for
software.angie.k8s.v1alpha1.GlobalConfiguration.spec.listeners.port:
got "string", expected "integer"; if you choose to ignore these
errors, turn validation off with --validate=false
```

- Пример проверки сервера API Kubernetes:

```
$ kubectl apply -f global-configuration.yaml --validate=false

The GlobalConfiguration "angie-configuration" is invalid: []: Invalid
value: map[string]interface {}{ ... }: validation failure list:
spec.listeners.port in body must be of type integer: "string"
```

Если ресурс не отклонен (то есть не нарушает структурную схему), ANIC проверит его дополнительно.

Всесторонняя валидация

ANIC проверяет поля ресурса GlobalConfiguration. Если ресурс недопустим, ANIC не будет его использовать. Рассмотрим следующие два случая:

1. Если при запуске пода ANIC ресурс GlobalConfiguration недопустим, ANIC не сможет запуститься и завершит работу с ошибкой.
2. Если ресурс GlobalConfiguration становится недействительным, когда ANIC запущен, то ANIC проигнорирует новую версию. Он сообщит об ошибке и продолжит использовать предыдущую версию. Когда ресурс снова станет действительным, ANIC начнет его использовать.

Примечание

Если ресурс GlobalConfiguration был удален во время работы ANIC, тот продолжит использовать предыдущую версию ресурса.

Вы можете проверить, успешно ли ANIC применил конфигурацию для GlobalConfiguration. Для нашего ресурса GlobalConfiguration *angie-configuration* мы можем запустить:

```
$ kubectl describe gc angie-configuration -n angie-ingress

. . .
Events:
  Type          Reason          Age   From                      Message
  --          -
  Normal        Updated         11s   angie-ingress-controller  GlobalConfiguration
angie-ingress/angie-configuration was updated
```

Обратите внимание, что раздел "События" (Events) включает событие Normal с причиной Updated, которое информирует нас о том, что конфигурация была успешно применена.

Если вы создадите недопустимый ресурс, ANIC отклонит его и выдаст событие Rejected. Например, если вы создадите ресурс GlobalConfiguration *angie-configuration* с несколькими прослушивателями, для которых задан один и тот же протокол UDP и порт 53, вы получите:


```
$ kubectl describe gc angie-configuration -n angie-ingress
```

```
. . .
```

```
Events:
```

Type	Reason	Age	From	Message
Normal	Updated	55s	angie-ingress-controller	GlobalConfiguration angie-ingress/angie-configuration was updated
Warning	Rejected	6s	angie-ingress-controller	GlobalConfiguration angie-ingress/angie-configuration is invalid and was rejected: spec.listeners: Duplicate value: "Duplicated port/protocol combination 53/UDP"

Обратите внимание, что раздел "События" (Events) включает предупреждающее событие с указанием причины отклонения.

2.5 Policy

Ресурс Policy позволяет настраивать такие функции, как контроль доступа и ограничение скорости; их можно добавить к вашим *ресурсам VirtualServer и VirtualServerRoute*.

Он реализован как *пользовательский ресурс*.

Это справочная документация по ресурсу Policy.

2.5.1 Предварительные требования

Политики работают совместно с *ресурсами VirtualServer и VirtualServerRoute*, которые необходимо создавать отдельно.

2.5.2 Спецификация Policy

Ниже приведен пример политики, которая разрешает доступ клиентам из подсети 10.0.0.0/8 и запрещает доступ любым другим:

```
apiVersion: k8s.angie.software/v1
kind: Policy
metadata:
  name: allow-localhost
spec:
  accessControl:
    allow:
      - 10.0.0.0/8
```


Поле	Описание	Тип	Обязательно
<code>accessControl</code>	Политика контроля доступа, основанная на IP-адресе клиента.	<i>AccessControl</i>	Нет
<code>ingressPolicy</code>	Указывает, какой экземпляр ANIC должен обрабатывать ресурс Policy.	<i>string</i>	Нет
<code>rateLimit</code>	Политика ограничения скорости управляет скоростью обработки запросов по определенному ключу.	<i>RateLimit</i>	Нет
<code>basicAuth</code>	Политика базовой аутентификации настраивает в Angie аутентификацию клиентских запросов с использованием базовой аутентификации HTTP по учетным данным.	<i>BasicAuth</i>	Нет
<code>ingressMTLS</code>	Политика IngressMTLS настраивает проверку сертификата клиента.	<i>IngressMTLS</i>	Нет
<code>egressMTLS</code>	Политика EgressMTLS настраивает аутентификацию и проверку сертификата апстрима.	<i>EgressMTLS</i>	Нет
<code>OIDC</code>	Политика OIDC настраивает аутентификацию через провайдера OIDC.	<i>OIDC</i>	Нет
<code>JWT</code>	Политика JWT настраивает Angie для аутентификации запросов клиентов с использованием JSON Web Tokens.	<i>JWT</i>	Нет

Примечание

Политика должна включать в себя ровно одно значение.

AccessControl

Политика контроля доступа настраивает в Angie отклонение или принятие запросов от клиентов с указанными IP-адресами и подсетями.

Например, следующая политика разрешает доступ клиентам из подсети 10.0.0.0/8 и запрещает доступ любым другим:

```
accessControl:
  allow:
    - 10.0.0.0/8
```

Напротив, приведенная ниже политика делает обратное: запрещает доступ клиентам с 10.0.0.0/8 и разрешает доступ любым другим клиентам:

```
accessControl:
  deny:
    - 10.0.0.0/8
```

Примечание

Функция реализована с использованием модуля Angie `http_access`. Политика контроля доступа ANIC поддерживает либо разрешающие, либо запрещающие правила, но не оба вида сразу (в отличие от модуля).

По- ле	Описание	Тип	Обяза- тельно
<code>allow</code>	Разрешает доступ для указанных сетей или адресов. Например, 192.168.1.1 или 10.1.1.0/16.	<code>string[]</code>	Нет
<code>deny</code>	Запрещает доступ для указанных сетей или адресов. Например, 192.168.1.1 или 10.1.1.0/16.	<code>string[]</code>	Нет

AccessControl должен включать либо `allow`, либо `deny`.

Поведение слияния AccessControl

Ресурс VirtualServer или VirtualServerRoute может ссылаться на несколько политик контроля доступа. Например, здесь мы ссылаемся на две политики, в каждой из которых настроен список разрешений:

```
policies:
- name: allow-policy-one
- name: allow-policy-two
```

Когда вы ссылаетесь на несколько политик контроля доступа, ANIC объединит их содержимое в один список разрешений или запретов.

Ссылки как на разрешающие, так и на запрещающие политики, как показано в примере ниже, не поддерживаются. Если указаны ссылки как на разрешающие, так и на запрещающие списки, ANIC использует только политики разрешающих списков.

```
policies:
- name: deny-policy
- name: allow-policy-one
- name: allow-policy-two
```

RateLimit

Политика ограничения скорости настраивает в Angie ограничение скорости обработки запросов.

Например, следующая политика ограничит все последующие запросы, поступающие с одного IP-адреса, при превышении скорости в 10 запросов в секунду:

```
rateLimit:
rate: 10r/s
zoneSize: 10M
key: ${binary_remote_addr}
```

Примечание

Функция реализована с использованием модуля Angie `http_limit_req`.

Поле	Описание	Тип	Обязательно
<code>rate</code>	Допустимая скорость запросов. Скорость указывается в запросах в секунду (r/s) или запросах в минуту (r/m).	string	Да
<code>key</code>	Ключ, к которому применяется ограничение скорости. Может содержать текст, переменные или их комбинацию. Переменные должны заключены в <code>\${}</code> . Например: <code>\$binary_remote_addr</code> . Допустимые переменные: <code>\$binary_remote_addr</code> , <code>\$request_uri</code> , <code>\$url</code> , <code>\$http_</code> , <code>\$args</code> , <code>\$arg_</code> , <code>\$cookie_</code> .	string	Да
<code>zoneSize</code>	Размер зоны разделяемой памяти. Допускаются только положительные значения. Допустимые суффиксы - <code>k</code> или <code>m</code> ; если суффикс не задан, предполагается <code>k</code> .	string	Да
<code>delay</code>	Указывает предел, при достижении которого избыточные запросы становятся отложенными. Если этот параметр не задан, задерживаются все избыточные запросы.	int	Нет
<code>noDelay</code>	Отключает задержку избыточных запросов при ограничении количества запросов. Имеет приоритет над <code>delay</code> , если заданы оба параметра.	bool	Нет
<code>burst</code>	Избыточные запросы задерживаются до тех пор, пока их количество не превысит размер <code>burst</code> , после чего запрос завершается с ошибкой.	int	Нет
<code>dryRun</code>	Включает режим сухого прогона. В этом режиме ограничение скорости фактически не применяется, но количество избыточных запросов учитывается, как обычно, в зоне разделяемой памяти.	bool	Нет
<code>logLevel</code>	Устанавливает желаемый уровень ведения журнала для случаев, когда сервер отказывается обрабатывать запросы из-за превышения скорости или задерживает обработку запросов. Допустимые значения: <code>info</code> , <code>notice</code> , <code>warn</code> или <code>error</code> . Значение по умолчанию - <code>error</code> .	string	Нет
<code>rejectCode</code>	Задаёт код состояния, возвращаемый в ответ на отклоненные запросы. Значение должно попадать в диапазон 400..599. Значение по умолчанию - 503.	int	Нет

Для каждой политики, на которую ссылается `VirtualServer` или его `VirtualServerRoute`, ANIC сгенерирует единую зону ограничения скорости, определенную директивой `limit_req_zone`. Если два ресурса `VirtualServer` ссылаются на одну и ту же политику, ANIC сгенерирует две разные зоны ограничения скорости, по одной на каждый `VirtualServer`.

Поведение слияния RateLimit

Ресурс `VirtualServer` или `VirtualServerRoute` может ссылаться на несколько политик ограничения скорости. Например, здесь мы ссылаемся на две политики:

```
policies:
- name: rate-limit-policy-one
- name: rate-limit-policy-two
```

Когда вы ссылаетесь на несколько политик ограничения скорости, ANIC настроит в Angie использование всех указанных ограничений скорости. Если определено несколько политик, каждая дополнительная политика наследует параметры `dryRun`, `LogLevel` и `rejectCode` из первой политики, на которую идет ссылка (`rate-limit-policy-one` в примере выше).

BasicAuth

Настраивает в Angie аутентификацию клиентских запросов при помощи базовой схемы аутентификации HTTP.

Например, следующая политика будет отклонять все запросы, которые не содержат действительную комбинацию имени пользователя и пароля в заголовке HTTP Authentication

```
basicAuth:
  secret: httpasswd-secret
  realm: "My API"
```

Примечание

Функция реализована с использованием модуля Angie http_auth_basic.

Поле	Описание	Тип	Обязательно
secret	Имя секрета Kubernetes, в котором хранится конфигурация Htpasswd. Он должен находиться в том же пространстве имен, что и ресурс Policy. Секрет должен иметь тип <code>angie.software/htpasswd</code> , а конфигурация должна храниться в секрете по ключу <code>htpasswd</code> ; в противном случае секрет будет отклонен как недействительный.	string	Да
realm	Область для базовой аутентификации.	string	Нет

Поведение слияния BasicAuth

Ресурс VirtualServer или VirtualServerRoute может ссылаться на несколько политик базовой аутентификации. При этом будет применяться только одна из них. Все последующие ссылки будут проигнорированы. Например, здесь мы ссылаемся на две политики:

```
policies:
- name: basic-auth-policy-one
- name: basic-auth-policy-two
```

В этом примере ANIC будет использовать конфигурацию из первой ссылки на политику "basic-auth-policy-one" и игнорирует "basic-auth-policy-two".

IngressMTLS

Политика IngressMTLS настраивает проверку сертификата клиента.

Например, следующая политика будет проверять сертификат клиента, используя сертификат Центра Сертификации, указанный в `ingress-mtls-secret`:

```
ingressMTLS:
  clientCertSecret: ingress-mtls-secret
  verifyClient: "on"
  verifyDepth: 1
```

Ниже приведен пример `ingress-mtls-secret` типа `angie.software/ca`


```
kind: Secret
metadata:
  name: ingress-mtls-secret
apiVersion: v1
type: angie.software/ca
data:
  ca.crt: <base64encoded-certificate>
```

У ресурса VirtualServer, который ссылается на политику IngressMTLS, должно быть следующие настройки:

- включена *терминация TLS*.
- ссылка на политику в *спецификации VirtualServer*. Не разрешается ссылаться на политику IngressMTLS в *маршруте* или *вложенном маршруте* VirtualServerRoute.

Если эти условия нарушены, Angie будет отправлять клиентам код состояния 500.

Вы можете передавать сведения о сертификате клиента, включая сам сертификат, серверам апстрима. Например:

```
action:
  proxy:
    upstream: webapp
    requestHeaders:
      set:
        - name: client-cert-subj-dn
          value: ${ssl_client_s_dn} # subject DN
        - name: client-cert
          value: ${ssl_client_escaped_cert} # клиентский сертификат в формате PEM
      ↪ (urlencoded)
```

Мы используем параметр `requestHeaders` в *Action.Proxy* для задания значений двух заголовков, которые Angie будет передавать серверам апстрима. См. список встроенных переменных, поддерживаемых модулем `http_ssl`, которые вы можете использовать для передачи сведений о сертификате клиента.

Примечание

Функция реализована с использованием модуля Angie `http_ssl`.

Использование списка отзыва сертификатов

Политика IngressMTLS поддерживает настройку списка CRL для политики. Это можно сделать одним из двух способов.

Примечание

Одновременно можно использовать только один из этих параметров конфигурации.

1. Добавление в тип секрета `angie.software/ca` поля `ca.crl`, которое содержит список отзыва сертификатов в кодировке base64. Пример YAML:

```
kind: Secret
metadata:
  name: ingress-mtls-secret
apiVersion: v1
```



```
type: angie.software/ca
data:
  ca.crt: <base64encoded-certificate>
  ca.crl: <base64encoded-crl>
```

- Добавление поля `crlFileName` с именем CRL-файла в спецификацию политики IngressMTLS.

Примечание

Этот параметр конфигурации следует использовать только при наличии CRL-файла размером более 1 МБ; в противном случае рекомендуется использовать для управления CRL тип секрета `angie.software/ca`.

Пример YAML:

```
apiVersion: k8s.angie.software/v1
kind: Policy
metadata:
  name: ingress-mtls-policy
spec:
  ingressMTLS:
    clientCertSecret: ingress-mtls-secret
    crlFileName: webapp.crl
    verifyClient: "on"
    verifyDepth: 1
```

Предупреждение

При настройке CRL с помощью поля `ingressMTLS.crlFileName` следует учитывать дополнительный контекст:

- ANIC ожидает, что CRL, в данном случае `webapp.crl`, будет находиться в каталоге `/etc/angie/secrets`. Для развертывания ANIC необходимо будет добавить точку подключения тома. Добавьте свой CRL в каталог `/etc/angie/secrets`.
- При обновлении содержимого списка CRL (например, был отозван новый сертификат) Angie необходимо перезагрузить, чтобы отразились последние изменения. В зависимости от вашей среды для этого может потребоваться обновить имя списка CRL и применить это обновление к политике `ingress-mtls.yaml`, чтобы Angie получил последнюю версию CRL.

Обратитесь к документации Kubernetes по [томам](#), чтобы найти наилучшую реализацию для вашей среды.

Поле	Описание	Тип	Обязательно
<code>clientCa</code>	Имя секрета Kubernetes, в котором хранится сертификат центра сертификации. Он должен находиться в том же пространстве имен, что и ресурс Policy. Секрет должен иметь тип <code>angie.software/ca</code> , а конфигурация должна храниться в секрете по ключу <code>ca.crt</code> ; в противном случае секрет будет отклонен как недействительный.	<code>string</code>	Да
<code>verifyClient</code>	Верификация для клиента. Допустимые значения: <code>"on"</code> , <code>"off"</code> , <code>"optional"</code> , <code>"optional_no_ca"</code> . Значение по умолчанию - <code>"on"</code> .	<code>string</code>	Нет
<code>verifyDepth</code>	Устанавливает глубину проверки в цепочке клиентских сертификатов. Значение по умолчанию равно 1.	<code>int</code>	Нет
<code>crlFile</code>	Имя файла списка отзыва сертификатов. ANIC будет искать этот файл в каталоге <code>/etc/angie/secrets</code>	<code>string</code>	Нет

Поведение слияния IngressMTLS

Ресурс `VirtualServer` может ссылаться только на одну политику `IngressMTLS`. Все последующие ссылки будут проигнорированы. Например, здесь мы ссылаемся на две политики:

```
policies:
- name: ingress-mtls-policy-one
- name: ingress-mtls-policy-two
```

В этом примере ANIC будет использовать конфигурацию из первой ссылки на политику `ingress-mtls-policy-one` и игнорирует `ingress-mtls-policy-two`.

EgressMTLS

`EgressMTLS` настраивает аутентификацию и проверку сертификатов для апстримов.

Например, следующая политика будет использовать `egress-mtls-secret` для аутентификации в приложении апстрима и `egress-trusted-ca-secret` для проверки сертификата приложения:

```
egressMTLS:
  tlsSecret: egress-mtls-secret
  trustedCertSecret: egress-trusted-ca-secret
  verifyServer: on
  verifyDepth: 2
```

Примечание

Функция реализована с использованием модуля Angie `http_proxy`.

Поле	Описание	Тип	Обязательно
<code>tlsSecret</code>	Имя секрета файла Kubernetes, в котором хранятся сертификат и ключ TLS. Он должен находиться в том же пространстве имен, что и ресурс Policy. Секрет должен иметь тип <code>kubernetes.io/tls</code> , сертификат - храниться в секрете под ключом <code>tls.crt</code> , а ключ - как <code>tls.key</code> ; в противном случае секрет будет отклонен как недействительный.	<code>string</code>	Нет
<code>trustedCa</code>	Имя секрета Kubernetes, в котором хранится сертификат центра сертификации. Он должен находиться в том же пространстве имен, что и ресурс Policy. Секрет должен иметь тип <code>angie.software/ca</code> , а конфигурация должна храниться в секрете по ключу <code>ca.crt</code> ; в противном случае секрет будет отклонен как недействительный.	<code>string</code>	Нет
<code>verifySecret</code>	Включает проверку сертификата HTTPS-сервера апстрима.	<code>bool</code>	Нет
<code>verifyDepth</code>	Устанавливает глубину проверки в цепочке сертификатов проксируемого HTTPS-сервера. Значение по умолчанию равно 1.	<code>int</code>	Нет
<code>sessionIdleTimeout</code>	Позволяет повторно использовать SSL-сеансы к апстримам. Значение по умолчанию равно <code>true</code> .	<code>bool</code>	Нет
<code>serverName</code>	Позволяет передавать имя сервера через расширение SNI.	<code>bool</code>	Нет
<code>sslName</code>	Позволяет переопределить имя сервера, используемое для проверки сертификата HTTPS-сервера апстрима.	<code>string</code>	Нет
<code>ciphers</code>	Указывает разрешенные шифры для запросов к HTTPS-серверу апстрима. Значение по умолчанию - <code>DEFAULT</code> .	<code>string</code>	Нет
<code>protocol</code>	Задаёт протоколы для запросов к HTTPS-серверу апстрима. Значение по умолчанию - <code>TLSv1, TLSv1.1, TLSv1.2</code> .	<code>string</code>	Нет

Поведение слияния EgressMTLS

Ресурс `VirtualServer` или `VirtualServerRoute` может ссылаться на несколько политик `EgressMTLS`. При этом будет применяться только одна из них. Все последующие ссылки будут проигнорированы. Например, здесь мы ссылаемся на две политики:

```
policies:
- name: egress-mtls-policy-one
- name: egress-mtls-policy-two
```

В этом примере ANIC будет использовать конфигурацию из первой ссылки на политику `egress-mtls-policy-one` и игнорирует `egress-mtls-policy-two`.

OIDC

OIDC (OpenID Connect) обеспечивает удобную аутентификацию пользователей через внешнего провайдера, используя безопасные токены для управления доступом в системе.

Примечание

Эта функция отключена по умолчанию. Чтобы ее включить, задайте аргумент командной строки `enable-oidc=true`.

Политика OIDC настраивает ANIC как клиент (relying party) для аутентификации через OpenID Connect:

```
policies:
- name: oidc-policy
```


Например, следующая конфигурация использует `clientID myclient` и `clientSecret oidc-secret` для аутентификации через провайдера OpenID Connect `https://idp.example.com`:

```
oidc:
  clientID: myclient
  clientSecret: oidc-secret
  authEndpoint: https://idp.example.com/openid-connect/auth
  jwksURI: https://idp.example.com/openid-connect/certs
  tokenEndpoint: https://idp.example.com/openid-connect/token
  scope: openid+profile+email
  accessTokenEnable: true
```

Обязательные условия:

- В спецификации `VirtualServer` задайте обязательные переменные `$jwt_claim_iat`, `$jwt_claim_iss`, `$jwt_claim_sub`, `$jwt_claim_aud` в *maps*. Переменные обеспечивают валидацию токенов в процессе аутентификации OIDC.
- Добавьте секрет с ключом клиента. Ключ должен быть закодирован в Base64:

```
apiVersion: v1
kind: Secret
metadata:
  name: oidc-secret
type: angie.software/oidc
data:
  client-secret: <client_secret>
```

Пошаговые инструкции по настройке см. в статье [Настройка OIDC](#).

Поле	Описание	Тип	Обязательно
<code>clientID</code>	Идентификатор клиента, предоставленный провайдером OIDC. Идентифицирует приложение, которое обращается за авторизацией.	string	да
<code>clientSecret</code>	Имя секрета, где хранится ключ клиента. Должен находиться в том же пространстве имен, что и <code>Policy</code> .	string	да
<code>authEndpoint</code>	URL конечной точки авторизации, предоставленной провайдером OIDC. Это адрес, по которому будут отправляться запросы для аутентификации пользователя.	string	да
<code>jwksURI</code>	URI, по которому провайдер OIDC предоставляет сертификаты (JSON Web Key Set) для проверки выданных сервером JWT-токенов (JSON Web Token).	string	да
<code>tokenEndpoint</code>	URL для получения токенов аутентификации и обновления от провайдера OIDC.	string	да
<code>scope</code>	Список OIDC-областей, которые нужно запросить у провайдера. Значение по умолчанию - <code>openid</code> . Это значение является обязательным. Вы также можете добавлять другие области, используя знак +, например: <code>openid+profile+email</code> .	string	да
<code>accessTokenEnable</code>	Включает использование Bearer-токена для авторизации доступа к защищенным ресурсам на проксируемом сервере.	bool	нет

Примечание

В ресурсах `VirtualServer` можно использовать только одну политику OIDC, причем ее можно применять к разным маршрутам `VirtualServer`. Например, если в конфигурации есть несколько

маршрутов для обработки запросов, все они могут использовать одну и ту же политику OIDC для аутентификации пользователей.

Поведение слияния OIDC

Ресурс VirtualServer может ссылаться на несколько политик OIDC. При этом будет применяться только одна из них. Все последующие ссылки будут проигнорированы. Например, здесь мы ссылаемся на две политики:

```
policies:
- name: oidc-policy-one
- name: oidc-policy-two
```

В этом примере ANIC будет использовать конфигурацию из первой ссылки на политику oidc-policy-one и игнорирует oidc-policy-two.

JWT

Политика JWT настраивает в Angie аутентификацию запросов клиентов с использованием JSON Web Tokens.

```
policies:
- name: jwt-policy
```

Примечание

Эта функция отключена по умолчанию. Чтобы ее включить, задайте аргумент командной строки `-enable-jwt=true`.

Следующая политика будет отклонять все запросы, которые не содержат действительный JWT в токене заголовка HTTP:

```
jwt:
  realm: MyProductAPI
  secret: jwk-secret
  token: $http_token
```

Обязательные условия:

- Добавьте секрет с ключом клиента. Ключ должен быть закодирован в Base64:

```
apiVersion: v1
kind: Secret
metadata:
  name: jwk-secret
type: angie.software/jwk
data:
  jwk: <client_secret>
```


Поле	Описание	Тип	Обязательно
<code>realm</code>	Область, которую клиент увидит при запросе аутентификации, например, строка, описывающая защищенный ресурс.	<code>string</code>	да
<code>secret</code>	Имя секрета Kubernetes, который хранит JSON Web Key (JWK), используемый для проверки подписей токенов JWT. Angie будет использовать ключи из этого секрета для валидации подписей JWT и проверки их подлинности. Секрет должен находиться в том же пространстве имен, что и Policy.	<code>string</code>	да
<code>token</code>	Указывает, откуда нужно извлечь JWT. Например, переменная <code>\$http_token</code> ссылается на значение заголовка HTTP <code>token</code> , который будет передан клиентом.	<code>string</code>	нет

Поведение слияния JWT

Ресурс `VirtualServer` может ссылаться на несколько политик JWT. При этом будет применяться только одна из них. Все последующие ссылки будут проигнорированы. Например, здесь мы ссылемся на две политики:

```
policies:
- name: jwt-policy-one
- name: jwt-policy-two
```

В этом примере ANIC будет использовать конфигурацию из первой ссылки на политику `jwt-policy-one` и игнорирует `jwt-policy-two`.

Применение политик

Политики можно применять как к ресурсам `VirtualServer`, так и к `VirtualServerRoute`. Например:

```
- VirtualServer:
  apiVersion: k8s.angie.software/v1
  kind: VirtualServer
  metadata:
    name: cafe
    namespace: cafe
  spec:
    host: cafe.example.com
    tls:
      secret: cafe-secret
    policies: # spec policies
      - name: policy1
    upstreams:
      - name: coffee
        service: coffee-svc
        port: 80
    routes:
      - path: /tea
        policies: # route policies
          - name: policy2
            namespace: cafe
            route: tea/tea
      - path: /coffee
        policies: # route policies
          - name: policy3
```



```
namespace: cafe
action:
pass: coffee
```

В случае VirtualServer политику можно применить:

- для всех маршрутов (политики спецификации)
- к определенному маршруту (политики маршрутов)

Политики маршрутов имеют приоритет над политиками спецификации *того же типа*. Если в примере выше тип политик `policy-1` и `policy-3` - `AccessControl`, то для запросов к `cafe.example.com/coffee` Angie применит `policy-3`.

Переопределение обеспечивается Angie: политики спецификации реализуются в контексте конфигурации `server`, а политики маршрутов реализуются в контексте `location`. В результате приоритет в рамках одного типа имеют политики маршрутов.

- Recypc VirtualServerRoute, на который ссылается указанный выше VirtualServer:

```
apiVersion: k8s.angie.software/v1
kind: VirtualServerRoute
metadata:
  name: tea
  namespace: tea
spec:
  host: cafe.example.com
  upstreams:
    - name: tea
      service: tea-svc
      port: 80
  subroutes: # subroute policies
    - path: /tea
      policies:
        - name: policy4
          namespace: tea
      action:
        pass: tea
```

В VirtualServerRoute можно применить политику к вложенному маршруту (политики вложенных маршрутов).

Политики вложенных маршрутов имеют приоритет над политиками спецификации того же типа. В приведенном выше примере, если тип политик `policy-1` (в VirtualServer) и `policy-4` - `AccessControl`, то для запросов к `cafe.example.com/tea` Angie будет применять `policy-4`. Как и в случае с VirtualServer, переопределение обеспечивается средствами Angie.

Политики вложенных маршрутов всегда имеют приоритет над политиками маршрутов независимо от типа. Например, политика `policy-2` в маршруте VirtualServer будет проигнорирована на вложенном маршруте `/tea`, поскольку у того есть свои собственные политики (в нашем случае это только `policy4`). Если бы у вложенного маршрута не было политик, то была бы применена `policy-2`. Это переопределение выполняет ANIC - контекст `location` для вложенного маршрута будет содержать либо политики маршрута, либо политики вложенного маршрута, но не то и другое вместе.

Недопустимые политики

Angie будет рассматривать политику как недействительную, если выполняется одно из следующих условий:

- Политика не проходит *всестороннюю валидацию*.
- Политика отсутствует в кластере.
- Политика не соответствует требованиям, предъявляемым к ее конкретному типу. Например, политика `ingressMTLS` требует, чтобы в `VirtualServer` была включена терминация TLS.

В случае недопустимой политики Angie возвращает код состояния 500 для клиентских запросов со следующими правилами:

- Если на политику ссылается маршрут `VirtualServer` или вложенный маршрут `VirtualServerRoute`, Angie будет возвращать код состояния 500 для запросов к URI такого маршрута.
- Если ссылка на политику задана в спецификации `VirtualServer`, Angie будет возвращать код состояния 500 для запросов ко всем URI этого `VirtualServer`.

Если политика недействительна, `VirtualServer` или `VirtualServerRoute` будет иметь *статус с предупреждением* о состоянии и сообщением, объясняющим, почему политика не была признана недействительной.

Валидация

Для ресурса `Policy` доступны два типа валидации:

- *Структурная валидация* с помощью `kubectl` и сервера Kubernetes API.
- *Всесторонняя валидация* с помощью ANIC.

Структурная валидация

Пользовательское определение ресурса для `Policy` включает структурную схему OpenAPI, которая описывает тип каждого поля ресурса.

Если вы попытаетесь создать (или обновить) ресурс, который нарушает структурную схему (например, использует строковое значение вместо массива строк в поле `allow`), то `kubectl` и сервер Kubernetes API отклонят ресурс.

- Пример проверки `kubectl`:

```
kubectl apply -f access-control-policy-allow.yaml

error: error validating "access-control-policy-allow.yaml": error validating
↳data: ValidationError(Policy.spec.accessControl.allow): invalid type for
↳software.angie.k8s.v1.Policy.spec.accessControl.allow: got "string", expected
↳"array"; if you choose to ignore these errors, turn validation off with --
↳validate=false
```

- Пример проверки сервера Kubernetes API:

```
kubectl apply -f access-control-policy-allow.yaml --validate=false

The Policy "webapp-policy" is invalid: spec.accessControl.allow: Invalid value:
↳"string": spec.accessControl.allow in body must be of type array: "string"
```

Если ресурс прошел структурную валидацию, выполняется всесторонняя валидация ANIC.

Всесторонняя валидация

ANIC проверяет поля ресурса Policy. Если ресурс недопустим, ANIC отклонит его. Ресурс останется в кластере, но ANIC будет игнорировать его.

Можно использовать `kubectl`, чтобы проверить, успешно ли ANIC применил конфигурацию Policy. Для политики `mypolicy` мы можем запустить:

```
kubectl describe pol mypolicy
. . .
Events:
  Type          Reason          Age   From                      Message
  ----
Normal    AddedOrUpdated   11s   angie-ingress-controller  Policy default/mypolicy was
↪added or updated
```

Обратите внимание, что раздел "События" (Events) включает событие Normal с причиной AddedOrUpdated, которое информирует нас о том, что конфигурация была успешно применена.

Если вы создадите недопустимый ресурс, ANIC отклонит его и выдаст событие Rejected. Например, если вы создадите политику `mypolicy` с недопустимым IP-адресом `10.0.0.` в поле `allow`, то вы получите:

```
kubectl describe policy mypolicy
. . .
Events:
  Type          Reason          Age   From                      Message
  ----
Warning    Rejected         7s   angie-ingress-controller  Policy default/mypolicy is invalid
↪and was rejected: spec.accessControl.allow[0]: Invalid value: "10.0.0.": must be a
↪CIDR or IP
```

Обратите внимание, что раздел "События" (Events) включает предупреждающее событие с указанием причины отклонения.

Кроме того, эта информация также доступна в поле `status` ресурса Policy. Обратите внимание на раздел "Статус" (Status) политики:

```
kubectl describe pol mypolicy
. . .
Status:
  Message: Policy default/mypolicy is invalid and was rejected: spec.accessControl.
↪allow[0]: Invalid value: "10.0.0.": must be a CIDR or IP
  Reason:   Rejected
  State:    Invalid
```

Примечание

Если вы сделаете существующий ресурс недействительным, ANIC отклонит его.

2.6 TransportServer

og:description

Ресурс TransportServer для настройки балансировки нагрузки TCP, UDP и TLS Passthrough в ANIC

Ресурс TransportServer позволяет настраивать балансировку нагрузки по протоколам TCP, UDP и TLS Passthrough. Он реализован как [пользовательский ресурс](#).

Это справочная документация по ресурсу TransportServer.

2.6.1 Предварительные требования

- Для TCP и UDP ресурс TransportServer должен использоваться совместно с ресурсом GlobalConfiguration, который должен быть создан отдельно.
- Для TLS Passthrough обязательно включите параметр командной строки `-enable-tls-passthrough` в ANIC.

2.6.2 Спецификация TransportServer

Ресурс TransportServer определяет конфигурацию балансировки нагрузки для трафика TCP, UDP или TLS Passthrough. Ниже приведено несколько примеров:

- Балансировка нагрузки TCP:

```
apiVersion: k8s.angie.software/v1alpha1
kind: TransportServer
metadata:
  name: dns-tcp
spec:
  listener:
    name: dns-tcp
    protocol: TCP
  tls:
    secret: cafe-secret
  upstreams:
  - name: dns-app
    service: dns-service
    port: 5353
    action:
      pass: dns-app
```

- Балансировка нагрузки UDP:

```
apiVersion: k8s.angie.software/v1alpha1
kind: TransportServer
metadata:
  name: dns-udp
spec:
  listener:
    name: dns-udp
    protocol: UDP
  upstreams:
  - name: dns-app
    service: dns-service
    port: 5353
    upstreamParameters:
```



```
udpRequests: 1
udpResponses: 1
action:
pass: dns-app
```

- Балансировка нагрузки TLS Passthrough:

```
apiVersion: k8s.angie.software/v1alpha1
kind: TransportServer
metadata:
name: secure-app
spec:
listener:
  name: tls-passthrough
  protocol: TLS_PASSTHROUGH
host: app.example.com
upstreams:
- name: secure-app
  service: secure-app
  port: 8443
  action:
  pass: secure-app
```

Поле	Описание	Тип	Обязательно
<code>listener</code>	Прослушиватель, через который Angie будет принимать входящие соединения и датаграммы.	<i>Listener</i>	Да
<code>host</code>	Хост (доменное имя) сервера. Это должен быть допустимый поддомен, как определено в RFC 1123, например <code>my-app</code> или <code>hello.example.com</code> . Домены с подстановочными знаками, такие как <code>*.example.com</code> , не допускаются. Требуется для балансировки нагрузки TLS Passthrough.	<code>string</code>	Нет
<code>tls</code>	Конфигурация терминирования TLS. Не поддерживается для балансировки нагрузки TLS Passthrough.	<i>TLS</i>	Нет
<code>upstream</code>	Список апстримов.	<i>upstream</i> /	Да
<code>upstream</code>	Параметры апстрима.	<i>Upstream</i>	Нет
<code>action</code>	Действие, выполняемое для клиентского соединения или датаграммы.	<i>Action</i>	Да
<code>ingress</code>	Указывает, какой экземпляр ANIC должен обрабатывать ресурс TransportServer.	<code>string</code>	Нет
<code>streamServer</code>	Задаёт пользовательский фрагмент в контексте <code>stream</code> .	<code>string</code>	Нет
<code>server</code>	Задаёт пользовательский фрагмент в контексте <code>server</code> .	<code>string</code>	Нет

Listener

Ссылается на прослушиватель, через который Angie будет принимать входящий трафик к TransportServer. Для TCP и UDP прослушиватель должен быть определен в ресурсе GlobalConfiguration. При ссылке на прослушиватель должны совпадать как имя, так и протокол. Для TLS Passthrough используйте встроенный прослушиватель с именем `tls-passthrough` и протоколом `TLS_PASSTHROUGH`.

Пример:

```
listener:
  name: dns-udp
```



```
protocol: UDP
```

Поле	Описание	Тип	Обязательно
<code>name</code>	Имя прослушивателя.	<code>string</code>	Да
<code>protocol</code>	Протокол прослушивателя.	<code>string</code>	Да

TLS

Поле `tls` определяет конфигурацию TLS для `TransportServer`. Обратите внимание, что текущая реализация поддерживает терминацию TLS на нескольких портах, где каждому приложению принадлежит выделенный порт. При этом ANIC терминирует TLS-соединения на каждом порту, где каждое приложение использует свой собственный сертификат или ключ, и направляет соединения соответствующему приложению (сервису) на основе этого входящего порта (т. е. любое TLS-соединение независимо от настроек SNI на порту будет перенаправлено в приложение, соответствующее этому порту).

Пример конфигурации показан ниже:

```
secret: cafe-secret
```

Поле	Описание	Тип	Обязательно
<code>secret</code>	Имя секрета с сертификатом TLS и ключом. Секрет должен принадлежать тому же пространству имен, что и транспортный сервер. Секрет должен иметь тип <code>kubernetes.io/tls</code> и содержать ключи с именами <code>tls.crt</code> и <code>tls.key</code> , содержащие сертификат и закрытый ключ, как описано здесь .	<code>string</code>	Нет

Upstream

Определяет конечное место назначения для `TransportServer`. Например:

```
name: secure-app
service: secure-app
port: 8443
maxFails: 3
maxConns: 100
failTimeout: 30s
loadBalancingMethod: least_conn
```


Поле	Описание	Тип	Обязательно
<code>name</code>	Имя апстрима. Это должна быть допустимая метка DNS, как определено в RFC 1035. Например, допустимы значения <code>hello</code> и <code>upstream-123</code> . Имя должно быть уникальным среди всех апстримов ресурса.	<code>string</code>	Да
<code>service</code>	Название сервиса. Сервис должен принадлежать к тому же пространству имен, что и ресурс. Если сервиса не существует, Angie предположит, что у него нет конечных точек, и будет закрывать клиентские соединения и игнорировать датаграммы.	<code>string</code>	Да
<code>port</code>	Порт службы. Если у сервиса этот порт не задан, Angie предположит, что у него нет конечных точек, и будет закрывать клиентские соединения и игнорировать датаграммы. Значение должно находиться в диапазоне 1..65535.	<code>int</code>	Да
<code>maxFails</code>	Задаёт число неудачных попыток установить связь с сервером, которые должны произойти в течение времени, заданного параметром <code>failTimeout</code> , чтобы сервер считался недоступным. Значение по умолчанию: 1.	<code>int</code>	Нет
<code>maxConn</code>	Задаёт максимальное число <code><server></code> подключений к проксируемому серверу. Значение по умолчанию равно нулю, что означает отсутствие ограничений. Значение по умолчанию равно 0.	<code>int</code>	Нет
<code>failTime</code>	Задаёт время, в течение которого должно произойти указанное количество неудачных попыток установить связь с сервером, чтобы считать сервер недоступным, и период времени, в течение которого сервер будет считаться недоступным. Значение по умолчанию равно 10 секундам.	<code>string</code>	Нет
<code>loadBal</code>	Метод балансировки нагрузки между серверами апстрима. По умолчанию соединения распределяются между серверами по методу взвешенной циклической балансировки. Доступные методы и подробности смотрите в разделе Апстрим.	<code>string</code>	Нет

UpstreamParameters

Различные параметры апстрима:

```
upstreamParameters:
  udpRequests: 1
  udpResponses: 1
  connectTimeout: 60s
  nextUpstream: true
  nextUpstreamTimeout: 50s
  nextUpstreamTries: 1
```


Поле	Описание	Тип	Обязательно
<code>udpRequests</code>	Количество датаграмм, после получения которых следующая датаграмма от того же клиента запускает новый сеанс. См. директиву <code>s_proxu_requests</code> . Значение по умолчанию равно 0.	<code>int</code>	Нет
<code>udpResponses</code>	Количество датаграмм, ожидаемых от проксируемого сервера в ответ на клиентскую датаграмму. См. директиву <code>s_proxu_responses</code> . По умолчанию количество датаграмм не ограничено.	<code>int</code>	Нет
<code>connectTimeout</code>	Тайм-аут установки соединения с проксируемым сервером. См. директиву <code>s_proxu_connect_timeout</code> . Значение по умолчанию - 60 секунд.	<code>string</code>	Нет
<code>nextUpstream</code>	Если соединение с проксируемым сервером установить не удастся, определяет, будет ли клиентское соединение передано на следующий сервер. См. директиву <code>s_proxu_next_upstream</code> . Значение по умолчанию равно <code>true</code> .	<code>bool</code>	Нет
<code>nextUpstreamTries</code>	Количество попыток до передачи соединения к следующему серверу. См. директиву <code>s_proxu_next_upstream_tries</code> . Значение по умолчанию равно 0.	<code>int</code>	Нет
<code>nextUpstreamTimeout</code>	Время, отведенное для передачи соединения к следующему серверу. См. директиву <code>s_proxu_next_upstream_timeout</code> . Значение по умолчанию - 0.	<code>string</code>	Нет

SessionParameters

Различные параметры для TCP-соединений и UDP-сеансов.

```
sessionParameters:
  timeout: 50s
```

Поле	Описание	Тип	Обязательно
<code>timeout</code>	Тайм-аут между двумя последовательными операциями чтения или записи в соединениях с клиентом или проксируемым сервером. См. директиву <code>s_proxu_timeout</code> . Значение по умолчанию равно 10m.	<code>string</code>	Нет

Action

Действие, которое необходимо выполнить для клиентского соединения или датаграммы.

В приведенном ниже примере клиентские подключения и датаграммы передаются на апстрим в `dns-app`:

```
action:
  pass: dns-app
```

Поле	Описание	Тип	Обязательно
<code>pass</code>	Передаёт соединения и датаграммы апстриму. Апстрим с таким именем должен быть определен в ресурсе.	<code>string</code>	Да

2.6.3 Использование TransportServer

Для работы с ресурсами TransportServer можно использовать обычные команды `kubectl`, аналогично ресурсам Ingress.

Например, следующая команда создает ресурс TransportServer, определенный в `transport-server-passthrough.yaml`, с именем `secure-app`:

```
kubectl apply -f transport-server-passthrough.yaml

transportserver.k8s.angie.software/secure-app created
```

Вы можете получить ресурс, выполнив:

```
kubectl get transportserver secure-app

NAME          AGE
secure-app    46sm
```

В `kubectl get` и подобных командах также можно использовать короткое имя `ts` вместо `transportserver`.

Использование фрагментов

Фрагменты позволяют вставлять элементы конфигурации Angie в различные контексты конфигурации Angie. В приведенном ниже примере мы используем фрагменты для настройки контроля доступа на TransportServer:

```
apiVersion: k8s.angie.software/v1alpha1
kind: TransportServer
metadata:
  name: cafe
spec:
  host: cafe.example.com
  serverSnippets: |
    deny 192.168.1.1;
    allow 192.168.1.0/24;
  upstreams:
  - name: tea
    service: tea-svc
    port: 80
```

Фрагменты также можно указать для потока. В приведенном ниже примере мы используем фрагменты для ограничения количества подключений:

```
apiVersion: k8s.angie.software/v1alpha1
kind: TransportServer
metadata:
  name: cafe
spec:
  host: cafe.example.com
  streamSnippets: limit_conn_zone $binary_remote_addr zone=addr:10m;
  serverSnippets: limit_conn addr 1;
  upstreams:
  - name: tea
    service: tea-svc
    port: 80
```


Фрагменты предназначены для продвинутых пользователей Angie, которым требуется больше контроля над генерируемой конфигурацией Angie.

Однако из-за недостатков, описанных ниже, фрагменты по умолчанию отключены. Чтобы использовать фрагменты, задайте аргумент командной строки `enable-snippets`.

Недостатки использования фрагментов:

- *Сложность.* Чтобы использовать фрагменты, требуется:
 - Понимать примитивы конфигурации Angie и реализовать правильную конфигурацию Angie.
 - Понимать, как ANIC генерирует конфигурацию Angie, чтобы фрагмент не мешал другим функциям конфигурации.
- *Сниженная надежность.* Неправильный фрагмент делает конфигурацию Angie недействительной, что приведет к ошибке при перезагрузке. Это мешает применить какие-либо обновления конфигурации, включая обновления для другого ресурса TransportServer, пока фрагмент не будет исправлен.
- *Последствия для безопасности.* Фрагменты предоставляют доступ к примитивам конфигурации Angie, и эти примитивы не проверяются самим ANIC.

i Примечание

Пока конфигурация Angie содержит недопустимый фрагмент, Angie будет продолжать работать с последней допустимой конфигурацией.

i Примечание

Чтобы настроить фрагменты в контексте `stream`, используйте ключ `stream-snippets` ConfigMap.

Валидация

Для ресурса TransportServer доступны два типа валидации:

- *Структурная валидация* с помощью `kubectl` и сервера Kubernetes API.
- *Всесторонняя валидация* с помощью ANIC.

Структурная валидация

Пользовательское определение ресурса для TransportServer включает структурную схему OpenAPI, которая описывает тип каждого поля ресурса.

Если вы попытаетесь создать (или обновить) ресурс с нарушением структурной схемы (например, используете строковое значение для поля порта апстрима), сервер `kubectl` и Kubernetes API отклонят такой ресурс:

- Пример проверки `kubectl`:

```
kubectl apply -f transport-server-passthrough.yaml

error: error validating "transport-server-passthrough.yaml": error validating
→data: ValidationError(TransportServer.spec.upstreams[0].port): invalid type
→for software.angie.k8s.v1alpha1.TransportServer.spec.upstreams.port: got
```



```
↪ "string", expected "integer"; if you choose to ignore these errors, turn
↪ validation off with --validate=false
```

- Пример проверки сервера Kubernetes API:

```
kubectl apply -f transport-server-passthrough.yaml --validate=false

The TransportServer "secure-app" is invalid: []: Invalid value:
↪ map[string]interface {}{ ... }: validation failure list:
spec.upstreams.port in body must be of type integer: "string"
```

Если ресурс не отклонен (то есть не нарушает структурную схему), ANIC проверит его дополнительно.

Всесторонняя валидация

ANIC проверяет поля ресурса TransportServer. Если ресурс недействителен, ANIC отклонит его: ресурс продолжит существовать в кластере, но ANIC будет его игнорировать.

Вы можете проверить, успешно ли ANIC применил конфигурацию TransportServer. Для примера TransportServer **secure-app** мы можем запустить:

```
kubectl describe ts secure-app

. . .
Events:
  Type          Reason          Age    From                      Message
  ----
Normal         AddedOrUpdated  3s     angie-ingress-controller  Configuration for default/
↪ secure-app was added or updated
```

Обратите внимание, что раздел **Events** (События) включает событие **Normal** с причиной **AddedOrUpdated**, которое информирует нас о том, что конфигурация была успешно применена.

Если вы создадите недопустимый ресурс, ANIC отклонит его и выдаст событие **Rejected**. Например, если вы создадите TransportServer **secure-app** с действием **pass**, которое ссылается на несуществующий апстрим, вы получите:

```
kubectl describe ts secure-app

. . .
Events:
  Type          Reason          Age    From                      Message
  ----
Warning         Rejected         2s     angie-ingress-controller  TransportServer default/secure-app
↪ is invalid and was rejected: spec.action.pass: Not found: "some-app"
```

Обратите внимание, что раздел событий включает событие **Warning** с причиной **Rejected**.

Примечание

Если вы внесете ошибку в уже существующий ресурс, ANIC отклонит его и удалит соответствующую конфигурацию из Angie.

Настройка с помощью ConfigMap

Ключи ConfigMap (за исключением `stream-snippets`, `stream-log-format`, `resolver-addresses`, `resolver-ipv6`, `resolver-valid` и `resolver-timeout`) не влияют на ресурсы `TransportServer`.

2.7 VirtualServer, VirtualServerRoute

Ресурсы `VirtualServer` и `VirtualServerRoute` реализуют сценарии использования, не поддерживаемые ресурсом `Ingress`, такие как разделение трафика и продвинутая маршрутизация на основе содержимого. Они реализованы как [пользовательские ресурсы](#).

Это справочная документация по обоим ресурсам.

2.7.1 Спецификация VirtualServer

Ресурс `VirtualServer` определяет конфигурацию балансировки нагрузки для доменного имени, например `example.com`. Ниже приведен пример такой конфигурации:

```
apiVersion: k8s.angie.software/v1
kind: VirtualServer
metadata:
  name: cafe
spec:
  host: cafe.example.com
  tls:
    secret: cafe-secret
  staticLocations:
    - type: root
      urlPath: /americano
      dirPath: /latte
  gunzip: on
  upstreams:
    - name: tea
      service: tea-svc
      port: 80
    - name: coffee
      service: coffee-svc
      port: 80
  routes:
    authRequest: /auth/p
    authRequestSets:
      - key: foo
        value: bar
    matches:
      - conditions:
          - variable: $request_method
            value: POST
        action:
          pass: tea-post
      - path: /coffee
        action:
          pass: coffee
      - path: ~ ^/decaf/.\*\.\.jpg$
        action:
```



```

    pass: coffee
- path: = /green/tea
  action:
    pass: tea
activeHealthProbes:
- name: activename1
  upstream: tea
  uri: uri
  port: 80
  interval: 3s
  isEssential: true
  isPersistent: true
  maxBody: 10m
  fails: 4
  passes: 5
  mode: onfail
maps:
- variable: $jwt_claim_iat
  source: $oidc_client
  parameters:
    - value: 'myclient'
      result: '80'
- variable: $jwt_claim_iss
  source: $oidc_client
  parameters:
    - value: 'myclient'
      result: 'PROVIDER_URL'
- variable: $jwt_claim_sub
  source: $oidc_client
  parameters:
    - value: 'myclient'
      result: 'myclient'
- variable: $jwt_claim_aud
  source: $oidc_client
  parameters:
    - value: 'myclient'
      result: 'myclient'
authRequestLocations:
- path: /auth/path
  proxyPass:
    upstreamName: "tea"
  proxyPassHeaders:
    - key: Content-Length
      value: "100"

```


Поле	Описание	Тип	Обязательно
<code>host</code>	Хост (доменное имя) сервера. Это должен быть допустимый под-домен, как определено в RFC 1123, например <code>my-app</code> или <code>hello.example.com</code> . При использовании домена с подстановочным знаком, например <code>*.example.com</code> , домен должен быть заключен в двойные кавычки. Значение <code>host</code> должно быть уникальным среди всех ресурсов Ingress и VirtualServer.	<code>string</code>	Да
<code>tls</code>	Конфигурация терминирования TLS.	<code>tls</code>	Нет
<code>staticLocations</code>	Список каталогов для раздачи статических файлов.	<code>staticLocations</code>	Нет
<code>gunzip</code>	Включает или отключает распаковку архивированных ответов для клиентов. Допустимые пары значений: "on" и "off", "true" и "false" или "yes" и "no". Если значение <code>gunzip</code> не установлено, то по умолчанию оно равно <code>off</code> .	<code>boolean</code>	Нет
<code>externalDNS</code>	Конфигурация ExternalDNS для VirtualServer.	<code>externalDNS</code>	Нет
<code>dos</code>	Ссылка на DosProtectedResource; установка этого параметра включает защиту VirtualServer от DOS-атак.	<code>string</code>	Нет
<code>policies</code>	Список политик.	<code>policyNames</code>	Нет
<code>upstreams</code>	Список апстримов.	<code>upstreams</code>	Нет
<code>routes</code>	Список маршрутов.	<code>routes</code>	Нет
<code>activeHealthChecks</code>	Список активных проверок работоспособности.	<code>activeHealthChecks</code>	Нет
<code>maps</code>	Список переменных, необходимых для валидации токенов в процессе аутентификации <code>OIDC</code> .	<code>maps</code>	Нет
<code>ingressClass</code>	Указывает, какой экземпляр ANIC должен обрабатывать ресурс VirtualServer.	<code>string</code>	Нет
<code>internal</code>	Указывает, является ли ресурс VirtualServer внутренним маршрутом.	<code>boolean</code>	Нет
<code>http-snippet</code>	Задаёт пользовательский фрагмент в контексте <code>http</code> .	<code>string</code>	Нет
<code>server-snippet</code>	Задаёт пользовательский фрагмент в контексте <code>server</code> . Имеет приоритет над ключом <code>ConfigMap server-snippets</code> .	<code>string</code>	Нет
<code>authRequest</code>	Задаёт список точек авторизации клиента при настройке <code>authRequest</code> .	<code>authRequests</code>	Нет

VirtualServer.TLS

Поле `tls` определяет конфигурацию TLS для ресурса VirtualServer. Например:

```
secret: cafe-secret
redirect:
  enable: true
ssl_session_timeout: 1h
ssl_session_cache: shared:SSL:10m
ssl_session_tickets: on
ssl_stapling: on
ssl_stapling_verify: on
```


Поле	Описание	Тип	Обязательно
<code>secret</code>	Имя секрета с сертификатом TLS и ключом. Секрет должен принадлежать тому же пространству имен, что и VirtualServer. Секрет должен иметь тип <code>kubernetes.io/tls</code> и содержать ключи с именами <code>tls.crt</code> и <code>tls.key</code> , содержащие сертификат и закрытый ключ, как описано здесь . Если секрет не существует или недействителен, Angie прервет любую попытку установить TLS-соединение с хостом VirtualServer. Если секрет не указан, но настроен секрет TLS с подстановочным знаком, Angie будет использовать секрет со знаком для терминации TLS.	<code>string</code>	Нет
<code>redirect</code>	Конфигурация перенаправления TLS для VirtualServer.	<code>tls.redirect</code>	Нет
<code>cert-manager</code>	Конфигурация TLS cert-manager для VirtualServer.	<code>tls.cert-manager</code>	Нет
<code>ssl_session_timeout</code>	Задаёт время, в течение которого клиент может повторно использовать параметры сессии. См. также директиву <code>ssl_session_timeout</code> в документации Angie. Значение по умолчанию <code>10m</code> .	<code>string</code>	Нет
<code>ssl_session_cache</code>	Задаёт тип и размеры кэшей для хранения параметров сессий. См. также директиву <code>ssl_session_cache</code> в документации Angie. Значение по умолчанию <code>shared:SSL:10m</code> .	<code>string</code>	Нет
<code>ssl_session_tickets</code>	Разрешает или запрещает возобновление сессий при помощи TLS session tickets. См. также директиву <code>ssl_session_tickets</code> в документации Angie. Значение по умолчанию <code>off</code> .	<code>string</code>	Нет
<code>ssl_stapling</code>	Разрешает или запрещает прикрепление OCSP-ответов сервером. См. также директиву <code>ssl_stapling</code> в документации Angie. Значение по умолчанию <code>on</code> .	<code>string</code>	Нет
<code>ssl_stapling_verify</code>	Разрешает или запрещает проверку сервером ответов OCSP. См. также директиву <code>ssl_stapling_verify</code> в документации Angie. Значение по умолчанию <code>on</code> .	<code>string</code>	Нет

VirtualServer.TLS.Redirect

Поле перенаправления настраивает перенаправление TLS для VirtualServer:

```
enable: true
code: 301
basedOn: scheme
```

Поле	Описание	Тип	Обязательно
<code>enable</code>	Включает перенаправление TLS для VirtualServer. Значение по умолчанию равно <code>false</code> .	<code>boolean</code>	Нет
<code>код</code>	Код состояния перенаправления. Допустимые значения: 301, 302, 307, 308. Значение по умолчанию - 301.	<code>int</code>	Нет
<code>basedOn</code>	Атрибут запроса, который Angie будет оценивать для отправки перенаправления. Допустимыми значениями являются <code>scheme</code> (схема запроса) или <code>x-forwarded-proto</code> (заголовок X-Forwarded-Proto запроса). Значение по умолчанию - <code>scheme</code> .	<code>string</code>	Нет

VirtualServer.TLS.CertManager

Поле `cert-manager` настраивает автоматическое управление сертификатами x509 для ресурсов VirtualServer с помощью `cert-manager` (cert-manager.io). Ознакомьтесь с [документацией по конфигурации cert-manager](#) для получения дополнительной информации о разворачивании и настройке эмитентов (Issuer). Пример:

```
cert-manager:
  cluster-issuer: "my-issuer-name"
```

Поле	Описание	Тип	Обязательно
<code>issuer</code>	Имя эмитента. Эмитент - это ресурс <code>cert-manager</code> , который описывает центр сертификации, способный подписывать сертификаты. Он должен находиться в том же пространстве имен, что и ресурс VirtualServer. Обратите внимание, что требуется задать <code>issuer</code> или <code>cluster-issuer</code> , но эти параметры взаимоисключающие - должен быть задан один и только один.	<code>string</code>	Нет
<code>cluster-issuer</code>	Имя ClusterIssuer. ClusterIssuer - это ресурс <code>cert-manager</code> , который описывает центр сертификации, способный подписывать сертификаты. Не имеет значения, в каком пространстве имен находится ваш VirtualServer, поскольку ClusterIssuer - это ресурсы, не относящиеся к пространствам имен. Обратите внимание, что требуется задать <code>issuer</code> или <code>cluster-issuer</code> , но эти параметры взаимоисключающие - должен быть задан один и только один.	<code>string</code>	Нет
<code>issuer-label</code>	Тип внешнего ресурса-эмитента, например <code>AWSPCAIssuer</code> . Это необходимо только для сторонних эмитентов. Его нельзя задавать, если также задан <code>cluster-issuer</code> .	<code>string</code>	Нет
<code>issuer-group</code>	Группа API внешнего контроллера-эмитента, например <code>awspca.cert-manager.io</code> . Это необходимо только для сторонних эмитентов. Его нельзя задавать, если также задан <code>cluster-issuer</code> .	<code>string</code>	Нет
<code>common-name</code>	Это поле позволяет настроить <code>spec.commonName</code> для создаваемого сертификата. Эта конфигурация добавляет CN к сертификату x509.	<code>string</code>	Нет
<code>duration</code>	Это поле позволяет настроить поле <code>spec.duration</code> для генерируемого сертификата. Оно должно быть задано с использованием формата <code>time.Duration</code> в Go, который не допускает суффикса <code>d</code> (дни). Указывайте такие значения, используя вместо них суффиксы <code>s</code> , <code>m</code> и <code>h</code> .	<code>string</code>	Нет
<code>renew-before</code>	Эта аннотация позволяет настроить поле <code>spec.renewBefore</code> для генерируемого сертификата. Оно должно быть задано с использованием формата <code>time.Duration</code> в Go, который не допускает суффикса <code>d</code> (дни). Указывайте такие значения, используя вместо них суффиксы <code>s</code> , <code>m</code> и <code>h</code> .	<code>string</code>	Нет
<code>usages</code>	Позволяет настроить поле <code>spec.usages</code> для генерируемого сертификата. Задайте строку со значениями, разделенными запятыми, т. е. соглашение о ключе, цифровая подпись, серверная аутентификация . Исчерпывающий список поддерживаемых способов использования ключей можно найти в документации API cert-manager .	<code>string</code>	Нет

VirtualServer.ExternalDNS

Поле ExternalDNS настраивает динамическое управление записями DNS для ресурсов VirtualServer с использованием [ExternalDNS](#). Ознакомьтесь с [документацией по конфигурации ExternalDNS](#) для получения дополнительной информации о развертывании и настройке ExternalDNS и поставщиков. Пример:

```
enable: true
```

Поле	Описание	Тип	Обязательно
enable	Включает интеграцию ExternalDNS для ресурса VirtualServer. Значение по умолчанию равно <code>false</code> .	string	Нет
labels	Настраивает метки, применяемые к ресурсам конечной точки, которые будут использоваться ExternalDNS.	map[string]	Нет
provider	Настраивает свойства, относящиеся к конкретному поставщику, которые содержат имя и значение конфигурации, специфичной для отдельных поставщиков DNS.	Provider /	Нет
recordTTL	TTL для записи DNS. По умолчанию это значение равно 0. См. документацию ExternalDNS TTL для определения значений по умолчанию для конкретного поставщика	int64	Нет
recordType	Тип создаваемой записи, например "A", "AAAA", "CNAME". Если значение не задано, оно автоматически вычисляется на основе внешних конечных точек.	string	Нет

VirtualServer.ExternalDNS.ProviderSpecific

Поле providerSpecific блока ExternalDNS позволяет указать свойства, специфичные для поставщика, которые представляют собой список пар "ключ-значение" для конфигураций, специфичных для отдельных поставщиков DNS. Пример:

```
- name: my-name
  value: my-value
- name: my-name2
  value: my-value2
```

Поле	Описание	Тип	Обязательно
name	Имя в паре "ключ-значение".	string	Да
value	Значение в паре "ключ-значение".	string	Да

VirtualServer.Policy

Ссылается на [ресурс Policy](#) по имени и необязательному пространству имен. Например:

```
name: access-control
```


Поле	Описание	Тип	Обязательно
<code>name</code>	Имя политики. Если политика не существует или недействительна, Angie выдаст сообщение об ошибке с кодом состояния 500.	<code>string</code>	Да
<code>namespace</code>	Пространство имен политики. Если не указано, используется пространство имен ресурса <code>VirtualServer</code> .	<code>string</code>	Нет

VirtualServer.Route

Маршрут определяет правила для сопоставления клиентских запросов с такими действиями, как передача запроса апстриму. Например:

```
path: /tea
action:
  pass: tea
```


Поле	Описание	Тип	Обязательно
<code>path</code>	Путь маршрута. Angie сопоставит его с URI запроса. Возможные значения: префикс (<code>/</code> , <code>/path</code>), точное совпадение (<code>=/exact/match</code>), регулярное выражение без учета регистра (<code>~*/Bar.*.jpg</code>) или регулярное выражение с учетом регистра (<code>~/foo.*.jpg</code>). В случае префикса (должен начинаться с <code>/</code>) или точного совпадения (должно начинаться с <code>=</code>) путь не должен содержать никаких пробельных символов, <code>{</code> , <code>}</code> или <code>;</code> . В случае регулярных выражений все двойные кавычки <code>"</code> должны быть экранированы, при этом совпадение не может заканчиваться неэкранированной обратной косой чертой <code>\</code> . Путь должен быть уникальным среди путей всех маршрутов <code>VirtualServer</code> . Дополнительные сведения см. в описании директивы <code>location</code> .	<code>string</code>	Да
<code>policies</code>	Список политик. Эти политики имеют приоритет над политиками того же типа, определенными в спецификации <code>VirtualServer</code> . Более подробную информацию смотрите в Применение политик .	<i>policy/</i>	Нет
<code>action</code>	Действие по умолчанию, выполняемое для запроса.	<i>Action</i>	Нет
<code>dos</code>	Ссылка на <code>DosProtectedResource</code> ; установка этого параметра включает защиту маршрута <code>VirtualServer</code> от DOS-атак.	<code>string</code>	Нет
<code>splits</code>	Конфигурация разделения трафика по умолчанию. Должно быть не менее 2 разделений.	<i>Split</i>	Нет
<code>matches</code>	Правила сопоставления для продвинутой маршрутизации на основе содержимого. Требуется задать <code>action</code> или <code>splits</code> по умолчанию. Несопоставленные запросы будут обрабатываться <code>action</code> или <code>splits</code> по умолчанию.	<i>matches</i>	Нет
<code>route</code>	Имя ресурса <code>VirtualServerRoute</code> , который определяет этот маршрут. Если <code>VirtualServerRoute</code> не принадлежит к тому же пространству имен, что и <code>VirtualServer</code> , необходимо включить пространство имен. Например: <code>tea-namespace/tea</code> .	<code>string</code>	Нет
<code>errorPage</code>	Настраиваемые ответы на коды ошибок. Angie будет использовать эти ответы вместо того, чтобы возвращать ответы об ошибках с серверов апстрима или ответы по умолчанию, сгенерированные Angie. Настраиваемый ответ может быть перенаправлением или сохраненным ответом. Например, это может быть перенаправление на другой URL-адрес, если вышестоящий сервер ответил кодом состояния 404.	<i>errorPage</i>	Нет
<code>location</code>	Задаёт пользовательский фрагмент в контексте местоположения. Имеет приоритет над ключом <code>ConfigMap location-snippets</code> .	<code>string</code>	Нет
<code>authReq</code>	Осуществляет авторизацию, основанную на результате выполнения подзапроса, и задает URI, на который будет отправлен подзапрос.	<code>auth_rec</code> <code> </code>	Нет
<code>authReq</code>	После завершения запроса авторизации устанавливает указанное значение для переменной в запросе.	<code>auth_rec</code> <code> </code>	Нет

Примечание

Маршрут должен включать в себя ровно одно из следующих действий: `action`, `splits` или `route`.

Maps

Определяет обязательные переменные `$jwt_claim_iat`, `$jwt_claim_iss`, `$jwt_claim_sub` и `$jwt_claim_aud` для валидации токенов в процессе *аутентификации OIDC*.

```
- variable: $jwt_claim_iat
  source: $oidc_client
  parameters:
    - value: 'myclient'
      result: '80'
- variable: $jwt_claim_iss
  source: $oidc_client
  parameters:
    - value: 'myclient'
      result: 'PROVIDER_URL'
- variable: $jwt_claim_sub
  source: $oidc_client
  parameters:
    - value: 'myclient'
      result: 'myclient'
- variable: $jwt_claim_aud
  source: $oidc_client
  parameters:
    - value: 'myclient'
      result: 'myclient'
```

Пример включения переменных `map` в зависимости от входного значения (`default`, `volatile`, `include`, `hostnames`):

```
maps:
- variable: $result_var
  source: $host
  parameters:
    - value: 'default'
      result: 'default_value'
    - value: 'volatile'
      result: ''
    - value: 'include'
      result: '/dev/stdout'
    - value: 'example.com'
      result: '1'
    - value: '*.example.com'
      result: '1'
```

См. также директиву `map` в документации Angie.

Поле	Описание	Тип	Обязательно
<code>\$jwt_claim_iat</code>	Настраивает параметр <code>iat</code> (issued at, время выпуска токена) для клиента.	string	Да
<code>\$jwt_claim_iss</code>	Параметр <code>iss</code> (issuer, издатель токена) сопоставляется с URL-адресом <code>PROVIDER_URL</code> . Этот параметр указывает на сервис, выпустивший токен.	string	Да
<code>\$jwt_claim_sub</code>	Параметр <code>sub</code> (subject, субъект токена). Идентифицирует пользователя или субъект, для которого был выдан токен.	string	Да
<code>\$jwt_claim_aud</code>	Параметр <code>aud</code> (audience, аудитория). Идентифицирует клиент, для которого был предназначен токен.	string	Да

authRequestLocations

Задаёт список точек авторизации для настройки `authRequest`. См. также директиву `auth_request` в документации Angie.

```
authRequestLocations:
- path: /auth/path
  proxyPass:
    upstreamName: "tea"
  proxyPassHeaders:
    - key: Content-Length
      value: "100"
```

Поле	Описание	Тип	Обязательно
<code>path</code>	Задаёт конкретный путь, на который будет отправляться запрос для проверки авторизации.	<code>string</code>	Да
<code>proxyPass</code>	Задаёт список апстримов, к которым будет направлен запрос. См. также директиву <code>proxy_pass</code> в документации Angie.	<code>string</code>	Да
<code>proxyPassHeaders</code>	Список заголовков, которые будут добавлены или изменены при проксировании запросов.	<code>string</code>	Да

2.7.2 Спецификация VirtualServerRoute

Ресурс `VirtualServerRoute` определяет маршрут для `VirtualServer`. Он может состоять из одного вложенного маршрута или нескольких. `VirtualServerRoute` является альтернативой объединяемым типам Ingress.

В приведенном ниже примере виртуальный сервер `cafe` из пространства имен `cafe-ns` определяет маршрут с путем `/coffee`, который далее определяется через `VirtualServerRoute coffee` из пространства имен `coffee-ns`.

VirtualServer:

```
apiVersion: k8s.angie.software/v1
kind: VirtualServer
metadata:
  name: cafe
  namespace: cafe-ns
spec:
  host: cafe.example.com
  upstreams:
    - name: tea
      service: tea-svc
      port: 80
  routes:
    - path: /tea
      action:
        pass: tea
    - path: /coffee
      route: coffee-ns/coffee
```

VirtualServerRoute:

```
apiVersion: k8s.angie.software/v1
kind: VirtualServerRoute
metadata:
```



```
name: coffee
namespace: coffee-ns
spec:
  host: cafe.example.com
  upstreams:
    - name: latte
      service: latte-svc
      port: 80
    - name: espresso
      service: espresso-svc
      port: 80
  subroutes:
    - path: /coffee/latte
      action:
        pass: latte
    - path: /coffee/espresso
      action:
        pass: espresso
```

Обратите внимание, что каждый вложенный маршрут должен иметь путь `path`, начинающийся с того же префикса (здесь `"/coffee"`), что и в маршруте `VirtualServer`. Кроме того, `host` в `VirtualServerRoute` должен совпадать с `host` у `VirtualServer`.

Поле	Описание	Тип	Обязательно
<code>host</code>	Хост (доменное имя) сервера. Это должен быть допустимый поддомен, как определено в RFC 1123, например <code>my-app</code> или <code>hello.example.com</code> . При использовании домена с подстановочным знаком, например <code>*.example.com</code> , домен должен быть заключен в двойные кавычки. Значение должно совпадать с <code>host</code> у <code>VirtualServer</code> , который ссылается на этот ресурс.	<code>string</code>	Да
<code>upstream</code>	Список апстримов.	<code>upstream</code> /	Нет
<code>subroute</code>	Список вложенных маршрутов.	<code>subroute</code> /	Нет
<code>ingressClass</code>	Указывает, какой экземпляр ANIC должен обрабатывать ресурс <code>VirtualServerRoute</code> . Значение должно совпадать с <code>ingressClassName</code> у <code>VirtualServer</code> , который ссылается на этот ресурс.	<code>string</code>	Нет

VirtualServerRoute.Subroute

Определяет правила сопоставления клиентских запросов и действий, например передача запроса апстриму. Например:

```
path: /coffee
action:
  pass: coffee
```


Поле	Описание	Тип	Обязательно
<code>path</code>	Путь вложенного маршрута. Angie сопоставит его с URI запроса. Возможные значения: префикс (<code>/</code> , <code>/path</code>), точное совпадение (<code>=/exact/match</code>), регулярное выражение без учета регистра (<code>~*/Bar.*.jpg</code>) или регулярное выражение с учетом регистра (<code>~*/foo.*.jpg</code>). В случае префикса путь должен начинаться с того же пути, что и путь маршрута <code>VirtualServer</code> , который ссылается на этот ресурс. В случае точного совпадения или регулярного выражения путь должен совпадать с путем маршрута <code>VirtualServer</code> , который ссылается на этот ресурс. В случае префикса или точного совпадения путь не должен содержать никаких пробельных символов, <code>{</code> , <code>}</code> или <code>;</code> . В случае регулярных выражений все двойные кавычки <code>"</code> должны быть экранированы, при этом совпадение не может заканчиваться неэкранированной обратной косой чертой <code>.</code> Путь должен быть уникальным среди путей всех вложенных маршрутов <code>VirtualServerRoute</code> .	<code>string</code>	Да
<code>policies</code>	Список политик. Эти политики имеют приоритет над <i>всеми</i> политиками, определенными в маршруте <code>VirtualServer</code> , который ссылается на этот ресурс. Они также имеют приоритет над политиками того же типа, определенными в <i>спецификации</i> <code>VirtualServer</code> . Более подробную информацию смотрите в разделе <i>Применение политик</i> .	<code>policy/</code>	Нет
<code>action</code>	Действие по умолчанию, выполняемое для запроса.	<code>Action</code>	Нет
<code>dos</code>	Ссылка на <code>DosProtectedResource</code> ; установка этого параметра включает защиту вложенного маршрута <code>VirtualServerRoute</code> от DOS-атак.	<code>string</code>	Нет
<code>splits</code>	Конфигурация разделения трафика по умолчанию. Должно быть не менее 2 разделений.	<code>split/</code>	Нет
<code>matches</code>	Правила сопоставления для продвинутой маршрутизации на основе содержимого. Требуется задать <code>action</code> или <code>splits</code> по умолчанию. Несопоставленные запросы будут обрабатываться <code>action</code> или <code>splits</code> по умолчанию.	<code>matches</code>	Нет
<code>errorPage</code>	Настраиваемые ответы на коды ошибок. Angie будет использовать эти ответы вместо того, чтобы возвращать ответы об ошибках с серверов апстрима или ответы по умолчанию, сгенерированные Angie. Настраиваемый ответ может быть перенаправлением или сохраненным ответом. Например, это может быть перенаправление на другой URL-адрес, если вышестоящий сервер ответил кодом состояния 404.	<code>errorPage/</code>	Нет
<code>location</code>	Задаёт пользовательский фрагмент в контексте местоположения. Переопределяет значение <code>location-snippets</code> <code>VirtualServer</code> (если задано) или ключ <code>ConfigMap</code> <code>location-snippets</code> .	<code>string</code>	Нет

Примечание

Вложенный маршрут должен включать в себя ровно одно из следующих действий: `action` или `splits`.

2.7.3 Общие части VirtualServer и VirtualServerRoute



Upstream

Апстрим определяет конечное место назначения для конфигурации маршрутизации. Например:

```
name: tea
service: tea-svc
subselector:
  version: canary
port: 80
lb-method: round_robin
fail-timeout: 10s
max-fails: 1
max-conns: 32
keepalive: 32
connect-timeout: 30s
read-timeout: 30s
send-timeout: 30s
next-upstream: "error timeout non_idempotent"
next-upstream-timeout: 5s
next-upstream-tries: 10
client-max-body-size: 2m
tls:
  enable: true
```

Примечание

Протокол WebSocket поддерживается без какой-либо дополнительной настройки.

Поле	Описание	Тип	Обязательно
name	Имя апстрима. Это должна быть допустимая метка DNS, как определено в RFC 1035. Например, допустимы значения <code>hello</code> и <code>upstream-123</code> . Имя должно быть уникальным среди всех апстримов ресурса.	string	Да
service	Название сервиса. Сервис должен принадлежать к тому же пространству имен, что и ресурс. Если сервиса не существует, Angie предположит, что у него нет конечных точек, и будет возвращать ответ 502 для запросов к этому апстриму.	string	Да
subselector	Выбирает поды внутри сервиса, используя ключи меток и значения. По умолчанию выбраны все поды сервиса.	map[string]	Нет
<div>  Примечание </div> <p>Ожидается, что указанные метки будут присутствовать в подах при их создании. Если метки подов изменяются, ANIC не увидит это изменение до тех пор, пока не будет изменено количество подов.</p>			
use-cluster	Позволяет использовать IP-адрес кластера и порт сервиса вместо использования IP-адреса и порта подов по умолчанию. Когда это поле включено, поля, которые настраивают поведение Angie, относящиеся к нескольким серверам апстрима (например, <code>lb-method</code> и <code>next-upstream</code>), не будут иметь никакого эффекта, поскольку ANIC настроит Angie только с одним сервером апстрима, который будет соответствовать IP-адресу кластера сервиса.	boolean	Нет
port	Порт службы. Если у сервиса не определен этот порт, Angie предположит, что у него нет конечных точек, и будет возвращать ответ 502 для запросов к этому апстриму. Значение должно находиться в диапазоне 1..65535.	uint16	Да
lb-method	Метод балансировки нагрузки. Чтобы использовать циклический метод, укажите <code>round_robin</code> . Значение по умолчанию указано в ключе <code>lb-method</code> ConfigMap.	string	Нет
fail-timeout	Время, в течение которого должно произойти указанное количество неудачных попыток установить связь с сервером апстрима, чтобы он считался недоступным. См. параметр <code>fail_timeout</code> директивы <code>server</code> . Значение по умолчанию задано в ключе ConfigMap <code>fail-timeout</code> .	string	Нет
max-fails	Количество неудачных попыток установить связь с сервером апстрима, которые должны произойти в течение времени, заданного в <code>fail-timeout</code> , чтобы считать сервер недоступным. См. параметр <code>max_fails</code> директивы сервера. Значение по умолчанию задано в ключе ConfigMap <code>max-fails</code> .	int	Нет
max-conn	Максимальное количество одновременных активных подключений к серверу апстрима. См. параметр <code>max_conns</code> директивы <code>server</code> . По умолчанию ограничений нет.	int	Нет
<div>  Примечание </div> <p>Если включены соединения <code>keepalive</code>, общее количество активных и неактивных соединений <code>keepalive</code> к серверу апстрима может превышать значение <code>max_conns</code>.</p>			
keepalive	Настраивает кэш для подключений к серверам апстрима. Значение 0 отключает кэш. См. директиву <code>u_keepalive</code> . Значение по умолчанию задано в ключе ConfigMap <code>keepalive</code> .	int	Нет
connect-timeout	Тайм-аут для установления соединения с сервером апстрима. См. директиву <code>proxy_connect_timeout</code> . Значение по умолчанию указано в ключе ConfigMap <code>proxy-connect-timeout</code> .	string	Нет
read-timeout	Тайм-аут для чтения ответа от сервера апстрима. См. директиву <code>proxy_read_timeout</code> . Значение по умолчанию указано в ключе ConfigMap <code>proxy-read-timeout</code> .	string	Нет
send-timeout	Тайм-аут для передачи запроса на сервер апстрима. См. директиву <code>proxy_send_timeout</code> . Значение по умолчанию указано в ключе ConfigMap <code>proxy-send-timeout</code> .	string	Нет

Upstream Buffers

Настраивает буферы, используемые для чтения ответа от сервера апстрима в рамках одного соединения.

```
number: 4
size: 8K
```

См. директиву `proxy_buffers` для получения дополнительной информации.

Поле	Описание	Тип	Обязательно
number	Задаёт количество буферов. Значение по умолчанию задано в ключе ConfigMap <code>proxy-buffers</code> .	int	Да
size	Задаёт размер буфера. Если значение не указано, то по умолчанию будет задано 8K. См. также ключ ConfigMap <code>proxy-buffers</code> .	string	Нет

Upstream.TLS

Поле	Описание	Тип	Обязательно
enable	Включает HTTPS для запросов к серверам апстрима. Значение по умолчанию равно <code>False</code> , что означает, что будет использоваться HTTP.	boolean	Нет

Примечание

По умолчанию Angie не будет проверять сертификат вышестоящего сервера. Чтобы включить проверку, настройте *политику EgressMTLS*.

Upstream.SessionCookie

Поле `SessionCookie` настраивает сохранение сеансов, что позволяет передавать запросы от одного и того же клиента на один и тот же сервер апстрима. Информация о назначенном сервере апстрима передается в сеансовом cookie, сгенерированном Angie.

В приведенном ниже примере мы настраиваем сохранение сеанса с помощью cookie сеанса для апстрима и задаем все доступные параметры:

```
name: tea
service: tea-svc
port: 80
sessionCookie:
  enable: true
  name: srv_id
  path: /
  expires: 1h
  domain: .example.com
  httpOnly: false
  secure: true
  samesite: strict
```


См. директиву `u_sticky` для получения дополнительной информации. Сеансовый cookie соответствует методу `sticky cookie`.

Поле	Описание	Тип	Обязательно
<code>enable</code>	Включает сохранение сеанса с помощью сеансового cookie для сервера апстрима. Значение по умолчанию равно <code>false</code> .	<code>boolean</code>	Нет
<code>name</code>	Имя cookie.	<code>string</code>	Да
<code>path</code>	Путь, для которого установлен cookie.	<code>string</code>	Нет
<code>expires</code>	Время, в течение которого браузер должен сохранять cookie. Может быть установлено специальное значение <code>max</code> ; тогда срок действия cookie истечет 31 декабря 2037 года в 23:55:55 по Гринвичу.	<code>string</code>	Нет
<code>domain</code>	Домен, для которого установлен cookie.	<code>string</code>	Нет
<code>httpOnly</code>	Добавляет атрибут <code>HttpOnly</code> к cookie.	<code>boolean</code>	Нет
<code>secure</code>	Добавляет атрибут <code>Secure</code> к cookie.	<code>boolean</code>	Нет
<code>samesite</code>	Добавляет атрибут <code>SameSite</code> к cookie. Допустимые значения: <code>strict</code> , <code>lax</code> , <code>none</code>	<code>string</code>	Нет

Upstream.SessionRoute

Поле `sessionRoute` настраивает сохранение маршрутов, что позволяет передавать запросы от одного и того же клиента на один и тот же сервер апстрима. Информация о назначенном сервере апстрима поддерживается в режиме `route <sticky>` в Angie.

В приведенном ниже примере мы формируем директиву Angie `sticky route $cookie_route $arg_route;`:

```
sessionRoute:
  enable: true
  variables:
    \- "$cookie_route"
    \- "$arg_route"
```

См. описание режима `route` директивы `u_sticky` для получения дополнительной информации.

Параметры:

Поле	Описание	Тип	Обязательно
<code>enable</code>	Включает сохранение сеанса в режиме <code>route</code> для сервера апстрима. Значение по умолчанию равно <code>false</code> .	<code>boolean</code>	Нет
<code>variables</code>	Список переменных, подставляемых в директиву <code>sticky route</code> в порядке следования.	<code>string[]</code>	Нет

ActiveHealthProbes

Поле позволяет настроить активную проверку работоспособности (health probe) для *upstream*. Вам необходимо задать имя проверки **name** и **upstream**, к которому она относится, а также другие параметры. Подробное описание параметров для активной проверки работоспособности см. в документации Angie, директива `upstream_probe`.

Пример:

```
activeHealthProbes:
- name: activename1
  upstream: tea-post
  uri: uri
  port: 80
  interval: 3s
  isEssential: true
  isPersistent: true
  maxBody: 10m
  fails: 4
  passes: 5
  mode: onfail
```

staticLocations

Поле позволяет задать каталог, из которого будут раздаваться статические файлы. Вы можете указать корневой каталог с помощью директивы **root** или другое расположение с помощью директивы **alias**, см. описания директив в документации Angie.

Пример конфигурации:

```
staticLocations:
- type: root
  urlPath: /static
  dirPath: /var/www/html
```

Поле	Описание	Тип	Обязательно
type	Способ определения пути к каталогу, из которого будут раздаваться статические файлы. Возможные значения: root , alias .	string	Да
urlPath	Префикс URL-адресов запрашиваемых статических файлов, используемый для location; см. описание директивы в документации Angie.	string	Да
dirPath	Каталог файловой системы, из которого будут обслуживаться запросы к указанному URL-адресу.	string	Да

Header

Определяет HTTP-заголовок:

```
name: Host
value: example.com
```


Поле	Описание	Тип	Обязательно
<code>name</code>	Имя заголовка.	<code>string</code>	Да
<code>value</code>	Значение заголовка.	<code>string</code>	Нет

Action

Определяет действие, которое необходимо выполнить для запроса.

В приведенном ниже примере клиентские запросы передаются на апстрим `coffee`:

```
path: /coffee
action:
  pass: coffee
```

Поле	Описание	Тип	Обязательно
<code>pass</code>	Передаёт запросы серверу апстрима. Апстрим с таким именем должен быть определен в ресурсе.	<code>string</code>	Нет
<code>redirect</code>	Перенаправляет запросы на указанный URL-адрес.	<code>action.redirect</code>	Нет
<code>return</code>	Возвращает предварительно сконфигурированный ответ.	<code>action.return</code>	Нет
<code>proxy</code>	Передаёт запросы апстриму, добавляет возможность изменять запрос и ответ (например, переписывать URI или изменять заголовки).	<code>action.proxy</code>	Нет

Примечание

Действие должно включать в себя ровно одно из следующих значений: `pass`, `redirect`, `return` или `proxy`.

Action.Redirect

Определяет перенаправление, возвращаемое для запроса.

В приведенном ниже примере клиентские запросы направляются на URL-адреса `http://myhost.ru`:

```
redirect:
  url: http://myhost.ru
  code: 301
```

Поле	Описание	Тип	Обязательно
<code>url</code>	URL-адрес, на который будет перенаправлен запрос. Поддерживаемые переменные Angie: <code>\$scheme</code> , <code>\$http_x_forwarded_proto</code> , <code>\$request_uri</code> , <code>\$host</code> . Переменные должны быть заключены в фигурные скобки. Например: <code>\$host\$request_uri</code> .	<code>string</code>	Да
<code>код</code>	Код состояния перенаправления. Допустимые значения: 301, 302, 307, 308. Значение по умолчанию - 301.	<code>int</code>	Нет

Action.Return

Определяет предварительно сконфигурированный ответ на запрос.

В приведенном ниже примере Angie будет отвечать предварительно настроенным ответом на каждый запрос:

```
return:
  code: 200
  type: text/plain
  body: "Hello World\n"
```

Поле	Описание	Тип	Обязательно
код	Код состояния ответа. Допустимые значения: 2XX, 4XX или 5XX. Значение по умолчанию равно 200.	int	Нет
тип	MIME-тип ответа. Значение по умолчанию - text/plain.	string	Нет
body	Основная часть ответа. Поддерживает переменные Angie*. Переменные должны быть заключены в фигурные скобки. Например: Запрос равен <code>\$request_uri</code> .	string	Да

Примечание

Поддерживаемые переменные Angie: `$request_uri`, `$request_method`, `$request_body`, `$scheme`, `$http_`, `$args`, `$arg_`, `$cookie_`, `$host`, `$request_time`, `$request_length`, `$angie_version`, `$pid`, `$connection`, `$remote_addr`, `$remote_port`, `$time_iso8601`, `$time_local`, `$server_addr`, `$server_port`, `$server_name`, `$server_protocol`, `$connections_active`, `$connections_reading`, `$connections_writing` и `$connections_waiting`.

Action.Proxy

Передаёт запросы апстриму с возможностью изменять запрос и ответ (например, переписывать URI или изменять заголовки).

В приведенном ниже примере URI запроса переписывается на `/`, а заголовки запроса и ответа изменяются:

```
proxy:
  upstream: coffee
  requestHeaders:
    pass: true
    set:
      - name: My-Header
        value: Value
      - name: Client-Cert
        value: ${ssl_client_escaped_cert}
  responseHeaders:
    add:
      - name: My-Header
        value: Value
      - name: IC-Angie-Version
        value: ${angie_version}
        always: true
    hide:
      - x-internal-version
```



```
ignore:
- Expires
- Set-Cookie
pass:
- Server
rewritePath: /
```

Поле	Описание	Тип	Обязательно
<code>upstream</code>	Имя апстрима, куда будут проксироваться запросы. Апстрим с таким именем должен быть определен в ресурсе.	<code>string</code>	Да
<code>requestHeaders</code>	Изменения заголовков запросов.	<code>Action.Proxy.RequestHeaders</code>	Нет
<code>responseHeaders</code>	Изменения в заголовках ответов.	<code>Action.Proxy.ResponseHeaders</code>	Нет
<code>rewritePath</code>	Переписанный URI. Если путь маршрута является регулярным выражением, т. е. начинается с <code>~</code> , то <code>rewritePath</code> может включать группы захвата <code>\$1-9</code> . Например, <code>\$1</code> - первая группа, и так далее.	<code>string</code>	Нет

Action.Proxy.RequestHeaders

Поле `requestHeaders` изменяет заголовки запроса к проксируемому серверу апстрима.

Поле	Описание	Тип	Обязательно
<code>pass</code>	Передаёт исходные заголовки запроса на проксируемый сервер апстрима. Дополнительные сведения см. в описании директивы <code>proxy_pass_request_headers</code> . Значение по умолчанию - <code>true</code> .	<code>bool</code>	Нет
<code>set</code>	Позволяет переопределять или добавлять поля для представления заголовков запросов, передаваемых на проксируемые серверы апстрима. Дополнительные сведения см. в описании директивы <code>proxy_set_header</code> .	<code>header /</code>	Нет

Action.Proxy.RequestHeaders.Set.Header

Определяет HTTP-заголовок:

```
name: My-Header
value: My-Value
```

Можно переопределить значение заголовка `Host` по умолчанию, которое ANIC устанавливает равным `v_host`:

```
name: Host
value: example.com
```

Поле	Описание	Тип	Обязательно
<code>name</code>	Имя заголовка.	<code>string</code>	Да
<code>value</code>	Значение заголовка. Поддерживает переменные Angie*. Переменные должны быть заключены в фигурные скобки. Например: <code>\$scheme</code> .	<code>string</code>	Нет

Примечание

Поддерживаемые переменные Angie: `$request_uri`, `$request_method`, `$request_body`, `$scheme`, `$http_`, `$args`, `$arg_`, `$cookie_`, `$host`, `$request_time`, `$request_length`, `$angie_version`, `$pid`, `$connection`, `$remote_addr`, `$remote_port`, `$time_iso8601`, `$time_local`, `$server_addr`, `$server_port`, `$server_name`, `$server_protocol`, `$connections_active`, `$connections_reading`, `$connections_writing`, `$connections_waiting`, `$ssl_cipher`, `$ssl_ciphers`, `$ssl_client_cert`, `$ssl_client_escaped_cert`, `$ssl_client_fingerprint`, `$ssl_client_i_dn`, `$ssl_client_i_dn_legacy`, `$ssl_client_raw_cert`, `$ssl_client_s_dn`, `$ssl_client_s_dn_legacy`, `$ssl_client_serial`, `$ssl_client_v_end`, `$ssl_client_v_remain`, `$ssl_client_v_start`, `$ssl_client_verify`, `$ssl_curves`, `$ssl_early_data`, `$ssl_protocol`, `$ssl_server_name`, `$ssl_session_id`, `$ssl_session_reused`.

Action.Proxy.ResponseHeaders

Поле `responseHeaders` изменяет заголовки ответа клиенту.

Поле	Описание	Тип	Обязательно
<code>hide</code>	Заголовки, которые не будут переданы в ответе клиенту с проксируемого сервера апстрима. Дополнительные сведения см. в описании директивы <code>proxy_hide_header</code> .	<code>string[]</code>	Нет
<code>pass</code>	Позволяет передавать скрытые поля заголовка клиенту с проксируемого сервера апстрима. Дополнительные сведения см. в описании директивы <code>proxy_pass_header</code> .	<code>string[]</code>	Нет
<code>ignore</code>	Отключает обработку определенных заголовков** при передаче клиенту ответа с проксируемого сервера апстрима. Дополнительные сведения см. в описании директивы <code>proxy_ignore_headers</code> .	<code>string[]</code>	Нет
<code>add</code>	Добавляет заголовки к ответу для клиента.	<code>addHeader</code>	Нет

Примечание

Скрытые заголовки по умолчанию: `Date`, `Server`, `X-Pad` и `X-Accel-...`

Примечание

Следующие поля могут быть проигнорированы: `X-Accel-Redirect`, `X-Accel-Expires`, `X-Accel-Limit-Rate`, `X-Accel-Buffering`, `X-Accel-Charset`, `Expires`, `Cache-Control`, `Set-Cookie` и `Vary`.

AddHeader

Определяет HTTP-заголовок с необязательным полем **always**:

```
name: My-Header
value: My-Value
always: true
```

Поле	Описание	Тип	Обязательно
name	Имя заголовка.	string	Да
value	Значение заголовка. Поддерживает переменные Angie*. Переменные должны быть заключены в фигурные скобки. Например: <code>\$scheme</code> .	string	Нет
always	Если установлено значение true, добавляет заголовок независимо от кода состояния ответа**. Значение по умолчанию - false. Дополнительные сведения см. в описании директивы <code>add_header</code> .	bool	Нет

Примечание

Поддерживаемые переменные Angie: `$request_uri`, `$request_method`, `$request_body`, `$scheme`, `$http_`, `$args`, `$arg_`, `$cookie_`, `$host`, `$request_time`, `$request_length`, `$angie_version`, `$pid`, `$connection`, `$remote_addr`, `$remote_port`, `$time_iso8601`, `$time_local`, `$server_addr`, `$server_port`, `$server_name`, `$server_protocol`, `$connections_active`, `$connections_reading`, `$connections_writing`, `$connections_waiting`, `$ssl_cipher`, `$ssl_ciphers`, `$ssl_client_cert`, `$ssl_client_escaped_cert`, `$ssl_client_fingerprint`, `$ssl_client_i_dn`, `$ssl_client_i_dn_legacy`, `$ssl_client_raw_cert`, `$ssl_client_s_dn`, `$ssl_client_s_dn_legacy`, `$ssl_client_serial`, `$ssl_client_v_end`, `$ssl_client_v_remain`, `$ssl_client_v_start`, `$ssl_client_verify`, `$ssl_curves`, `$ssl_early_data`, `$ssl_protocol`, `$ssl_server_name`, `$ssl_session_id`, `$ssl_session_reused`.

Примечание

Если значение **always** - false, заголовок ответа добавляется только в том случае, если код состояния ответа - это 200, 201, 204, 206, 301, 302, 303, 304, 307 или 308.

Split

Определяет вес действия в составе конфигурации разделений.

В приведенном ниже примере Angie передает 80% запросов вышестоящему **coffee-v1**, а оставшиеся 20% - **coffee-v2**:

```
splits:
- weight: 80
  action:
    pass: coffee-v1
- weight: 20
  action:
    pass: coffee-v2
```


Поле	Описание	Тип	Обязательно
weight	Вес действия. Значение должно попадать в диапазон 1..99. Сумма весов всех разделений должна быть равна 100.	int	Да
action	Действие, которое необходимо выполнить для запроса.	:ref` :action	Да

Match

Определяет сопоставление между условиями и действием или разделениями.

В приведенном ниже примере Angie направляет запросы с путем `/coffee` в разные апстримы на основе значения cookie `user`:

- `user=john -> coffee-future`
- `user=bob -> coffee-deprecated`
- Если cookie не установлен или не равен ни `john`, ни `bob`, Angie перенаправляет запрос в `coffee-stable`

```
path: /coffee
matches:
- conditions:
  - cookie: user
    value: john
    action:
      pass: coffee-future
- conditions:
  - cookie: user
    value: bob
    action:
      pass: coffee-deprecated
action:
  pass: coffee-stable
```

В следующем примере Angie направляет запросы на основе значения встроенной переменной `v_request_method`, которая представляет HTTP-метод запроса:

- все запросы `POST -> coffee-post`
- все прочие запросы `-> coffee`

```
path: /coffee
matches:
- conditions:
  - variable: $request_method
    value: POST
    action:
      pass: coffee-post
action:
  pass: coffee
```


Поле	Описание	Тип	Обязательно
<code>condition</code>	Список условий. Должен включать по крайней мере одно условие.	<code>condition</code>	Да
<code>action</code>	Действие, которое необходимо выполнить для запроса.	<code>Action</code>	Нет
<code>splits</code>	Конфигурация разбиений для разделения трафика. Должно быть указано не менее двух разделений.	<code>split[]</code>	Нет

Примечание

Сопоставление должно использовать ровно одно из следующих значений: `action` или `splits`.

Condition

Определяет условие в сопоставлении.

Поле	Описание	Тип	Обязательно
<code>header</code>	Имя заголовка. Должно состоять из буквенно-цифровых символов или <code>-</code> .	<code>string</code>	Нет
<code>cookie</code>	Имя cookie. Должно состоять из буквенно-цифровых символов или <code>-</code> .	<code>string</code>	Нет
<code>argument</code>	Имя аргумента. Должно состоять из буквенно-цифровых символов или <code>_</code> .	<code>string</code>	Нет
<code>variable</code>	Имя переменной Angie. Должно начинаться с <code>\$</code> . См. список поддерживаемых переменных после таблицы.	<code>string</code>	Нет
<code>value</code>	Значение, которому должно соответствовать условие. Как определить значение, показано ниже в таблице.	<code>string</code>	Да

Примечание

Условие должно включать ровно одно из следующих значений: `header`, `cookie`, `argument` или `variable`.

Поддерживаемые переменные Angie: `$args`, `$http2`, `$https`, `$remote_addr`, `$remote_port`, `$query_string`, `$request`, `$request_body`, `$request_uri`, `$request_method`, `$scheme`.

Значение поддерживает два вида сопоставления:

- *Сравнение строк без учета регистра.* Например:
 - `john` - сопоставление без учета регистра, которое выполняется успешно для таких строк, как `john`, `John`, `JOHN`.
 - `!john` - отрицание соответствия без учета регистра для `john`, которое выполняется успешно для таких строк, как `bob`, `anything`, `"` (пустая строка).
- *Сопоставление с регулярным выражением.* Обратите внимание, что Angie поддерживает регулярные выражения, совместимые с языком программирования Perl (PCRE). Например:
 - `~^yes` - регулярное выражение с учетом регистра, которое соответствует любой строке, начинающейся с `yes`. Например: `yes`, `yes123`.

- `!~yes` - отрицание предыдущего регулярного выражения, которое успешно выполняется для строк типа `YES`, `Yes123`, `noyes`. (Механизм отрицания не является частью синтаксиса PCRE).
- `~*no$` -- регулярное выражение без учета регистра, которое соответствует любой строке, заканчивающейся на `no`. Например: `no`, `123no`, `123NO`.

i Примечание

Значение не должно содержать неэкранированных двойных кавычек (") и не должно заканчиваться неэкранированной обратной косой чертой (\). Например, следующие значения недопустимы: `some"value`, `somevalue\`.

ErrorPage

Определяет настраиваемый ответ для маршрута на случай, когда сервер апстрима отвечает кодом состояния ошибки (или его генерирует Angie). В качестве ответа может быть задано перенаправление или сохраненный ответ. Дополнительные сведения см. в описании директивы `error_page`.

```
path: /coffee
errorPages:
- codes: [502, 503]
  redirect:
    code: 301
    url: https://angie.software
- codes: [404]
  return:
    code: 200
    body: "Original resource not found, but success!"
```

Поле	Описание	Тип	Обязательно
<code>codes</code>	Список кодов состояния ошибки.	<code>int[]</code>	Да
<code>redirect</code>	Действие перенаправления для заданных кодов состояния.	<code>errorPag</code>	Нет
<code>return</code>	Сохраненное ответное действие для заданных кодов состояния.	<code>errorPag</code>	Нет

i Примечание

Страница с ошибкой должна содержать ровно одно из следующих значений: `return` или `redirect`.

ErrorPage.Redirect

Определяет перенаправление для `errorPage`.

В приведенном ниже примере Angie отвечает перенаправлением, когда ответ от сервера апстрима имеет код состояния 404.

```
codes: [404]
redirect:
  code: 301
  url: ${scheme}://cafe.example.com/error.html
```


Поле	Описание	Тип	Обязательно
код	Код состояния перенаправления. Допустимые значения: 301, 302, 307, 308. Значение по умолчанию - 301.	int	Нет
url	URL-адрес, на который будет перенаправлен запрос. Поддерживаемые переменные Angie: <code>\$scheme</code> и <code>\$http_x_forwarded_proto</code> . Переменные должны быть заключены в фигурные скобки. Например: <code>\$scheme</code> .	string	Да

ErrorPage.Return

Определяет сохраненный ответ для `errorPage`.

В приведенном ниже примере Angie выдает сохраненный ответ, когда ответ от сервера апстрима имеет код состояния 401 или 403.

```
codes: [401, 403]
return:
  code: 200
  type: application/json
  body: |
    {"msg": "You don't have permission to do this"}
  headers:
    - name: x-debug-original-statuses
      value: ${upstream\_status}
```

Поле	Описание	Тип	Обязательно
code	Код состояния ответа. По умолчанию используется код состояния исходного ответа.	int	Нет
type	MIME-тип ответа. Значение по умолчанию - <code>text/html</code> .	string	Нет
body	Тело ответа. Поддерживаемая переменная Angie: <code>\$upstream_status</code> . Переменные должны быть заключены в фигурные скобки. Например: <code>\$upstream_status</code> .	string	Да
headers	Настраиваемые заголовки ответа.	<i>errorPag</i>	Нет <i>leader</i>

ErrorPage.Return.Header

Определяет HTTP-заголовок для сохраненного ответа у `errorPage`:

```
name: x-debug-original-statuses
value: ${upstream_status}
```

Поле	Описание	Тип	Обязательно
name	Имя заголовка.	string	Да
value	Значение заголовка. Поддерживаемая переменная Angie: <code>\$upstream_status</code> . Переменные должны быть заключены в фигурные скобки. Например: <code>\$upstream_status</code> .	string	Нет

2.7.4 Использование VirtualServer и VirtualServerRoute

Для работы с ресурсами VirtualServer и VirtualServerRoute можно использовать обычные команды kubectl, аналогично ресурсам Ingress.

Например, следующая команда создает ресурс VirtualServer, определенный в cafe-virtual-server.yaml с именем cafe:

```
kubectl apply -f cafe-virtual-server.yaml

virtualserver.k8s.angie.software "cafe" created
```

Вы можете получить ресурс, выполнив:

```
kubectl get virtualserver cafe
```

NAME	STATE	HOST	IP	PORTS	AGE
cafe	Valid	cafe.example.com	12.13.23.123	[80,443]	3m

В kubectl get и подобных командах также можно использовать короткое имя **vs** вместо virtualserver.

Работать с ресурсами VirtualServerRoute можно аналогично. В командах kubectl используйте virtualserverroute или короткое имя **vsr**.

Использование фрагментов

Фрагменты позволяют вставлять элементы конфигурации Angie в различные контексты конфигурации Angie. В приведенном ниже примере мы используем фрагменты кода для настройки нескольких функций Angie на VirtualServer:

```
apiVersion: k8s.angie.software/v1
kind: VirtualServer
metadata:
  name: cafe
  namespace: cafe
spec:
  http-snippets: |
    limit_req_zone $binary_remote_addr zone=mylimit:10m rate=1r/s;
    proxy_cache_path /tmp keys_zone=one:10m;
  host: cafe.example.com
  tls:
    secret: cafe-secret
  server-snippets: |
    limit_req zone=mylimit burst=20;
  upstreams:
    - name: tea
      service: tea-svc
      port: 80
    - name: coffee
      service: coffee-svc
      port: 80
  routes:
    - path: /tea
      location-snippets: |
        proxy_cache one;
        proxy_cache_valid 200 10m;
      action:
        pass: tea
```



```
- path: /coffee
  action:
    pass: coffee
```

Фрагменты предназначены для продвинутых пользователей Angie, которым требуется больше контроля над генерируемой конфигурацией Angie.

Однако из-за недостатков, описанных ниже, фрагменты по умолчанию отключены. Чтобы использовать фрагменты, задайте аргумент командной строки **enable-snippets**.

Недостатки использования фрагментов:

- *Сложность.* Чтобы использовать фрагменты, требуется:
 - Понимать примитивы конфигурации Angie и реализовать правильную конфигурацию Angie.
 - Понимать, как ANIC генерирует конфигурацию Angie, чтобы фрагмент не мешал другим функциям конфигурации.
- *Сниженная надежность.* Неправильный фрагмент делает конфигурацию Angie недействительной, что приведет к ошибке при перезагрузке. Это помешает применить какие-либо обновления конфигурации, включая обновления для других ресурсов VirtualServer и VirtualServerRoute, пока фрагмент не будет исправлен.
- *Последствия для безопасности.* Фрагменты предоставляют доступ к примитивам конфигурации Angie, и эти примитивы не проверяются самим ANIC. Например, через фрагмент можно настроить в Angie произвольную отправку сертификатов TLS и ключей, используемых для терминирования TLS у ресурсов Ingress и VirtualServer.

Чтобы помочь отлавливать ошибки при использовании фрагментов, ANIC сообщает об ошибках перезагрузки конфигурации в журналах, а также в полях событий и состояния ресурсов VirtualServer и VirtualServerRoute.

Примечание

Пока конфигурация Angie содержит недопустимый фрагмент, Angie будет продолжать работать с последней допустимой конфигурацией.

Валидация

Для ресурсов VirtualServer и VirtualServerRoute доступны два типа валидации:

- *Структурная валидация* с помощью **kubectl** и сервера Kubernetes API.
- *Всесторонняя валидация* с помощью ANIC.

Структурная валидация

Пользовательские определения ресурсов для VirtualServer и VirtualServerRoute включают структурную схему OpenAPI, которая описывает тип каждого поля этих ресурсов.

Если вы попытаетесь создать (или обновить) ресурс с нарушением структурной схемы (например, используете строковое значение для поля порта апстрима), **kubectl** и сервер Kubernetes API отклонят такой ресурс:

- Пример проверки **kubectl**:

```
kubectl apply -f cafe-virtual-server.yaml

error: error validating "cafe-virtual-server.yaml": error validating
```



```
data: ValidationError(VirtualServer.spec.upstreams[0].port): invalid
type for software.angie.k8s.v1.VirtualServer.spec.upstreams.port: got
"string", expected "integer"; if you choose to ignore these errors,
turn validation off with --validate=false
```

- Пример проверки сервера Kubernetes API:

```
kubectl apply -f cafe-virtual-server.yaml --validate=false

The VirtualServer "cafe" is invalid: []: Invalid value:
map[string]interface {}{ ... }: validation failure list:
spec.upstreams.port in body must be of type integer: "string"
```

Если ресурс не отклонен (т. е. не нарушает структурную схему), ANIC проверит его дополнительно.

Всесторонняя валидация

ANIC проверяет поля ресурсов `VirtualServer` и `VirtualServerRoute`. Если ресурс недействителен, ANIC отклонит его: ресурс продолжит существовать в кластере, но ANIC будет его игнорировать.

Вы можете проверить, успешно ли ANIC применил конфигурацию для `VirtualServer`. Для нашего примера `VirtualServer` `cafe` мы можем запустить:

```
kubectl describe vs cafe

. . .
Events:
  Type          Reason          Age   From                      Message
  ----
  Normal        AddedOrUpdated  16s   angie-ingress-controller  Configuration for default/
→cafe was added or updated
```

Обратите внимание, что раздел "События" (Events) включает событие `Normal` с причиной `AddedOrUpdated`, которое информирует нас о том, что конфигурация была успешно применена.

Если вы создадите недопустимый ресурс, ANIC отклонит его и выдаст событие `Rejected`. Например, если вы создадите `VirtualServer` `cafe` с двумя серверами апстрима с одинаковым именем `tea`, вы получите:

```
kubectl describe vs cafe

. . .
Events:
  Type          Reason          Age   From                      Message
  ----
  Warning        Rejected         12s   angie-ingress-controller  VirtualServer default/cafe is
→invalid and was rejected: spec.upstreams[1].name: Duplicate value: "tea"
```

Обратите внимание, что раздел "События" (Events) включает предупреждающее событие с указанием причины отклонения.

Кроме того, эта информация также доступна в поле `status` ресурса `VirtualServer`. Обратите внимание на раздел `Status` `VirtualServer`:

```
kubectl describe vs cafe

. . .
Status:
  External Endpoints:
```



```

Ip:      12.13.23.123
Ports:   [80,443]
Message: VirtualServer default/cafe is invalid and was rejected: spec.upstreams[1].
→name: Duplicate value: "tea"
Reason:   Rejected
State:    Invalid

```

ANIC проверяет ресурсы VirtualServerRoute аналогичным образом.

Примечание

Если вы внесете ошибку в существующий ресурс, ANIC отклонит его и удалит соответствующую конфигурацию из Angie.

2.7.5 Настройка с помощью ConfigMap

Вы можете дополнительно настроить конфигурацию Angie для ресурсов VirtualServer и VirtualServerRoutes, используя ConfigMap. Поддерживается большинство ключей ConfigMap, за следующими исключениями:

- proxy-hide-headers
- proxy-pass-headers
- hsts
- hsts-max-age
- hsts-include-subdomains
- hsts-behind-proxy
- redirect-to-https
- ssl-redirect

2.8 Расширенная конфигурация с помощью аннотаций

Здесь объясняется, как включить расширенную функциональность ANIC с помощью аннотаций.

Ресурс Ingress может использовать базовые функции Angie, такие как маршрутизация на основе хоста или пути и TLS-терминация. Расширенные функции, такие как переписывание URI запроса или вставка дополнительных заголовков ответа, могут быть включены с помощью аннотаций. Аннотации позволяют настраивать поведение Angie для каждого Ingress-ресурса.

Помимо расширенных функций, аннотации необходимы для настройки поведения Angie, например, установки значений таймаутов соединений.

Настройка также доступна через ресурсы *ConfigMap*: аннотации имеют приоритет.

2.8.1 Использование аннотаций

Этот пример использует аннотации для настройки конфигурации ресурса Ingress:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: cafe-ingress-with-annotations
  annotations:
    `angie.software/proxy-connect-timeout: "30s"
    `angie.software/proxy-read-timeout: "20s"
    `angie.software/client-max-body-size: "4m"
    `angie.software/server-snippets: |
      location / {
        return 302 /coffee;
      }
spec:
  rules:
  - host: cafe.example.com
    http:
      paths:
      - path: /tea
        pathType: Prefix
        backend:
          service:
            name: tea-svc
            port:
              number: 80
      - path: /coffee
        pathType: Prefix
        backend:
          service:
            name: coffee-svc
            port:
              number: 80
```

2.8.2 Валидация

ANIC проверяет аннотации ресурсов Ingress. Если Ingress некорректен, ANIC отклонит его: Ingress продолжит существовать в кластере, но ANIC будет его игнорировать.

Вы можете проверить, успешно ли ANIC применил конфигурацию для ресурса Ingress. Для примера Ingress `cafe-ingress-with-annotations` вы можете выполнить следующую команду:

```
$ kubectl describe ing cafe-ingress-with-annotations
```

```
...
Events:
  Type          Reason             Age   From                      Message
  ----          -
  Normal        AddedOrUpdated     3s    angie-ingress-controller  Configuration for default/
↪cafe-ingress-with-annotations was added or updated
```

Раздел событий включает событие Normal с причиной AddedOrUpdated, которое сообщает нам, что конфигурация была успешно применена.

Если вы создадите некорректный Ingress, ANIC отклонит его и сгенерирует событие Rejected. Например, если вы создадите Ingress `cafe-ingress-with-annotations` с аннотацией `angie.software/`

redirect-to-https, установленной на `yes please` вместо `true`, вы получите:

```
$ kubectl describe ing cafe-ingress-with-annotations
```

```
Events:
  Type      Reason      Age   From                                     Message
  ----      -
Warning    Rejected    13s   angie-ingress-controller               annotations.``angie.software/
↪redirect-to-https: Invalid value: "yes please": must be a boolean
```

Обратите внимание, что раздел событий включает событие Warning с причиной Rejected.

i Примечание

Если вы сделаете существующий Ingress некорректным, ANIC отклонит его и удалит соответствующую конфигурацию из ANIC.

2.8.3 Сводка аннотаций

В таблице ниже приведены доступные аннотации.

Общая настройка


Аннотация	Ключ ConfigMap	Описание	Значение по умолчанию	Пример
angie.software/ proxy-connect-timeout	proxy-connect-t	Устанавливает значение для директив proxy_connect_timeout и grpc_connect_timeout.	60s	
angie.software/ proxy-read-timeout	proxy-read-time	Устанавливает значение для директив proxy_read_timeout и grpc_read_timeout.	60s	
angie.software/ proxy-send-timeout	proxy-send-time	Устанавливает значение для директив proxy_send_timeout и grpc_send_timeout.	60s	
angie.software/ client-max-body-size	client-max-body	Устанавливает значение для директивы client_max_body_size (ограничивает размер тела запроса клиента). Alias для nginx.ingress.kubernetes.io/proxy-body-size.	1m	
angie.software/ proxy-body-size	proxy-body-size	Устанавливает максимально допустимый размер тела запроса клиента, передаваемого прокси-сервером дальше. См. также client_max_body_size.	1m	
angie.software/ proxy-buffering	proxy-buffering	Включает или отключает буферизацию ответов от проксируемого сервера. Alias для nginx.ingress.kubernetes.io/proxy-buffering.	True	
angie.software/ proxy-buffers	proxy-buffers	Устанавливает значение для директивы proxy_buffers. Alias для nginx.ingress.kubernetes.io/proxy-buffers-number.	Зависит от платформы.	
angie.software/ proxy-buffer-size	proxy-buffer-si	Устанавливает значение для директив proxy_buffer_size. Alias для nginx.ingress.kubernetes.io/proxy-buffer-size. и grpc_buffer_size.	Зависит от платформы.	
angie.software/ proxy-max-temp-file-size	proxy-max-temp-	Устанавливает значение для директивы proxy_max_temp_file_size. Alias для nginx.ingress.kubernetes.io/proxy-max-temp-file-size.	1024m	
angie.software/ server-tokens	server-tokens	Включает или отключает директиву server_tokens. Кроме того, с Angie можно указать строковое значение, включая пустую строку, что отключает вывод поля Server.	True	

2.8. Расширенная конфигурация с помощью аннотаций

Манипуляция URI и заголовками запросов

Аннотация	Ключ ConfigMap	Описание	Значение по умолчанию	Пример
angie.software/ proxy-hide-headers	proxy-hide-head	Устанавливает значение одной или нескольких директив proxy_hide_header. Пример: <code>"`angie.software/ proxy-hide-headers": "header-a,header-b"`</code>	Нет	
angie.software/ proxy-pass-headers	proxy-pass-head	Устанавливает значение одной или нескольких директив proxy_pass_header. Пример: <code>"`angie.software/ proxy-pass-headers": "header-a,header-b"`</code>	Нет	
angie.software/ rewrites	Нет	Конфигурирует перезапись URI с использованием директивы proxy_pass.	Нет	


Аутентификация и SSL/TLS

Аннотация	Ключ ConfigMap	Описание	Значение по умолчанию	Пример
<code>angie.software/redirect-to-https</code>	<code>redirect-to-htt</code>	Устанавливает правило перенаправления 301 на основе значения заголовка <code>http_x_forwarded_proto</code> в серверном блоке, чтобы заставить входящий трафик проходить через HTTPS. Полезно при SSL-терминации в балансировщике нагрузки перед ANIC.	False	
<code>ingress.kubernetes.io/ssl-redirect</code>	<code>ssl-redirect</code>	Устанавливает некондиционное правило перенаправления 301 для всего входящего HTTP трафика, чтобы заставить входящий трафик проходить через HTTPS.	True	
<code>angie.software/hsts</code>	<code>hsts</code>	Включает HTTP Strict Transport Security (HSTS): заголовок HSTS добавляется к ответам от проксируемых серверов. В заголовок включается директива <code>preload</code> .	False	
<code>angie.software/hsts-max-age</code>	<code>hsts-max-age</code>	Устанавливает значение директивы <code>max-age</code> заголовка HSTS.	2592000 (1 месяц)	
<code>angie.software/hsts-include-subdomains</code>	<code>hsts-include-su</code>	Добавляет директиву <code>includeSubDomains</code> в заголовки HSTS.	False	
<code>angie.software/hsts-behind-proxy</code>	<code>hsts-behind-pro</code>	Включает HSTS на основе значения заголовка запроса <code>http_x_forwarded_proto</code> . Следует использовать, только когда в балансировщике нагрузки (прокси) перед ANIC настроена TLS-терминация.	False	
<div>  Примечание Для управления перенаправлением с HTTP на HTTPS настройте аннотацию <code>angie.software/redirect-to-https</code>. </div>				
<code>angie.software/basic-auth-secret</code>	Нет	Указывает ресурс Secret с списком пользователей для HTTP Basic аутентификации.	Нет	
<code>angie.software/basic-auth-realm</code>	Нет	Указывает область.	Нет	

Прослушиватели

Аннотация	Ключ ConfigMap	Описание	Значение по умолчанию	Пример
angie.software/ listen-ports	Нет	Конфигурирует HTTP порты, на которых Angie будет слушать.	[80]	
angie.software/ listen-ports-ssl	Нет	Конфигурирует HTTPS порты, на которых Angie будет слушать.	[443]	

Бэкенд-сервисы (апстримы)

Аннотация	Ключ ConfigMap	Описание	Значение по умолчанию	Пример
nginx.ingress.kubernetes.io/backend-protocol	Нет	Устанавливает протокол для взаимодействия с backend-подами (службами) в Kubernetes.	Нет	
angie.software/lb-method	lb-method	Устанавливает метод балансировки нагрузки. Для использования метода round-robin укажите "round_robin".	"random two least_conr	
angie.software/ssl-services	Нет	Включает HTTPS или gRPC через SSL при подключении к конечным точкам сервисов.	Нет	
angie.software/grpc-services	Нет	Включает gRPC для сервисов.	Нет	
<div>  Примечание Требуется HTTP/2 (см. ключ <code>http2</code> в ConfigMap); работает только для Ingress с включенной TLS-терминацией. </div>				
angie.software/websocket-services	Нет	Включает WebSocket для сервисов.	Нет	
angie.software/max-fails	max-fails	Устанавливает значение параметра <code>max_fails</code> директивы <code>u_server</code> .	1	
angie.software/max-conns	Нет	Устанавливает значение параметра <code>max_conns</code> директивы <code>u_server</code> .	0	
angie.software/upstream-zone-size	upstream-zone-size	Устанавливает размер зоны разделяемой памяти для апстрима. Для Angie специальное значение 0 отключает зоны общей памяти. Для Angie зоны общей памяти требуются и не могут быть отключены. Специальное значение 0 будет проигнорировано.	256K	
angie.software/fail-timeout	fail-timeout	Устанавливает значение параметра <code>fail_timeout</code> директивы <code>u_server</code> .	10s	
angie.software/sticky-cookie-service	Нет	Конфигурирует сохранение сессий.	Нет	
angie.software/keepalive	keepalive	Устанавливает значение директивы <code>u_keepalive</code> . Обратите внимание, что <code>proxy_set_header Connection ""</code> ; добавляется в сгенерированную конфигурацию, когда значение > 0.	0	

2.8. Расширенная конфигурация с помощью аннотаций

85

angie.software/health-checks	Нет	Включает активные проверки состояния.	False	
angie.software/	Нет	Конфигурирует активные	False	

Ограничение скорости

Аннотация	Ключ ConfigMap	Описание	Значение по умолчанию	Пример
angie.software/limit-req-rate	Нет	Включает ограничение скорости запросов для этого Ingress, создавая limit_req_zone и применяя limit_req для каждого location. Все серверы/location одного Ingress используют одну зону. Должен иметь единицу r/s или r/m.	Нет	200r/s
angie.software/limit-req-key	Нет	Ключ, к которому применяется ограничение скорости. Может содержать текст, переменные или их комбинацию. Переменные должны быть окружены \${}.	\${binary_remote_addr}	
angie.software/limit-req-zone-size	Нет	Конфигурирует размер созданной limit_req_zone.	10m	20m
angie.software/limit-req-delay	Нет	Конфигурирует параметр delay директивы limit_req.	0	100
angie.software/limit-req-no-delay	Нет	Конфигурирует параметр nodelay директивы limit_req.	false	true
angie.software/limit-req-burst	Нет	Конфигурирует параметр burst директивы limit_req.	Нет	100
angie.software/limit-req-dry-run	Нет	Включает режим проверки. В этом режиме ограничение скорости не применяется, но количество избыточных запросов учитывается, как обычно, в зоне общей памяти.	false	true
angie.software/limit-req-log-level	Нет	Устанавливает желаемый уровень логирования для случаев, когда сервер отказывается обрабатывать запросы из-за превышения скорости или задержек в обработке запросов. Разрешенные значения: info, notice, warn или error.	error	info
angie.software/limit-req-reject-code	Нет	Устанавливает код состояния, который возвращается в ответ на отклоненные запросы. Должен находиться в диапазоне 400..599.	429	503
angie.software/limit-req-scale	Нет	Включает постоянное ограничение скорости, деля настроенное значение скорости на количество подов Ingress, в настоящее время обслуживающих трафик. Эта корректировка обеспечивает постоянство ограничения скорости, даже если количество подов изменяется из-за автоскейлинга.	false	true

Фрагменты и пользовательские шаблоны

Аннотация	Ключ ConfigMap	Описание	Значение по умолчанию	Пример
angie.software/location-snippets	location-snippets	Устанавливает пользовательский фрагмент в контексте location.	Нет	
angie.software/server-snippets	server-snippets	Устанавливает пользовательский фрагмент в контексте server.	Нет	

ГЛАВА 3

Журналы и мониторинг

ANIC поддерживает мониторинг с помощью метрик Prometheus и ведение журналов.

Просмотр журналов

Просмотр состояния сервера

Просмотр состояния ресурсов

3.1 Просмотр журналов

В ANIC можно посмотреть журнал процесса Ingress Controller (процесса, который генерирует конфигурацию Angie и перезагружает Angie для ее применения), а также журнал доступа и журнал ошибок Angie. Все записи идут в стандартный вывод и стандартный поток ошибок процесса Ingress Controller. Чтобы посмотреть журнал, вы можете выполнить команду `kubectl logs` для пода ANIC.

Например:

```
kubectl logs <angie-ingress-pod> -n angie-ingress
```

3.1.1 Журнал процесса Ingress Controller

Журнал процесса Ingress Controller можно настроить с помощью *аргумента командной строки* `-v`, который задает уровень детализации журнала. Значение по умолчанию — `1`, при этом значении записывается минимальное количество событий. Значение `3` полезно для устранения неполадок: вы сможете увидеть, как Ingress Controller получает обновления от Kubernetes API, генерирует конфигурацию Angie и перезагружает Angie.

3.1.2 Журналы Angie

Angie включает два журнала:

- *Журнал доступа.* В этот журнал Angie записывает информацию о запросах клиентов сразу после обработки запроса. Журнал доступа настраивается через *ключи ConfigMap*: `log-format` для HTTP- и HTTPS-трафика и `stream-log-format` для сквозного трафика TCP, UDP и TLS. Вы можете отключить запись журнала доступа с помощью ключа `access-log-off`.
- *Журнал ошибок.* В этот журнал Angie записывает информацию о возникших проблемах различного уровня критичности. Этот журнал настраивается через ключ `error-log-level` в *ConfigMap*. Чтобы включить отладочное логирование, установите значение `debug`, а также задайте аргумент командной строки `-angie-debug`. Angie будет запущен с отладочной версией бинарного файла `angie-debug`.

3.2 Просмотр состояния сервера

Angie поставляется со *страницей статуса Stub Status*, которая отображает основные метрики.

3.2.1 Доступ к Stub Status

Необходимые условия:

1. Stub Status должен быть включен по умолчанию. Убедитесь, что *аргумент командной строки* `angie-status` задан как `true`.
2. По умолчанию Stub Status доступен на порту 8080. Порт можно изменить с помощью аргумента командной строки `angie-status-port`. Если ваш порт отличается от 8080, измените команду `kubectl proxy` ниже.

Чтобы открыть страницу статуса, выполните следующие действия:

1. Используйте команду `kubectl port-forward`, чтобы перенаправить соединения с порта 8080 на вашем локальном компьютере на порт 8080 пода ANIC (замените `<angie-ingress-pod>` на фактическое имя пода):

```
kubectl port-forward <angie-ingress-pod> 8080:8080 --namespace=angie-ingress
```

2. Откройте браузер по адресу `http://127.0.0.1:8080/stub_status`.

Чтобы получить доступ к `stub status` извне (без `kubectl port-forward`), выполните следующие действия:

1. Настройте с помощью *аргумента командной строки* `-angie-status-allow-cidrs` блоками IP/CIDR, для которых вы хотите разрешить доступ к статусу. По умолчанию доступ разрешен для `127.0.0.1::1`.
2. Используйте IP/порт, через который доступен под ANIC, чтобы подключиться к странице статуса по пути `/stub_status`.

3.3 Просмотр состояния ресурсов

3.3.1 Ресурсы Ingress

Ресурс Ingress может иметь состояние, куда входит адрес (IP-адрес или DNS-имя), через который становятся общедоступными узлы этого ресурса Ingress. Адрес можно видеть в выходных данных команды `kubectl get ingress` в столбце ADDRESS, как показано ниже:

```
$ kubectl get ingresses
```

NAME	HOSTS	ADDRESS	PORTS	AGE
myapp-ingress	myapp.example.com	12.13.23.123	80, 443	2m

ANIC должен быть сконфигурирован таким образом, чтобы сообщать о состоянии Ingress:

1. Используйте флаг командной строки `-report-ingress-status`.
2. Определите источник для внешнего адреса. Это может быть:
 - Определенный пользователем адрес, указанный в ключе ConfigMap `external-status-address`.
 - Служба типа LoadBalancer, настроенная с внешним IP-адресом или без него и указанная с помощью флага командной строки `-external-service`.

См. документацию по [ключам ConfigMap](#) и [аргументам командной строки](#).

Примечание

При завершении работы ANIC не очищает статус ресурсов Ingress.

3.3.2 Ресурсы VirtualServer и VirtualServerRoute

Ресурс VirtualServer или VirtualServerRoute содержит поле состояния с информацией о состоянии ресурса и IP-адрес, через который становятся общедоступными узлы этого ресурса. Вы можете увидеть состояние в выходных данных команд `kubectl get virtualservers` или `kubectl get virtualserverroutes`, как показано ниже:

```
$ kubectl get virtualservers
```

NAME	STATE	HOST	IP	PORTS	AGE
myapp	Valid	myapp.example.com	12.13.23.123	[80,443]	34s

Чтобы просмотреть внешний адрес имени узла, связанный с ресурсом VirtualServer, используйте параметр `-o wide`:

```
$ kubectl get virtualservers -o wide
```

NAME	STATE	HOST	IP	EXTERNALHOSTNAME
↪ mysite	Valid	mysite.example.com	PORTS	AGE
↪ 12345678.eu-west-2.elb.amazonaws.com			[80,443]	106s

Примечание

При наличии нескольких адресов отображается только первый из них.

Чтобы просмотреть дополнительные адреса или дополнительную информацию о *статусе* ресурса, используйте следующую команду:

```
$ kubectl describe virtualserver <NAME>

. . .
Status:
  External Endpoints:
    Ip:      12.13.23.123
    Ports:   [80,443]
  Message:   Configuration for myapp/myapp was added or updated
  Reason:    AddedOrUpdated
  State:     Valid
```

Спецификация состояния

Следующие поля отображаются как в статусе VirtualServer, так и в статусе VirtualServerRoute:

Поле	Описание	Тип
State	Текущее состояние ресурса. Возможные значения: Valid (допустимо), Warning (внимание) и Invalid (недопустимо). Дополнительные сведения см. в поле message .	string
Reason	Причина последнего обновления.	string
Message	Дополнительная информация о состоянии.	string
ExternalEndpoints	Список внешних конечных точек, для которых хосты ресурса являются общедоступными.	<i>externalEndpoint/</i>

Следующее поле отображается только в состоянии VirtualServerRoute:

Поле	Описание	Тип
ReferenceName	VirtualServer, который ссылается на этот VirtualServerRoute. Формат: пространство имен/имя.	string

ExternalEndpoint

Поле	Описание	Тип
IP	Внешний IP-адрес.	string
Hostname	Адрес имени узла внешнего балансировщика LoadBalancer.	string
Ports	Список внешних портов.	string

ANIC должен быть настроен таким образом, чтобы сообщать о состоянии VirtualServer или VirtualServerRoute.

Если вы хотите, чтобы ANIC сообщал о **внешних конечных точках**, определите источник для внешнего адреса. Это может быть:

- Определенный пользователем адрес, указанный в ключе ConfigMap **external-status-address**.
- Служба типа LoadBalancer, настроенная с внешним IP-адресом или без него и указанная с помощью флага командной строки **-external-service**.

См. документацию по *ключам ConfigMap* и *аргументам командной строки*.

Остальные поля будут включаться в отчет и без настроенного внешнего адреса.

Примечание

При завершении работы ANIC не очищает статус ресурсов VirtualServer и VirtualServerRoute.

3.3.3 Ресурсы Policy

Ресурс Policy включает в себя поле статуса с информацией о состоянии ресурса. Вы можете увидеть статус в выходных данных команды `kubectl get policy`, как показано ниже:

```
$ kubectl get policy
```

NAME	STATE	AGE
webapp-policy	Valid	30s

Чтобы просмотреть дополнительные адреса или дополнительную информацию о *статусе* ресурса, используйте следующую команду:

```
$ kubectl describe policy <NAME>
```

```

. . .
Status:
  Message: Configuration for default/webapp-policy was added or updated
  Reason:  AddedOrUpdated
  State:   Valid

```

Спецификация состояния

В состоянии Policy отображаются следующие поля:

Поле	Описание	Тип
State	Текущее состояние ресурса. Возможные значения: Valid (допустимо) или Invalid (недопустимо). Дополнительные сведения см. в поле message .	string
Reason	Причина последнего обновления.	string
Message	Дополнительная информация о состоянии.	string

3.3.4 Ресурсы TransportServer

Ресурс TransportServer включает в себя поле состояния с информацией о состоянии ресурса. Вы можете увидеть его в выходных данных команды `kubectl get transportserver`, как показано ниже:

```
$ kubectl get transportserver
```

NAME	STATE	REASON	AGE
dns-tcp	Valid	AddedOrUpdated	47m

Чтобы просмотреть дополнительные адреса или дополнительную информацию о *статусе* ресурса, используйте следующую команду:


```
$ kubectl describe transportserver <NAME>
```

```
. . .
```

```
Status:
```

```
Message: Configuration for default/dns-tcp was added or updated
```

```
Reason: AddedOrUpdated
```

```
State: Valid
```

Спецификация состояния

В состоянии TransportServer отображаются следующие поля:

Поле	Описание	Тип
State	Текущее состояние ресурса. Возможные значения: Valid (допустимо), Warning (внимание) и Invalid (недопустимо). Дополнительные сведения см. в поле message .	string
Reason	Причина последнего обновления.	string
Message	Дополнительная информация о состоянии.	string

ГЛАВА 4

Типовые задачи

В этом разделе собраны примеры настройки ANIC под разные задачи.

Создание кастомных страниц ошибок

Сопоставление путей с помощью регулярных выражений

4.1 Создание кастомных страниц ошибок

В ANIC можно настроить кастомные страницы ошибок, например с более информативными сообщениями для пользователей (для статусов наподобие 502).

Добавить кастомную страницу можно двумя способами:

- пересобрать образ ANIC с новой страницей;
- настроить ConfigMap без пересборки образа ANIC.

4.1.1 Пересборка образа ANIC с кастомной страницей

Этот вариант предполагает создание нового Docker-образа ANIC, который включает кастомную страницу ошибки. Такой вариант подойдет, если кастомная страница ошибки редко меняется и ее удобно включить в образ.

Выполните следующие шаги:

1. Создайте Dockerfile и добавьте в него новую страницу ошибки:

```
FROM anic.docker.angie.software/anic:latest
COPY 502.html /usr/share/angie/html/
```

2. Соберите и загрузите новый образ в кластер (см. Установка с помощью Helm).
3. Разверните обновленный образ в кластере.
4. Добавьте в Ingress-ресурс аннотацию для использования кастомной страницы ошибки:


```
annotations:
  angie.software/server-snippets: |
    error_page 502 /502.html;
    location = /502.html {
      root /usr/share/angie/html;
      internal;
    }
```

Примечание

Также можно использовать ConfigMap, тогда правило будет применяться ко всем серверам.

Этот способ подходит как для Ingress-ресурса, так и для VirtualServer.

4.1.2 Использование ConfigMap без пересборки образа

Если нет возможности пересобрать образ ANIC, можно поместить страницу ошибки в под через ConfigMap. Также этот способ удобен, если требуется оперативно менять содержимое страницы ошибки.

Выполните следующие шаги:

1. Создайте ConfigMap с HTML-страницей ошибки:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: error-page
  namespace: angie
data:
  502.html: |
    <!DOCTYPE html>
    <html>
      <head>
        <title>ERROR PAGE</title>
      </head>
      <body>
        ERROR PAGE
      </body>
    </html>
```

2. Добавьте в файл `values.yaml` эту ConfigMap:

```
volumes:
- name: error-page
  configMap:
    name: error-page

## The volumeMounts of the Ingress Controller pods.
volumeMounts: []
- name: error-page
  mountPath: /usr/share/angie/html/custom_error
```

3. Добавьте в Ingress-ресурс аннотацию:

```
annotations:
  angie.software/server-snippets: |
```



```
error_page 502 /502.html;
location = /502.html {
    root /usr/share/angie/html/custom_error;
    internal;
}
```

Этот способ подходит как для Ingress-ресурса, так и для VirtualServer.

4.2 Сопоставление путей с помощью регулярных выражений

Здесь показано, как настроить пути в ресурсах Ingress и Mergeable Ingress с помощью *аннотации path-regex* и регулярных выражений.

Для аннотации `path-regex` возможны следующие значения:

- `case_insensitive` - этот модификатор удобно использовать для маршрутизации, когда регистр не важен, и вы хотите избежать проблем, например с неправильным вводом URL пользователями (запросы на `/Tea`, `/tea`, и `/TEA` будут направлены на один и тот же ресурс).
- `case_sensitive` - этот модификатор можно использовать для более строгого контроля маршрутизации, когда регистр имеет значение, например в API (`/User/123` и `/user/123` могут означать разные сущности).

Рекомендуем также ознакомиться с работой директивы `location` в документации Angie.

4.2.1 Пример настройки ресурса Ingress

Чтобы настроить регулярные выражения для путей ресурса Ingress, выполните следующие шаги:

1. Добавьте в файл `cafe-ingress.yaml` аннотацию `angie.software/path-regex` со значением `case_sensitive`.

Пример

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: cafe-ingress
  annotations:
    angie.software/path-regex: "case_sensitive"
spec:
  tls:
    - hosts:
        - cafe.example.com
      secretName: cafe-secret
  rules:
    - host: cafe.example.com
      http:
        paths:
          - path: /tea/[A-Z0-9]
            backend:
              serviceName: tea-svc
              servicePort: 80
          - path: /coffee/[A-Z0-9]
            backend:
```



```
serviceName: coffee-svc
servicePort: 80
```

2. Выполните команду:

```
kubectl create -f cafe-ingress.yaml
```

Пути `tea` и `coffee` в конфигурации Angie будут выглядеть следующим образом:

```
location ~ "/tea/[A-Z0-9]"
```

```
location ~ "/coffee/[A-Z0-9]"
```

Примечание

Обратите внимание, что модификатор регулярного выражения `case_sensitive` применяется ко всем путям.

3. Если вы хотите изменить значение на `case_insensitive`, обновите файл.

Пример

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: cafe-ingress
  annotations:
    angie.software/path-regex: "case_insensitive"
spec:
  tls:
    - hosts:
        - cafe.example.com
    secretName: cafe-secret
  rules:
    - host: cafe.example.com
  http:
    paths:
      - path: /tea/[A-Z0-9]
        backend:
          serviceName: tea-svc
          servicePort: 80
      - path: /coffee/[A-Z0-9]
        backend:
          serviceName: coffee-svc
          servicePort: 80
```

Теперь пути `/tea/[A-Z0-9]` и `/coffee/[A-Z0-9]` в конфигурации Angie будут выглядеть так:

```
location ~* "/tea/[A-Z0-9]"
```

```
location ~* "/coffee/[A-Z0-9]"
```


Примечание

Обратите внимание, что модификатор регулярного выражения `case_insensitive` применяется ко всем путям.

4.2.2 Пример настройки ресурса Mergeable Ingress

Создание Master Ingress и Minion Ingress

1. Создайте Master Ingress в файле `cafe-master.yaml`.

Пример

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: cafe-ingress-master
  annotations:
    angie.software/mergeable-ingress-type: "master"
spec:
  ingressClassName: angie
  tls:
  - hosts:
    - cafe.example.com
    secretName: cafe-secret
  rules:
  - host: cafe.example.com
```

2. Выполните команду:

```
kubectl create -f cafe-master.yaml
```

3. Проверьте, что Master Ingress создан:

```
kubectl get ingress cafe-ingress-master
```

NAME	CLASS	HOSTS	PORTS	AGE
cafe-ingress-master	angie	cafe.example.com	80, 443	29s

```
kubectl describe ingress cafe-ingress-master
```

```
Name:          cafe-ingress-master
Labels:        <none>
Namespace:     default
Address:
Ingress Class:  angie
Default backend: <default>
TLS:
cafe-secret terminates cafe.example.com
Rules:
Host      Path  Backends
----    -
*         *    <default>
Annotations:  angie.software/mergeable-ingress-type: master
```



```
Events:
Type      Reason           Age   From                                     Message
-----
Normal    AddedOrUpdated    62s   angie-ingress-controller               Configuration for
↪ default/cafe-ingress-master was added or updated
```

- Создайте первый Minion Ingress для tea в файле tea-minion.yaml.

Пример

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: cafe-ingress-tea-minion
  annotations:
    angie.software/mergeable-ingress-type: "minion"
spec:
  ingressClassName: angie
  rules:
  - host: cafe.example.com
    http:
      paths:
      - path: /tea
        pathType: Prefix
      backend:
        service:
          name: tea-svc
          port:
            number: 80
```

- Выполните команду:

```
kubectl create -f tea-minion.yaml
```

- Проверьте, что Minion Ingress создан:

```
kubectl get ingress cafe-ingress-tea-minion
```

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
cafe-ingress-tea-minion	angie	cafe.example.com		80	23m

```
kubectl describe ingress cafe-ingress-tea-minion
```

```
Name:                cafe-ingress-tea-minion
Labels:              <none>
Namespace:           default
Address:
Ingress Class:       angie
Default backend:     <default>
Rules:
Host                Path    Backends
-----
cafe.example.com    /tea    tea-svc:80 (10.244.0.6:8080,10.244.0.7:8080,10.244.0.
↪ 8:8080)
Annotations:         angie.software/mergeable-ingress-type: minion
Events:
```


Type	Reason	Age	From	Message
-----	-----	----	----	-----
Normal	AddedOrUpdated	24m	angie-ingress-controller	Configuration for ↵ ↵default/cafe-ingress-tea-minion was added or updated

7. Создайте второй Minion Ingress для coffee в файле coffee-minion.yaml.

Пример

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: cafe-ingress-coffee-minion
  annotations:
    angie.software/mergeable-ingress-type: "minion"
spec:
  ingressClassName: angie
  rules:
  - host: cafe.example.com
    http:
      paths:
      - path: /coffee
        pathType: Prefix
        backend:
          service:
            name: coffee-svc
            port:
              number: 80
```

8. Выполните команду:

```
kubectl create -f coffee-minion.yaml
```

9. Проверьте, что Minion Ingress создан:

```
kubectl get ingress cafe-ingress-coffee-minion
```

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
cafe-ingress-coffee-minion	angie	cafe.example.com		80	5m21s

```
kubectl describe ingress cafe-ingress-coffee-minion
```

```
Name:          cafe-ingress-coffee-minion
Labels:         <none>
Namespace:     default
Address:
Ingress Class:  angie
Default backend: <default>
Rules:
Host           Path           Backends
-----
cafe.example.com /coffee       coffee-svc:80 (10.244.0.6:8080,10.244.0.7:8080,10.
↵244.0.8:8080)
Annotations:    angie.software/mergeable-ingress-type: minion
Events:
Type           Reason           Age           From           Message
```



```

-----
Normal   AddedOrUpdated   5m52s   angie-ingress-controller   Configuration for
↪ default/cafe-ingress-coffee-minion was added or updated

```

Теперь у вас есть Master Ingress и два Minion Ingress. Два Minion Ingress определяются путями /tea и /coffee.

Модификация путей с помощью регулярных выражений

Ниже показано, как изменить пути /tea и /coffee с помощью регулярных выражений.

1. Добавьте аннотацию `path-regex` со значением `case_insensitive` в Minion Ingress (tea) и измените путь с помощью регулярных выражений (в примере ниже: /tea/[A-Z0-9]).

Пример

```

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: cafe-ingress-tea-minion
  annotations:
    angie.software/mergeable-ingress-type: "minion"
    angie.software/path-regex: "case_insensitive"
spec:
  ingressClassName: angie
  rules:
  - host: cafe.example.com
    http:
      paths:
      - path: /tea/[A-Z0-9]
        pathType: Prefix
        backend:
          service:
            name: tea-svc
            port:
              number: 80

```

2. Примените изменения:

```
kubectl apply -f tea-minion.yaml
```

3. Проверьте, что изменения применились:

```

kubectl describe ingress cafe-ingress-tea-minion

Name:                cafe-ingress-tea-minion
Labels:              <none>
Namespace:           default
Address:
Ingress Class:       angie
Default backend:     <default>
Rules:
Host                Path                Backends
-----
cafe.example.com    /tea/[A-Z0-9]       tea-svc:80 (10.244.0.6:8080,10.244.0.7:8080,10.
↪ 244.0.8:8080)

```



```
Annotations:      Angie Software/mergeable-ingress-type: minion
                  Angie Software/path-regex: case_insensitive

Events:
Type      Reason          Age          From          Message
-----
Normal    AddedOrUpdated    47s (x2 over 34m)    Angie Ingress Controller    Configuration for default/cafe-ingress-tea-minion was added or updated
```

Добавленная аннотация `path-regex` обновляет путь `/tea/[A-Z0-9]` с использованием модификатора регулярного выражения `case_insensitive`.

Обновленный путь (`location`) в конфигурационном файле Angie будет выглядеть так:

```
location ~* "/tea/[A-Z0-9]"
```

Примечание

Обратите внимание, что аннотация `path-regex` применяется только к путям, определенным в соответствующем Minion Ingress (`tea`). Пути, определенные во втором Minion Ingress (`coffee`), не меняются.

- Аналогичным образом используйте модификатор регулярного выражения `case_sensitive` для второго Minion Ingress (`coffee`) в файле `coffee-minion.yaml`.

Пример

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: cafe-ingress-coffee-minion
  annotations:
    Angie Software/mergeable-ingress-type: "minion"
    Angie Software/path-regex: "case_sensitive"
spec:
  ingressClassName: angie
  rules:
  - host: cafe.example.com
    http:
      paths:
      - path: /coffee/[A-Za-z0-9]
        pathType: Prefix
        backend:
          service:
            name: coffee-svc
            port:
              number: 80
```

- Примените изменения:

```
kubectl apply -f coffee-minion.yaml
```

- Проверьте, что изменения применились:

```
kubectl describe ingress cafe-ingress-coffee-minion

Name:      cafe-ingress-coffee-minion
```



```

Labels:          <none>
Namespace:       default
Address:
Ingress Class:   angie
Default backend: <default>
Rules:
Host             Path             Backends
----
cafe.example.com /coffee/[A-Za-z0-9]  coffee-svc:80 (10.244.0.10:8080,10.244.0.
→9:8080)
Annotations:     angie.software/mergeable-ingress-type: minion
                  angie.software/path-regex: case_sensitive
Events:
Type   Reason           Age   From                                     Message
----   -
Normal AddedOrUpdated  11m   angie-ingress-controller Configuration for
→default/cafe-ingress-coffee-minion was added or updated

```

Добавленная аннотация **path-regex** обновляет путь `/coffee/[A-Za-z0-9]`, используя модификатор регулярного выражения **case_sensitive**.

Обновленный путь в конфигурационном файле Angie будет выглядеть так:

```
location ~ "/coffee/[A-Za-z0-9]"
```


ГЛАВА 5

Примеры для пользовательских ресурсов

В этом разделе собраны примеры настройки и типовые конфигурации для пользовательских ресурсов.

Базовая конфигурация

Базовая аутентификация

Базовая балансировка TCP- и UDP-трафика

Контроль доступа

Конфигурация для нескольких пространств имен

Ограничение скорости запросов (rateLimit)

Поддержка переписывания (rewrites)

Распределение трафика

Расширенная маршрутизация

Сохранение сессий

Cert-manager

gRPC

Ingress MTLS

JWKS

JWT

OIDC

TLS Passthrough

5.1 Базовая конфигурация

Ниже приведен пример настройки балансировки нагрузки с терминацией TLS для простого веб-приложения с использованием ресурса VirtualServer. Приложение "cafe" позволяет получить `tea` через сервис `tea` или `coffee` через сервис `coffee`. Выбор определяется URI HTTP-запроса: запросы с URI, оканчивающимися на `/tea`, возвращают `tea`, а с `/coffee` — `coffee`.

5.1.1 Предварительные действия

1. Установите ANIC с включенной поддержкой пользовательских ресурсов.
2. Сохраните публичный IP-адрес ANIC в переменной оболочки:

```
$ IC_IP=<ваш_IP-адрес>
```

3. Сохраните HTTPS-порт ANIC в переменной оболочки:

```
$ IC_HTTPS_PORT=<номер порта>
```

5.1.2 Настройка базовой конфигурации

1. Создайте Deployment и Service для `tea` и `coffee`:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: coffee
spec:
  replicas: 2
  selector:
    matchLabels:
      app: coffee
  template:
    metadata:
      labels:
        app: coffee
    spec:
      containers:
        - name: coffee
          image: angiesoftware/angie-hello:plain-text
          ports:
            - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: coffee-svc
spec:
  ports:
    - port: 80
      targetPort: 8080
      protocol: TCP
      name: http
  selector:
    app: coffee
---
```



```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: tea
spec:
  replicas: 1
  selector:
    matchLabels:
      app: tea
  template:
    metadata:
      labels:
        app: tea
    spec:
      containers:
        - name: tea
          image: angiesoftware/angie-hello:plain-text
          ports:
            - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: tea-svc
spec:
  ports:
    - port: 80
      targetPort: 8080
      protocol: TCP
      name: http
  selector:
    app: tea

```

Примените настройки:

```
$ kubectl create -f cafe.yaml
```

2. Создайте секрет с TLS-сертификатом и ключом:

```

apiVersion: v1
kind: Secret
metadata:
  name: cafe-secret
type: kubernetes.io/tls
data:
  tls.crt: ↪LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSOtLS0tCk1JSURMakNDQWhZQ0NRREFPRj10THNhWFdqQU5CZ2txaGtpRz13MEJBUU
  tls.key: ↪LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSU1Fb3dJQkFBSONBUUVBcWVpcCs3TXZOYWRRJN21mM01wUHJ3ZG

```

Примените настройки:

```
$ kubectl create -f cafe-secret.yaml
```

3. Создайте ресурс VirtualServer:

```

apiVersion: k8s.angie.software/v1
kind: VirtualServer

```



```
metadata:
  name: cafe
spec:
  host: cafe.example.com
  tls:
    secret: cafe-secret
  upstreams:
  - name: tea
    service: tea-svc
    port: 80
  - name: coffee
    service: coffee-svc
    port: 80
  routes:
  - path: /tea
    action:
      pass: tea
  - path: /coffee
    action:
      pass: coffee
```

Примените настройки:

```
$ kubectl create -f cafe-virtual-server.yaml
```

4. Протестируйте конфигурацию.

Проверьте, что конфигурация успешно применена, посмотрев события ресурса VirtualServer:

```
$ kubectl describe virtualserver cafe
```

Ожидаемый результат:

```
...
Events:
  Type    Reason             Age   From              Message
  ----    -
  Normal  AddedOrUpdated     7s    ANIC              Configuration for
→default/cafe was added or updated
```

5. Получите доступ к приложению с помощью curl.

Используйте опцию `--insecure`, чтобы отключить проверку сертификата, и `--resolve`, чтобы задать IP-адрес и порт Ingress-контроллера для домена `cafe.example.com`.

Чтобы получить `coffee`:

```
$ curl --resolve cafe.example.com:$IC_HTTPS_PORT:$IC_IP \
https://cafe.example.com:$IC_HTTPS_PORT/coffee --insecure
```

Ожидаемый результат:

```
Server address: 10.16.1.182:80
Server name: coffee-7dbb5795f6-tnbtq
...
```

Чтобы получить `tea`:

```
$ curl --resolve cafe.example.com:$IC_HTTPS_PORT:$IC_IP \
https://cafe.example.com:$IC_HTTPS_PORT/tea --insecure
```


Ожидаемый результат:

```
Server address: 10.16.0.149:80
Server name: tea-7d57856c44-zlftd
...
```

5.2 Настройка базовой аутентификации

ANIC поддерживает аутентификацию запросов с использованием модуля Auth Basic. Ниже приведен пример развертывания веб-приложения, настройки балансировщика для ресурса VirtualServer и применения политики базовой аутентификации.

5.2.1 Предварительные действия

1. Установите ANIC.
2. Сохраните публичный IP-адрес ANIC в переменной оболочки:

```
$ IC_IP=<ваш_IP-адрес>
```

3. Сохраните HTTP-порт ANIC в переменную оболочки:

```
$ IC_HTTP_PORT=<номер порта>
```

5.2.2 Настройка базовой аутентификации

1. Создайте Deployment и Service для приложения:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: coffee
spec:
  replicas: 2
  selector:
    matchLabels:
      app: coffee
  template:
    metadata:
      labels:
        app: coffee
    spec:
      containers:
        - name: coffee
          image: angiesoftware/angie-hello:plain-text
          ports:
            - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: coffee-svc
spec:
  ports:
    - port: 80
```



```

    targetPort: 8080
    protocol: TCP
    name: http
  selector:
    app: coffee
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tea
spec:
  replicas: 1
  selector:
    matchLabels:
      app: tea
  template:
    metadata:
      labels:
        app: tea
    spec:
      containers:
        - name: tea
          image: angiesoftware/angie-hello:plain-text
          ports:
            - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: tea-svc
spec:
  ports:
    - port: 80
      targetPort: 8080
      protocol: TCP
      name: http
  selector:
    app: tea

```

Примените настройки:

```
$ kubectl apply -f cafe.yaml
```

- Создайте секрет типа `angie.software/httpasswd` с именем `cafe-passwd`, который будет использоваться для базовой аутентификации. Секрет должен содержать список пар `user:password` в формате Base64:

```

apiVersion: v1
kind: Secret
metadata:
  name: cafe-secret
type: kubernetes.io/tls
data:
  tls.crt:   

  ↪LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSB0tLS0tCk1JSURMakNDQWhZQ0NRREFPRj10THNhWFdqQU5CZ2txaGtpRz13MEJBUE  

  tls.key:   

  ↪LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSU1Fb3dJQkFBSONBUUVBcWVpcCs3TXZOYWRRJN21mMO1wUHJ3ZC

```



```
kind: Secret
metadata:
  name: cafe-passwd
apiVersion: v1
type: angie.software/htpasswd
stringData:
  htpasswd: |
    foo:$2y$10$e4CiBwLq9JW93jV8r9CW.RE6fbsT3szmIsUhwqYuPfVlggXiBY76
    qux:$apr1$st218vzc$A3H7I83N9vLmczj73Byi3/
  # bar
  # quux
```

Примените настройки:

```
$ kubectl apply -f cafe-passwd.yaml
```

- Создайте политику `basic-auth-policy`, которая ссылается на секрет из предыдущего шага и разрешает запросы к веб-приложению только при наличии действительной пары `user:password`.

```
apiVersion: k8s.angie.software/v1
kind: Policy
metadata:
  name: basic-auth-policy
spec:
  basicAuth:
    realm: Cafe App
    secret: cafe-passwd
```

Примените настройки:

```
$ kubectl apply -f basic-auth-policy.yaml
```

- Создайте ресурс `VirtualServer` для веб-приложения:

```
apiVersion: k8s.angie.software/v1
kind: VirtualServer
metadata:
  name: cafe
spec:
  host: cafe.example.com
  policies:
    - name: basic-auth-policy
  upstreams:
    - name: tea
      service: tea-svc
      port: 80
    - name: coffee
      service: coffee-svc
      port: 80
  routes:
    - path: /tea
      action:
        pass: tea
    - path: /coffee
      action:
        pass: coffee
```

Примените настройки:


```
$ kubectl apply -f cafe-virtual-server.yaml
```

Обратите внимание, что VirtualServer должен ссылаться на политику `basic-auth-policy`, созданную на шаге 3.

5. Протестируйте конфигурацию.

Если вы укажете неверные данные авторизации, ANIC отклонит запрос для указанного ресурса VirtualServer при попытке обратиться к приложению:

```
$ curl --resolve cafe.example.com:$IC_HTTP_PORT:$IC_IP http://cafe.example.com:
→$IC_HTTP_PORT/
<html>
<head><title>401 Authorization Required</title></head>
<body>
<center><h1>401 Authorization Required</h1></center>
<hr><center>Angie/1.8.1</center>
</body>
</html>
```

Если вы укажете действительные данные, запрос будет выполнен:

```
$ curl --resolve cafe.example.com:$IC_HTTPS_PORT:$IC_IP https://cafe.example.com:
→$IC_HTTPS_PORT/coffee --insecure -u foo:bar

Server address: 10.244.0.6:8080
Server name: coffee-7b9b4bbd99-bdbxm
Date: 20/Jun/2024:11:43:34 +0000
URI: /coffee
Request ID: f91f15d1af17556e552557df2f5a0dd2
```

5.3 Базовая балансировка TCP- и UDP-трафика

Ниже приведен пример разворачивания DNS-сервера в кластере и настройки балансировки TCP- и UDP-трафика для него с использованием ресурса `TransportServer`. ANIC будет передавать все соединения или датаграммы, поступающие на порт 5353, в поды DNS-сервера.

5.3.1 Предварительные действия

1. Установите ANIC:
 - Убедитесь, что ресурс `GlobalConfiguration` развернут и ANIC использует его.
 - Откройте порт 5353 как для TCP-, так и для UDP-трафика.
2. Сохраните публичный IP-адрес ANIC в переменной оболочки:

```
$ IC_IP=<ваш_IP-адрес>
```

3. Сохраните порт 5353, используемый ANIC, в переменной оболочки:

```
$ IC_5353_PORT=<номер порта>
```

Примечание

Если вы хотите настроить ANIC с помощью сервиса `LoadBalancer`, то в нем нельзя использовать протоколы TCP и UDP одновременно. В этом случае создайте два отдельных

сервиса: для TCP и для UDP. Соответственно, у вас будет два разных публичных IP-адреса (см. примеры на последнем шаге).

4. Убедитесь, что у вас установлена утилита `dig` (используется для тестирования).

Примечание

В процессе установки ANIC ресурс `GlobalConfiguration` должен быть развернут в пространстве имен `angie-ingress` с именем `angie-configuration`. Если это не так, обновите файл `global-configuration.yaml`, правильно указав пространство имен и имя.

5.3.2 Балансировка TCP/UDP-трафика

1. Разверните DNS-сервер:
 - Разверните две реплики `CoreDNS`, настроенные на пересылку DNS-запросов на 8.8.8.8.
 - Создайте сервис `coredns`, который откроет порт 5353 как для TCP, так и для UDP.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: coredns
data:
  Corefile: |
    .:5353 {
      forward . 8.8.8.8:53
      log
    }
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: coredns
spec:
  replicas: 2
  selector:
    matchLabels:
      app: coredns
  template:
    metadata:
      labels:
        app: coredns
    spec:
      containers:
        - name: coredns
          image: coredns/coredns:1.10.0
          args: [ "-conf", "/etc/coredns/Corefile" ]
          volumeMounts:
            - name: config-volume
              mountPath: /etc/coredns
              readOnly: true
          ports:
            - containerPort: 5353
              name: dns
              protocol: UDP
```



```

- containerPort: 5353
  name: dns-tcp
  protocol: TCP
  securityContext:
    readOnlyRootFilesystem: true
volumes:
- name: config-volume
  configMap:
    name: coredns
    items:
    - key: Corefile
      path: Corefile
---
apiVersion: v1
kind: Service
metadata:
  name: coredns
spec:
  selector:
    app: coredns
  ports:
  - name: dns
    port: 5353
    protocol: UDP
  - name: dns-tcp
    port: 5353
    protocol: TCP

```

Примените настройки:

```
$ kubectl apply -f dns.yaml
```

2. Настройте слушатели.

Обновите ресурс `GlobalConfiguration`, добавив два слушателя: один для TCP-порта 5353 и один для UDP-порта 5353:

```

apiVersion: k8s.angie.software/v1alpha1
kind: GlobalConfiguration
metadata:
  name: angie-configuration
  namespace: angie-ingress
spec:
  listeners:
  - name: dns-udp
    port: 5353
    protocol: UDP
  - name: dns-tcp
    port: 5353
    protocol: TCP

```

Примените настройки:

```
$ kubectl apply -f global-configuration.yaml
```

3. Проверьте, что конфигурация успешно применена, просмотрев события `GlobalConfiguration`:


```
$ kubectl describe gc angie-configuration -n angie-ingress
```

Пример вывода:

```
Events:
  Type      Reason      Age           From    Message
  ----      -
  Normal    Updated     0s (x2 over 10s)  anic    GlobalConfiguration anic/angie-
→configuration was updated
```

4. Настройте балансировку нагрузки.

Создайте ресурс `TransportServer`, чтобы настроить балансировку TCP:

```
apiVersion: k8s.angie.software/v1alpha1
kind: TransportServer
metadata:
  name: dns-tcp
spec:
  listener:
    name: dns-tcp
    protocol: TCP
  upstreams:
  - name: dns-app
    service: coredns
    port: 5353
  action:
    pass: dns-app
```

Примените настройки:

```
$ kubectl apply -f transport-server-tcp.yaml
```

5. Проверьте, что конфигурация успешно применена:

```
$ kubectl describe ts dns-tcp
```

Пример вывода:

```
Events:
  Type      Reason      Age    From    Message
  ----      -
  Normal    AddedOrUpdated  3s    anic    Configuration for default/dns-tcp was
→added or updated
```

6. Создайте ресурс `TransportServer`, чтобы настроить балансировку UDP:

```
apiVersion: k8s.angie.software/v1alpha1
kind: TransportServer
metadata:
  name: dns-udp
spec:
  listener:
    name: dns-udp
    protocol: UDP
  upstreams:
  - name: dns-app
    service: coredns
    port: 5353
```



```
upstreamParameters:
  udpRequests: 1
  udpResponses: 1
action:
  pass: dns-app
```

Примените настройки:

```
$ kubectl apply -f transport-server-udp.yaml
```

7. Проверьте, что конфигурация успешно применена:

```
$ kubectl describe ts dns-udp
```

Пример вывода:

```
Events:
  Type      Reason             Age   From   Message
  ----      -
  Normal    AddedOrUpdated     0s    anic   Configuration for default/dns-udp was
→added or updated
```

8. Протестируйте конфигурацию.

Чтобы проверить, что настроенный балансировщик нагрузки TCP/UDP работает, разрешите DNS-имя `kubernetes.io` с помощью DNS-сервера.

Разрешение `kubernetes.io` через TCP:

```
$ dig @$IC_IP -p $IC_5353_PORT kubernetes.io +tcp

; <<>> DiG 9.10.3-P4-Debian <<>> @<REDACTED> -p 5353 kubernetes.io +tcp
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 44784
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;kubernetes.io.                IN      A

;; ANSWER SECTION:
kubernetes.io.                3596    IN      A      147.75.40.148

;; Query time: 134 msec
;; SERVER: <REDACTED>#5353(<REDACTED>)
;; WHEN: Thu Mar 12 22:01:55 UTC 2020
;; MSG SIZE rcvd: 71
```

Разрешение `kubernetes.io` через UDP:

```
$ dig @$IC_IP -p $IC_5353_PORT kubernetes.io

; <<>> DiG 9.10.3-P4-Debian <<>> @<REDACTED> -p 5353 kubernetes.io
; (1 server found)
;; global options: +cmd
;; Got answer:
```



```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 39087
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;kubernetes.io.                IN      A

;; ANSWER SECTION:
;kubernetes.io.                2157    IN      A      147.75.40.148

;; Query time: 134 msec
;; SERVER: <REDACTED>#5353(<REDACTED>)
;; WHEN: Thu Mar 12 22:02:12 UTC 2020
;; MSG SIZE rcvd: 71
```

5.4 Контроль доступа

Ниже приведен пример развертывания веб-приложения, настройки балансировки нагрузки с помощью VirtualServer и применения политики управления доступом для запрета и разрешения трафика из определенной подсети.

5.4.1 Предварительные действия

1. Установите ANIC.
2. Сохраните публичный IP-адрес ANIC в переменной оболочки:

```
$ IC_IP=<ваш_IP-адрес>
```

3. Сохраните HTTP-порт ANIC в переменной оболочки:

```
$ IC_HTTP_PORT=<номер порта>
```

5.4.2 Настройка контроля доступа

1. Создайте Deployment и Service для приложения:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webapp
spec:
  selector:
    matchLabels:
      app: webapp
  template:
    metadata:
      labels:
        app: webapp
    spec:
      containers:
        - name: webapp
          image: angiesoftware/angie-hello:plain-text
```



```

        ports:
          - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: webapp-svc
spec:
  ports:
    - port: 80
      targetPort: 8080
      protocol: TCP
      name: http
  selector:
    app: webapp

```

Примените настройки:

```
$ kubectl apply -f webapp.yaml
```

- Создайте политику `webapp-policy`, которая запрещает запросы от клиентов с IP-адресами из подсети 10.0.0.0/8. Убедитесь, что поле `deny` в файле `access-control-policy-deny.yaml` настроено в соответствии с вашей средой:

```

apiVersion: k8s.angie.software/v1
kind: Policy
metadata:
  name: webapp-policy
spec:
  accessControl:
    deny:
      - 10.0.0.0/8

```

Примените настройки:

```
$ kubectl apply -f access-control-policy-deny.yaml
```

- Создайте ресурс `VirtualServer` для веб-приложения:

```

apiVersion: k8s.angie.software/v1
kind: VirtualServer
metadata:
  name: webapp
spec:
  host: webapp.example.com
  policies:
    - name: webapp-policy
  upstreams:
    - name: webapp
      service: webapp-svc
      port: 80
  routes:
    - path: /
      action:
        pass: webapp

```

Примените настройки:


```
$ kubectl apply -f virtual-server.yaml
```

Обратите внимание, что VirtualServer должен ссылаться на политику `webapp-policy`, созданную на шаге 2.

4. Протестируйте конфигурацию.

Попробуйте обратиться к приложению:

```
$ curl --resolve webapp.example.com:$IC_HTTP_PORT:$IC_IP http://webapp.example.  
→com:$IC_HTTP_PORT
```

Ожидаемый результат:

```
<html>  
<head><title>403 Forbidden</title></head>  
<body>  
<center><h1>403 Forbidden</h1></center>  
<hr><center>Angie/1.8.1</center>  
</body>  
</html>
```

Ответ 403 Forbidden означает успешное срабатывание политики блокировки запросов.

5. Обновите политику, чтобы разрешить запросы от клиентов из подсети 10.0.0.0/8. Убедитесь, что поле `allow` в файле `access-control-policy-allow.yaml` настроено в соответствии с вашей средой:

```
apiVersion: k8s.angie.software/v1  
kind: Policy  
metadata:  
  name: webapp-policy  
spec:  
  accessControl:  
    allow:  
      - 10.0.0.0/8
```

Обновите политику:

```
$ kubectl apply -f access-control-policy-allow.yaml
```

6. Повторно протестируйте конфигурацию.

Попробуйте обратиться к приложению:

```
$ curl --resolve webapp.example.com:$IC_HTTP_PORT:$IC_IP http://webapp.example.  
→com:$IC_HTTP_PORT
```

Ожидаемый результат:

```
Server address: 10.64.0.13:8080  
Server name: webapp-5cbbc7bd78-wf85w
```

Ответ 200 OK означает успешное разрешение запроса после обновления политики.

5.5 Конфигурация для нескольких пространств имен

Ниже приведен пример использования ресурсов `VirtualServer` и `VirtualServerRoute` при настройке балансировки нагрузки для модифицированного приложения "cafe" из примера базовой конфигурации. Конфигурация балансировки нагрузки, а также `Deployments` и `Services` размещены в нескольких пространствах имен.

Вместо одного пространства имен теперь используются три: `tea`, `coffee` и `cafe`:

- В пространстве имен `tea` созданы `Deployment`, `Service` и соответствующая конфигурация балансировки нагрузки.
- В пространстве имен `coffee` созданы `Deployment`, `Service` и соответствующая конфигурация балансировки нагрузки.
- В пространстве имен `cafe` создан секрет с TLS-сертификатом и ключом, а также конфигурация балансировки нагрузки для приложения "cafe". Эта конфигурация ссылается на конфигурации `coffee` и `tea`.

5.5.1 Предварительные действия

1. Установите ANIC с включенными пользовательскими ресурсами.
2. Сохраните публичный IP-адрес ANIC в переменной оболочки:

```
$ IC_IP=<ваш_IP-адрес>
```

3. Сохраните HTTPS-порт ANIC в переменной оболочки:

```
$ IC_HTTPS_PORT=<номер порта>
```

5.5.2 Настройка конфигурации для нескольких пространств имен

1. Создайте необходимые пространства имен `tea`, `coffee` и `cafe`:

```
apiVersion: v1
kind: Namespace
metadata:
  name: cafe
---
apiVersion: v1
kind: Namespace
metadata:
  name: tea
---
apiVersion: v1
kind: Namespace
metadata:
  name: coffee
```

Примените настройки:

```
$ kubectl create -f namespaces.yaml
```

2. Создайте `Deployment` и `Service` для `tea` в пространстве имен `tea`:

```
apiVersion: apps/v1
kind: Deployment
metadata:
```



```

name: tea
namespace: tea
spec:
  replicas: 1
  selector:
    matchLabels:
      app: tea
  template:
    metadata:
      labels:
        app: tea
    spec:
      containers:
        - name: tea
          image: angiesoftware/angie-hello:plain-text
          ports:
            - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: tea-svc
  namespace: tea
spec:
  ports:
    - port: 80
      targetPort: 8080
      protocol: TCP
      name: http
  selector:
    app: tea

```

Примените настройки:

```
$ kubectl create -f tea.yaml
```

3. Создайте Deployment и Service для `coffee` в пространстве имен `coffee`:

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: coffee
  namespace: coffee
spec:
  replicas: 1
  selector:
    matchLabels:
      app: coffee
  template:
    metadata:
      labels:
        app: coffee
    spec:
      containers:
        - name: coffee
          image: angiesoftware/angie-hello:plain-text
          ports:
            - containerPort: 8080

```



```
---
apiVersion: v1
kind: Service
metadata:
  name: coffee-svc
  namespace: coffee
spec:
  ports:
    - port: 80
      targetPort: 8080
      protocol: TCP
      name: http
  selector:
    app: coffee
```

Примените настройки:

```
$ kubectl create -f coffee.yaml
```

4. Создайте ресурс VirtualServerRoute для **tea** в пространстве имен **tea**:

```
apiVersion: k8s.angie.software/v1
kind: VirtualServerRoute
metadata:
  name: tea
  namespace: tea
spec:
  host: cafe.example.com
  upstreams:
    - name: tea
      service: tea-svc
      port: 80
  subroutes:
    - path: /tea
      action:
        pass: tea
```

Примените настройки:

```
$ kubectl create -f tea-virtual-server-route.yaml
```

5. Создайте ресурс VirtualServerRoute для **coffee** в пространстве имен **coffee**:

```
apiVersion: k8s.angie.software/v1
kind: VirtualServerRoute
metadata:
  name: coffee
  namespace: coffee
spec:
  host: cafe.example.com
  upstreams:
    - name: coffee
      service: coffee-svc
      port: 80
  subroutes:
    - path: /coffee
      action:
        pass: coffee
```


Примените настройки:

```
$ kubectl create -f coffee-virtual-server-route.yaml
```

6. Создайте секрет с TLS-сертификатом и ключом в пространстве имен **cafe**:

```
apiVersion: v1
kind: Secret
metadata:
  name: cafe-secret
  namespace: cafe
type: kubernetes.io/tls
data:
  tls.crt: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSOtLS0tCk1JSURMakNDQWhZQONRREFPRj10THNhWFdqQU5CZ2txaGtpRz13MEJBUU
  tls.key: LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSU1Fb3dJQkFBS0NBUEVBcWVpcCs3TXZOYWRRJN21mM01wUHJ3ZC
```

Примените настройки:

```
$ kubectl create -f cafe-secret.yaml
```

7. Создайте ресурс VirtualServer для приложения "cafe" в пространстве имен **cafe**:

```
apiVersion: k8s.angie.software/v1
kind: VirtualServer
metadata:
  name: cafe
  namespace: cafe
spec:
  host: cafe.example.com
  tls:
    secret: cafe-secret
  routes:
    - path: /tea
      route: tea/tea
    - path: /coffee
      route: coffee/coffee
```

Примените настройки:

```
$ kubectl create -f cafe-virtual-server.yaml
```

8. Протестируйте конфигурацию.

Проверьте, что конфигурация была успешно применена, просмотрев события ресурсов VirtualServerRoute и VirtualServer:

```
$ kubectl describe virtualserverroute tea -n tea
```

Вывод:

```
Events:
  Type    Reason                  Age   From    Message
  ----    -
  Warning  NoVirtualServersFound  2m    ANIC    No
  VirtualServer references VirtualServerRoute tea/tea
  Normal   AddedOrUpdated         1m    ANIC    Configuration
  for tea/tea was added or updated
```



```
$ kubectl describe virtualserverroute coffee -n coffee
```

Вывод:

```
Events:
  Type          Reason              Age   From          Message
  ----          -
Warning        NoVirtualServersFound 2m    ANIC          No
→VirtualServer references VirtualServerRoute coffee/coffee
Normal         AddedOrUpdated      1m    ANIC          Configuration
→for coffee/coffee was added or updated
```

```
$ kubectl describe virtualserver cafe -n cafe
```

Вывод:

```
Events:
  Type          Reason              Age   From          Message
  ----          -
Normal         AddedOrUpdated      1m    ANIC          Configuration for cafe/
→cafe was added or updated
```

- Используйте `curl` для доступа к приложению. Опция `--insecure` отключает проверку сертификата, так как используется самоподписанный сертификат. Опция `--resolve` позволяет установить IP-адрес и HTTPS-порт Ingress-контроллера для доменного имени приложения "cafe".

Чтобы получить `coffee`:

```
$ curl --resolve cafe.example.com:$IC_HTTPS_PORT:$IC_IP https://cafe.example.com:
→$IC_HTTPS_PORT/coffee --insecure
```

Ожидаемый результат:

```
Server address: 10.16.1.193:80
Server name: coffee-7dbb5795f6-mltpf
...
```

Чтобы получить `tea`:

```
$ curl --resolve cafe.example.com:$IC_HTTPS_PORT:$IC_IP https://cafe.example.com:
→$IC_HTTPS_PORT/tea --insecure
```

Ожидаемый результат:

```
Server address: 10.16.0.157:80
Server name: tea-7d57856c44-674b8
...
```


5.6 Ограничение скорости запросов

В этом примере разворачивается веб-приложение, настраивается балансировка нагрузки с помощью VirtualServer и применяется политика ограничения скорости запросов.

5.6.1 Предварительные действия

1. Установите ANIC.
2. Сохраните публичный IP-адрес ANIC в переменной оболочки:

```
$ IC_IP=<ваш_IP-адрес>
```

3. Сохраните HTTP-порт ANIC в переменной оболочки:

```
$ IC_HTTP_PORT=<номер порта>
```

5.6.2 Развертывание веб-приложения

1. Создайте Deployment и Service для приложения:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webapp
spec:
  replicas: 1
  selector:
    matchLabels:
      app: webapp
  template:
    metadata:
      labels:
        app: webapp
    spec:
      containers:
        - name: webapp
          image: angiesoftware/angie-hello:plain-text
          ports:
            - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: webapp-svc
spec:
  ports:
    - port: 80
      targetPort: 8080
      protocol: TCP
      name: http
  selector:
    app: webapp
```

Примените настройки:


```
$ kubectl apply -f webapp.yaml
```

- Создайте политику с именем `rate-limit-policy`, которая разрешает только один запрос в секунду с одного IP-адреса.

```
apiVersion: k8s.angie.software/v1
kind: Policy
metadata:
  name: rate-limit-policy
spec:
  rateLimit:
    rate: 1r/s
    key: ${binary_remote_addr}
    zoneSize: 10M
```

Примените настройки:

```
$ kubectl apply -f rate-limit.yaml
```

- Создайте ресурс `VirtualServer` для веб-приложения:

```
apiVersion: k8s.angie.software/v1
kind: VirtualServer
metadata:
  name: webapp
spec:
  host: webapp.example.com
  policies:
    - name: rate-limit-policy
  upstreams:
    - name: webapp
      service: webapp-svc
      port: 80
  routes:
    - path: /
      action:
        pass: webapp
```

Примените настройки:

```
$ kubectl apply -f virtual-server.yaml
```

`VirtualServer` ссылается на политику `rate-limit-policy`, созданную выше.

- Протестируйте конфигурацию.

Если вы будете запрашивать приложение с частотой выше одного запроса в секунду, ANIC начнет отклонять ваши запросы:

```
$ curl --resolve webapp.example.com:$IC_HTTP_PORT:$IC_IP http://webapp.example.
→com:$IC_HTTP_PORT/
Server address: 10.8.1.19:8080
Server name: webapp-dc88fc766-zr7f8
...
```

```
$ curl --resolve webapp.example.com:$IC_HTTP_PORT:$IC_IP http://webapp.example.
→com:$IC_HTTP_PORT/
<html>
<head><title>503 Service Temporarily Unavailable</title></head>
```



```
<body>
<center><h1>503 Service Temporarily Unavailable</h1></center>
<hr><center>Angie/1.8.1</center>
</body>
</html>
```

Примечание

Вывод команды сокращен для наглядности примера.

5.7 Поддержка переписывания (rewrites)

ANIC позволяет преобразовывать (переписывать) URI запроса перед отправкой в приложение. Например, путь запроса `/tea/green` может быть перезаписан как `/green`. Для настройки изменения URI необходимо использовать `ActionProxy` в `VirtualServer` или `VirtualServerRoute`.

5.7.1 Пример с префиксом в пути

В следующем примере нагрузка между двумя приложениями, требующими изменения URI, балансируется с помощью префиксного сопоставления:

```
apiVersion: k8s.angie.software/v1
kind: VirtualServer
metadata:
  name: cafe
spec:
  host: cafe.example.com
  upstreams:
  - name: tea
    service: tea-svc
    port: 80
  - name: coffee
    service: coffee-svc
    port: 80
  routes:
  - path: /tea/
    action:
      proxy:
        upstream: tea
        rewritePath: /
  - path: /coffee
    action:
      proxy:
        upstream: coffee
        rewritePath: /beans
```

Изменения URI для `tea-svc` (обратите внимание, что запросы к `/tea` перенаправляются на `/tea/` с добавлением косой черты в конце):

Исходный URI	Переписанный URI
<code>/tea/</code>	<code>/</code>
<code>/tea/abc</code>	<code>/abc</code>

Изменения URI для `coffee-svc`:

Исходный URI	Переписанный URI
/coffee	/beans
/coffee/	/beans/
/coffee/abc	/beans/abc

5.7.2 Пример с регулярными выражениями

Если путь представляет собой регулярное выражение, а не префикс или точное соответствие, `rewritePath` может содержать группы захвата `$1-9`.

Пример:

```
apiVersion: k8s.angie.software/v1
kind: VirtualServer
metadata:
  name: cafe
spec:
  host: cafe.example.com
  upstreams:
  - name: tea
    service: tea-svc
    port: 80
  routes:
  - path: ~ /tea/(?.*)
    action:
      proxy:
        upstream: tea
        rewritePath: /$1
```

В этом примере группа захвата `(.*)` используется в `rewritePath` как `/$1`. Это необходимо для передачи оставшейся части URI запроса (после `/tea`).

Примеры изменения URI для `tea-svc`:

Исходный URI	Переписанный URI
/tea	/
/tea/	/
/tea/abc	/abc

5.8 Распределение трафика

Ниже приведен пример использования ресурса `VirtualServer` для настройки распределения трафика в приложении `Cafe`.

Относительно базовой конфигурации были внесены следующие изменения:

- Вместо одной версии сервиса `coffee` теперь есть две: `coffee-v1-svc` и `coffee-v2-svc`.
- 90% трафика направляется на `coffee-v1-svc`, а оставшиеся 10% — на `coffee-v2-svc`.
- Для упрощения примера убраны TLS-терминация и сервис `tea`.

5.8.1 Предварительные действия

1. Установите ANIC с включенными пользовательскими ресурсами.
2. Сохраните публичный IP-адрес ANIC в переменной оболочки:

```
$ IC_IP=<ваш_IP-адрес>
```

3. Сохраните HTTP-порт ANIC в переменной оболочки:

```
$ IC_HTTP_PORT=<номер порта>
```

5.8.2 Настройка распределения трафика

1. Создайте Deployment и Service для coffee:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: coffee-v1
spec:
  replicas: 2
  selector:
    matchLabels:
      app: coffee-v1
  template:
    metadata:
      labels:
        app: coffee-v1
    spec:
      containers:
        - name: coffee-v1
          image: angiesoftware/angie-hello:plain-text
          ports:
            - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: coffee-v1-svc
spec:
  ports:
    - port: 80
      targetPort: 8080
      protocol: TCP
      name: http
  selector:
    app: coffee-v1
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: coffee-v2
spec:
  replicas: 2
  selector:
    matchLabels:
      app: coffee-v2
```



```

template:
  metadata:
    labels:
      app: coffee-v2
  spec:
    containers:
      - name: coffee-v2
        image: angiesoftware/angie-hello:plain-text
        ports:
          - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: coffee-v2-svc
spec:
  ports:
    - port: 80
      targetPort: 8080
      protocol: TCP
      name: http
  selector:
    app: coffee-v2

```

Примените настройки:

```
$ kubectl create -f cafe.yaml
```

2. Настройте балансировку нагрузки.

Создайте ресурс VirtualServer:

```

apiVersion: k8s.angie.software/v1
kind: VirtualServer
metadata:
  name: cafe
spec:
  host: cafe.example.com
  upstreams:
    - name: coffee-v1
      service: coffee-v1-svc
      port: 80
    - name: coffee-v2
      service: coffee-v2-svc
      port: 80
  routes:
    - path: /coffee
      splits:
        - weight: 90
          action:
            pass: coffee-v1
        - weight: 10
          action:
            pass: coffee-v2

```

Примените настройки:

```
$ kubectl create -f cafe-virtual-server.yaml
```


3. Проверьте, что конфигурация успешно применена, просмотрев события VirtualServer:

```
$ kubectl describe virtualserver cafe
```

Пример вывода:

```
Events:
  Type      Reason             Age   From              Message
  ----      -
  Normal    AddedOrUpdated     5s    anic               Configuration for
  → default/cafe was added or updated
```

4. Проверьте работу приложения с помощью curl. Используйте --resolve, чтобы указать IP-адрес и HTTP-порт ANIC для домена cafe.example.com. Выполните несколько запросов, чтобы убедиться, что ANIC направляет трафик на разные версии сервиса coffee:

```
$ curl --resolve cafe.example.com:$IC_HTTP_PORT:$IC_IP http://cafe.example.com:
  → $IC_HTTP_PORT/coffee
```

Результат:

- 90% запросов будут направлены на coffee-v1-svc:

```
Server address: 10.16.0.151:80
Server name: coffee-v1-78754bdcfb-7xp27
...
```

- 10% запросов будут направлены на coffee-v2-svc:

```
Server address: 10.16.0.152:80
Server name: coffee-v2-7fd446968b-lwhgcd
...
```

5.9 Расширенная маршрутизация

Ниже приведен пример настройки расширенной маршрутизации с помощью ресурса VirtualServer для приложения "cafe" из примера настройки базовой конфигурации.

Внесены следующие изменения:

- Вместо одной версии сервиса tea теперь две: tea-post-svc и tea-svc. POST-запросы для tea направляются в tea-post-svc. Остальные запросы (например, GET) направляются в tea-svc.
- Вместо одной версии сервиса coffee теперь две: coffee-v1-svc и coffee-v2-svc. Запросы, содержащие cookie version=v2, направляются в coffee-v2-svc. Все остальные запросы направляются в coffee-v1-svc.
- Для упрощения из примера удалена поддержка TLS.

5.9.1 Предварительные действия

1. Установите ANIC с включенными пользовательскими ресурсами.
2. Сохраните публичный IP-адрес ANIC в переменной оболочки:

```
$ IC_IP=<ваш_IP-адрес>
```

3. Сохраните HTTP-порт ANIC в переменной оболочки:

```
$ IC_HTTP_PORT=<номер порта>
```

5.9.2 Настройка расширенной маршрутизации

1. Создайте Deployment и Service для `coffee` и `tea`:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: coffee-v1
spec:
  replicas: 1
  selector:
    matchLabels:
      app: coffee-v1
  template:
    metadata:
      labels:
        app: coffee-v1
    spec:
      containers:
        - name: coffee-v1
          image: angiesoftware/angie-hello:plain-text
          ports:
            - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: coffee-v1-svc
spec:
  ports:
    - port: 80
      targetPort: 8080
      protocol: TCP
      name: http
  selector:
    app: coffee-v1
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: coffee-v2
spec:
  replicas: 1
  selector:
    matchLabels:
      app: coffee-v2
```



```

template:
  metadata:
    labels:
      app: coffee-v2
  spec:
    containers:
      - name: coffee-v2
        image: angiesoftware/angie-hello:plain-text
        ports:
          - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: coffee-v2-svc
spec:
  ports:
    - port: 80
      targetPort: 8080
      protocol: TCP
      name: http
  selector:
    app: coffee-v2
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tea-post
spec:
  replicas: 1
  selector:
    matchLabels:
      app: tea-post
  template:
    metadata:
      labels:
        app: tea-post
    spec:
      containers:
        - name: tea-post
          image: angiesoftware/angie-hello:plain-text
          ports:
            - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: tea-post-svc
spec:
  ports:
    - port: 80
      targetPort: 8080
      protocol: TCP
      name: http
  selector:
    app: tea-post
---

```



```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: tea
spec:
  replicas: 1
  selector:
    matchLabels:
      app: tea
  template:
    metadata:
      labels:
        app: tea
    spec:
      containers:
        - name: tea
          image: angiesoftware/angie-hello:plain-text
          ports:
            - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: tea-svc
spec:
  ports:
    - port: 80
      targetPort: 8080
      protocol: TCP
      name: http
  selector:
    app: tea

```

Примените настройки:

```
$ kubectl create -f cafe.yaml
```

2. Создайте ресурс VirtualServer:

```

apiVersion: k8s.angie.software/v1
kind: VirtualServer
metadata:
  name: cafe
spec:
  host: cafe.example.com
  upstreams:
    - name: tea-post
      service: tea-post-svc
      port: 80
    - name: tea
      service: tea-svc
      port: 80
    - name: coffee-v1
      service: coffee-v1-svc
      port: 80
    - name: coffee-v2
      service: coffee-v2-svc
      port: 80

```



```

routes:
- path: /tea
  authRequest: /auth/path
  authRequestSets:
    - key: foo
      value: bar
  matches:
    - conditions:
        - variable: $request_method
          value: POST
      action:
        pass: tea-post
    action:
      pass: tea-post
- path: /coffee
  matches:
    - conditions:
        - cookie: version
          value: v2
      action:
        pass: coffee-v2
    action:
      pass: coffee-v1
  authRequestLocations:
    - path: /auth/path
      proxyPass:
        upstreamName: "tea"
      proxyPassHeaders:
        - key: Content-Length
          value: "100"

```

Примените настройки:

```
$ kubectl create -f cafe-virtual-server.yaml
```

3. Протестируйте конфигурацию.

Проверьте, что конфигурация была успешно применена, просмотрев события ресурса VirtualServer:

```
$ kubectl describe virtualserver cafe
```

Вывод:

```

Events:
  Type    Reason             Age   From              Message
  ----    -
  Normal  AddedOrUpdated     2s    ANIC              Configuration for
→default/cafe was added or updated

```

4. Протестируйте доступ к сервису tea.

Отправьте POST-запрос и убедитесь, что ответ приходит от tea-post-svc:

```
$ curl --resolve cafe.example.com:$IC_HTTP_PORT:$IC_IP http://cafe.example.com:
→$IC_HTTP_PORT/tea -X POST
```

Ожидаемый результат:


```
Server address: 10.16.1.188:80
Server name: tea-post-b5dd479b4-6ssmh
. . .
```

Отправьте GET-запрос и убедитесь, что ответ приходит от `tea-svc`:

```
$ curl --resolve cafe.example.com:$IC_HTTP_PORT:$IC_IP http://cafe.example.com:
→$IC_HTTP_PORT/tea
```

Ожидаемый результат:

```
Server address: 10.16.1.189:80
Server name: tea-7d57856c44-2hsvr
. . .
```

5. Протестируйте доступ к сервису `coffee`.

Отправьте запрос с `cookie version=v2` и убедитесь, что ответ приходит от `coffee-v2-svc`:

```
$ curl --resolve cafe.example.com:$IC_HTTP_PORT:$IC_IP http://cafe.example.com:
→$IC_HTTP_PORT/coffee --cookie "version=v2"
```

Ожидаемый результат:

```
Server address: 10.16.1.187:80
Server name: coffee-v2-7fd446968b-vkthp
. . .
```

Отправьте запрос без `cookie` и убедитесь, что ответ приходит от `coffee-v1-svc`:

```
$ curl --resolve cafe.example.com:$IC_HTTP_PORT:$IC_IP http://cafe.example.com:
→$IC_HTTP_PORT/coffee
```

Ожидаемый результат:

```
Server address: 10.16.0.153:80
Server name: coffee-v1-78754bdcfb-bs9nh
. . .
```

5.10 Сохранение сессий

Часто необходимо, чтобы запросы от клиента всегда передавались одному и тому же контейнеру бэкенда. Вы можете включить такое поведение с помощью функции сохранения сессий, доступной в ANIC.

ANIC поддерживает метод `sticky cookie`. При использовании этого метода ANIC добавляет `session cookie` в первый ответ от бэкенда, идентифицируя контейнер, который отправил ответ. Когда клиент делает следующий запрос, он отправляет значение `cookie`, и ANIC направляет запрос в тот же контейнер.

5.10.1 Синтаксис

Чтобы включить сохранение сессий для одного или нескольких сервисов, нужно настроить блок `sessionCookie` в конфигурации апстрима для каждого сервиса. В аннотации указываются сервисы, для которых нужно включить сохранение сессий, а также различные параметры `cookie`. См. директиву `sticky` в конфигурации Angie.

5.10.2 Пример

В следующем примере мы включаем сохранение сессий для двух сервисов: `tea-svc` и `coffee-svc`:

```
apiVersion: k8s.angie.software/v1
kind: VirtualServer
metadata:
  name: cafe
spec:
  host: cafe.example.com
  tls:
    secret: cafe-secret
  upstreams:
  - name: tea
    service: tea-svc
    port: 80
    sessionCookie:
      enable: true
      name: srv_id
      path: /tea
      expires: 2h
  - name: coffee
    service: coffee-svc
    port: 80
    sessionCookie:
      enable: true
      name: srv_id
      path: /coffee
      expires: 1h
  routes:
  - path: /tea
    action:
      pass: tea
  - path: /coffee
    action:
      pass: coffee
```

Для обоих сервисов `sticky cookie` имеет одинаковое имя `srv_id`. Однако для каждого сервиса заданы разные значения времени жизни (`expires`) и пути (`path`).

Сохранение сессий продолжает работать даже в случае, если запущено несколько реплик ANIC.

5.11 Cert-manager

Ниже приведены примеры:

- разворачивания `cert-manager` и самоподписанного центра сертификации;
- разворачивания ANIC;
- разворачивания простого веб-приложения;
- настройки балансировки нагрузки для этого приложения с помощью ресурса `VirtualServer`.

5.11.1 Разворачивание `cert-manager` и самоподписанного центра сертификации

1. Разверните `cert-manager` и все необходимые зависимости:

```
$ kubectl apply -f https://github.com/cert-manager/cert-manager/releases/
→download/v1.8.0/cert-manager.yaml
```

2. Разверните Issuer (самоподписанный центр сертификации):

```
apiVersion: v1
kind: Namespace
metadata:
  name: sandbox
---
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: selfsigned-issuer
spec:
  selfSigned: {}
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: my-selfsigned-ca
  namespace: sandbox
spec:
  isCA: true
  commonName: my-selfsigned-ca
  secretName: root-secret
  privateKey:
    algorithm: ECDSA
    size: 256
  issuerRef:
    name: selfsigned-issuer
    kind: ClusterIssuer
    group: cert-manager.io
---
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: my-ca-issuer
  namespace: sandbox
spec:
  ca:
    secretName: root-secret
```


Примените настройки:

```
$ kubectl apply -f self-signed.yaml
```

5.11.2 Запуск примера

1. Установите ANIC. Включите поддержку `cert-manager` для ресурсов `VirtualServer`:

```
--enable-custom-resources --enable-cert-manager
```

2. Сохраните публичный IP-адрес ANIC в переменной оболочки:

```
$ IC_IP=<ваш_IP-адрес>
```

3. Сохраните HTTPS-порт ANIC в переменной оболочки:

```
$ IC_HTTPS_PORT=<номер порта>
```

4. Создайте Deployment и Service для `coffee` и `tea`:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: coffee
spec:
  replicas: 2
  selector:
    matchLabels:
      app: coffee
  template:
    metadata:
      labels:
        app: coffee
    spec:
      containers:
        - name: coffee
          image: angiesoftware/angie-hello:plain-text
          ports:
            - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: coffee-svc
spec:
  ports:
    - port: 80
      targetPort: 8080
      protocol: TCP
      name: http
  selector:
    app: coffee
---
apiVersion: apps/v1
kind: Deployment
metadata:
```



```

    name: tea
spec:
  replicas: 3
  selector:
    matchLabels:
      app: tea
  template:
    metadata:
      labels:
        app: tea
    spec:
      containers:
      - name: tea
        image: angiesoftware/angie-hello:plain-text
        ports:
        - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: tea-svc
  labels:
spec:
  ports:
  - port: 80
    targetPort: 8080
    protocol: TCP
    name: http
  selector:
    app: tea

```

Примените настройки:

```
$ kubectl create -f cafe.yaml
```

5. Создайте ресурс VirtualServer:

```

apiVersion: k8s.angie.software/v1
kind: VirtualServer
metadata:
  name: cafe
spec:
  host: cafe.example.com
  tls:
    secret: cafe-secret
    cert-manager:
      cluster-issuer: selfsigned-issuer
  upstreams:
  - name: tea
    service: tea-svc
    port: 80
  - name: coffee
    service: coffee-svc
    port: 80
  routes:
  - path: /tea
    action:
      pass: tea

```



```
- path: /coffee
  action:
    pass: coffee
```

Примените настройки:

```
$ kubectl create -f cafe-virtual-server.yaml
```

6. Протестируйте приложение.

Используйте `curl` для проверки сервисов `coffee` и `tea`: `--insecure`, чтобы отключить проверку сертификата, и `--resolve`, чтобы указать заголовок `Host` в запросе с `cafe.example.com`.

Запрос к сервису `coffee`:

```
$ curl --resolve cafe.example.com:$IC_HTTPS_PORT:$IC_IP https://cafe.example.com:
→$IC_HTTPS_PORT/coffee --insecure
```

Пример ответа:

```
Server address: 10.12.0.18:80
Server name: coffee-7586895968-r26zn
...
```

Запрос к сервису `tea`:

```
$ curl --resolve cafe.example.com:$IC_HTTPS_PORT:$IC_IP https://cafe.example.com:
→$IC_HTTPS_PORT/tea --insecure
```

Пример ответа:

```
Server address: 10.12.0.19:80
Server name: tea-7cd44fcb4d-xfw2x
...
```

5.12 gRPC

Для поддержки gRPC-приложений с помощью ресурсов `VirtualServer` необходимо добавить поле `type: grpc` в `upstream`. Если этот параметр не указан, по умолчанию будет использован протокол `http`.

5.12.1 Предварительная настройка

- Необходимо включить прослушиватель HTTP/2. См. `http2` в *ConfigMap*.
- Ресурсы `VirtualServer` и `VirtualServerRoute` для gRPC-приложений должны включать тер-минацию TLS.

5.12.2 Пример

```
apiVersion: k8s.angie.software/v1
kind: VirtualServer
metadata:
  name: grpc-vs
spec:
  host: grpc.example.com
  tls:
    secret: grpc-secret
  upstreams:
    - name: grpc1
      service: grpc-svc
      port: 50051
      type: grpc
  routes:
    - path: /helloworld.Greeter
      action:
        pass: grpc1
```

В этом примере `grpc-svc` — это сервис для gRPC-приложения. Он будет доступен по пути `/helloworld.Greeter`. Обратите внимание, что в конфигурации `upstream` используется поле `type: grpc`.

5.13 Ingress MTLS

Ниже приведен пример развертывания веб-приложения, настройки балансировки нагрузки с помощью `VirtualServer` и применения политики `Ingress MTLS`.

Примечание

Политика `Ingress MTLS` поддерживает настройку списка аннулированных сертификатов (CRL). Подробности см. *Использование списка отзыва сертификатов*.

5.13.1 Предварительные действия

1. Установите ANIC.
2. Сохраните публичный IP-адрес ANIC в переменной оболочки:

```
$ IC_IP=<ваш_IP-адрес>
```

3. Сохраните HTTPS-порт ANIC в переменной оболочки:

```
$ IC_HTTPS_PORT=<номер порта>
```


5.13.2 Настройка Ingress MTLS

1. Создайте Deployment и Service для приложения:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webapp
spec:
  replicas: 1
  selector:
    matchLabels:
      app: webapp
  template:
    metadata:
      labels:
        app: webapp
    spec:
      containers:
        - name: webapp
          image: angiesoftware/angie-hello:plain-text
          ports:
            - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: webapp-svc
spec:
  ports:
    - port: 80
      targetPort: 8080
      protocol: TCP
      name: http
  selector:
    app: webapp
```

Примените настройки:

```
$ kubectl apply -f webapp.yaml
```

2. Создайте секрет с именем `ingress-mtls-secret`, который будет использоваться для валидации Ingress MTLS:

```
kind: Secret
metadata:
  name: ingress-mtls-secret
apiVersion: v1
type: angie.software/ca
data:
  ca.crt: 
  LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSOtLS0tCk1JSUQvVENDQXVXZ0F3SUJBZ0lVSzdhbU14OFllLWwg1BVG51SkZETDlWS2
```

Примените настройки:

```
$ kubectl apply -f ingress-mtls-secret.yaml
```

3. Создайте политику с именем `ingress-mtls-policy`, которая ссылается на секрет из предыдущего шага:


```
apiVersion: k8s.angie.software/v1
kind: Policy
metadata:
  name: ingress-mtls-policy
spec:
  ingressMTLS:
    clientCertSecret: ingress-mtls-secret
    verifyClient: "on"
    verifyDepth: 1
```

Примените настройки:

```
$ kubectl apply -f ingress-mtls.yaml
```

4. Создайте секрет с TLS-сертификатом и ключом:

```
apiVersion: v1
kind: Secret
metadata:
  name: tls-secret
type: kubernetes.io/tls
data:
  tls.crt:   
→LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSB0tLS0tCk1JSURHeKNDQWdPZ0F3SUJBZ01VWU90ZXQ1cnpjd2pFM1o1QUQzQS9tdV  
  tls.key:   
→LS0tLS1CRUdJTiBQUk1WQVRFIETFWs0tLS0tCk1JSUV2UU1CQURBTk1na3Foa2lHOXcwQkFRRUZBQVNDQktjd2dnU2pBZ0
```

Примените настройки:

```
$ kubectl create -f tls-secret.yaml
```

5. Создайте ресурс VirtualServer для веб-приложения:

```
apiVersion: k8s.angie.software/v1
kind: VirtualServer
metadata:
  name: webapp
spec:
  host: webapp.example.com
  tls:
    secret: tls-secret
  policies:
    - name: ingress-mtls-policy
  upstreams:
    - name: webapp
      service: webapp-svc
      port: 80
  routes:
    - path: /
      action:
        pass: webapp
```

```
$ kubectl apply -f virtual-server.yaml
```

Примечание

VirtualServer должен ссылаться на политику `ingress-mtls-policy`, созданную на шаге 3.

6. Протестируйте конфигурацию.

Если вы попытаетесь обратиться к приложению без предоставления клиентского сертификата и ключа, ANIC отклонит запрос:

```
$ curl --insecure --resolve webapp.example.com:$IC_HTTPS_PORT:$IC_IP \
https://webapp.example.com:$IC_HTTPS_PORT/
```

Ожидаемый ответ:

```
<html>
<head><title>400 No required SSL certificate was sent</title></head>
<body>
<center><h1>400 Bad Request</h1></center>
<center>No required SSL certificate was sent</center>
<hr><center>Angie/1.8.1</center>
</body>
</html>
```

Если вы предоставите корректный клиентский сертификат и ключ, запрос выполнится успешно:

```
$ curl --insecure --resolve webapp.example.com:$IC_HTTPS_PORT:$IC_IP \
https://webapp.example.com:$IC_HTTPS_PORT/ --cert ./client-cert.pem --key ./
client-key.pem
```

Ожидаемый ответ:

```
Server address: 10.244.0.8:8080
Server name: webapp-7c6d448df9-9ts8x
Date: 23/Sep/2020:07:18:52 +0000
URI: /
Request ID: acb0f48057ccdfd250debe5afe58252a
```

5.14 JWKS

В этом примере:

- разворачивается веб-приложение;
- настраивается балансировка нагрузки с помощью `VirtualServer`;
- применяется политика JWT.

В отличие от примера с JWT, здесь внешний провайдер идентификации (IdP) определен с помощью поля `JwksURI`. В качестве провайдера используется `Keycloak`, развернутый как контейнер и доступный с помощью ANIC.

5.14.1 Предварительные действия


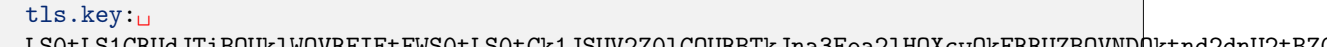
1. Установите ANIC.
2. Добавьте в файл `/etc/hosts` записи:

```
<ваш_IP-адрес> webapp.example.com
<ваш_IP-адрес> keycloak.example.com
```

Здесь `webapp.example.com` — домен веб-приложения, а `keycloak.example.com` — домен Keycloak.

5.14.2 Настройка JWKS

1. Создайте Secret с TLS-сертификатом и ключом для терминального TLS-шифрования в Keycloak:

```
apiVersion: v1
kind: Secret
metadata:
  name: tls-secret
type: kubernetes.io/tls
data:
  tls.crt: 
  tls.key: 
```

Примените настройки:

```
$ kubectl apply -f tls-secret.yaml
```

2. Создайте Deployment и Service для веб-приложения:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webapp
spec:
  replicas: 1
  selector:
    matchLabels:
      app: webapp
  template:
    metadata:
      labels:
        app: webapp
    spec:
      containers:
        - name: webapp
          image: angiesoftware/angie-hello:plain-text
          ports:
            - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: webapp-svc
spec:
  ports:
    - port: 80
      targetPort: 8080
      protocol: TCP
      name: http
  selector:
    app: webapp
```

Примените настройки:

```
$ kubectl apply -f webapp.yaml
```


3. Создайте Deployment и Service для Keycloak:

```
apiVersion: v1
kind: Service
metadata:
  name: keycloak
  labels:
    app: keycloak
spec:
  ports:
    - name: http
      port: 8080
      targetPort: 8080
  selector:
    app: keycloak
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: keycloak
  namespace: default
  labels:
    app: keycloak
spec:
  replicas: 1
  selector:
    matchLabels:
      app: keycloak
  template:
    metadata:
      labels:
        app: keycloak
    spec:
      containers:
        - name: keycloak
          image: quay.io/keycloak/keycloak:20.0.1
          args: ["start-dev"]
          env:
            - name: KEYCLOAK_ADMIN
              value: "admin"
            - name: KEYCLOAK_ADMIN_PASSWORD
              value: "admin"
            - name: KC_PROXY
              value: "edge"
          ports:
            - name: http
              containerPort: 8080
            - name: https
              containerPort: 8443
          readinessProbe:
            httpGet:
              path: /realms/master
              port: 8080
```

Примените настройки:

```
$ kubectl apply -f keycloak.yaml
```

4. Создайте VirtualServer для Keycloak:


```
apiVersion: k8s.angie.software/v1
kind: VirtualServer
metadata:
  name: keycloak
spec:
  host: keycloak.example.com
  tls:
    secret: tls-secret
    redirect:
      enable: true
  upstreams:
    - name: keycloak
      service: keycloak
      port: 8080
  routes:
    - path: /
      action:
        pass: keycloak
```

Примените настройки:

```
$ kubectl apply -f virtual-server-idp.yaml
```

5. Настройте Keycloak:

- Перейдите в Keycloak: <https://keycloak.example.com>.
- Создайте новую область Realm с именем `jwtks-example`.
- Перейдите во вкладку **Client**, создайте новый клиент с именем `jwtks-client` и включите для него **Client authentication** и **Authorization**.
- Перейдите во вкладку **Credentials** и скопируйте секрет клиента.

Сохраните секрет:

```
export SECRET=<client secret>
```

- Перейдите во вкладку **Users** и создайте пользователя `jwtks-user`.
- Перейдите во вкладку **Credentials** этого пользователя и установите пароль. Для примера подойдет любой пароль.

Сохраните пароль:

```
export PASSWORD=<user password>
```

6. Создайте политику `jwt-policy` с указанием `JwksURI` и настройте поле `JwksURI` так, чтобы оно разрешало только те запросы к веб-приложению, которые содержат действительный JWT.

В приведенной ниже политике замените область `jwtks-example` на область (`realm`), созданную вами. Значение `spec.jwt.token` в примере установлено в `$http_token`, так как токен клиента передается в заголовке HTTP.

```
apiVersion: k8s.angie.software/v1
kind: Policy
metadata:
  name: jwt-policy
spec:
  jwt:
    realm: MyProductAPI
    token: $http_token
```



```
jwksURI: http://keycloak.default.svc.cluster.local:8080/realms/jwks-example/
→protocol/openid-connect/certs
```

Примените политику:

```
$ kubectl apply -f jwks.yaml
```

7. Разверните ConfigMap с резолвером.

Если в `jwksURI` используется имя хоста, необходимо настроить `resolver`. Для этого необходимо добавить поле `resolver-addresses`.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: angie-config
  namespace: angie-ingress
data:
  resolver-addresses: kube-dns.kube-system.svc.cluster.local
  http-snippets: |
    subrequest_output_buffer_size 64k;
```

В этом примере мы создаем ConfigMap, используя стандартный DNS Kubernetes `kube-dns.kube-system.svc.cluster.local` в качестве адреса резолвера. Дополнительную информацию о `resolver-addresses` и других связанных ключах ConfigMap можно посмотреть в разделе *ConfigMap*.

Примечание

При установке значения `jwksURI` ответ может отличаться в зависимости от используемого IDP. В некоторых случаях ответ может быть слишком большим для корректной обработки Angie. Если это произойдет, необходимо настроить директиву `subrequest_output_buffer_size` в контексте `http`. Это можно сделать с помощью `http-snippets`. Значение `subrequest_output_buffer_size` указано только для примера и должно быть изменено в соответствии с вашей средой.

Примените конфигурацию:

```
$ kubectl apply -f angie-config.yaml
```

8. Настройте балансировку нагрузки. Создайте VirtualServer для веб-приложения:

```
apiVersion: k8s.angie.software/v1
kind: VirtualServer
metadata:
  name: webapp
spec:
  host: webapp.example.com
  policies:
    - name: jwt-policy
  upstreams:
    - name: webapp
      service: webapp-svc
      port: 80
  routes:
    - path: /
      action:
        pass: webapp
```


Примените настройки:

```
$ kubectl apply -f virtual-server.yaml
```

Обратите внимание, что `VirtualServer` ссылается на политику `jwt-policy`, созданную выше.

9. Получите токен клиента.

Для доступа к веб-приложению клиент должен передавать токен-носитель. Чтобы получить токен, выполните команду:

```
$ export TOKEN=$(curl -k -L -X POST 'https://keycloak.example.com/realms/jwks-
→example/protocol/openid-connect/token' \
-H 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode grant_type=password \
--data-urlencode scope=openid \
--data-urlencode client_id=jwks-client \
--data-urlencode client_secret=$SECRET \
--data-urlencode username=jwks-user \
--data-urlencode password=$PASSWORD \
| jq -r .access_token)
```

Эта команда сохранит токен в переменной окружения `TOKEN`.

10. Протестируйте конфигурацию.

- Попробуйте запрос без токена-носителя:

```
$ curl -H 'Accept: application/json' webapp.example.com
```

Ответ:

```
<html>
<head><title>401 Authorization Required</title></head>
<body>
<center><h1>401 Authorization Required</h1></center>
<hr><center>Angie/1.8.1</center>
</body>
</html>
```

- Запрос с корректным токеном-носителем:

```
$ curl -H 'Accept: application/json' -H "token: ${TOKEN}" webapp.example.com
```

Ответ сервера:

```
Server address: 10.42.0.7:8080
Server name: webapp-5c6fdbcbf9-pt9tp
Date: 13/Dec/2022:14:50:33 +0000
URI: /
Request ID: f1241390ac51318afa4fcc39d2341359
```


5.15 JWT

В примере ниже разворачивается веб-приложение, настраивается балансировка нагрузки с помощью VirtualServer и применяется политика JWT.

5.15.1 Предварительные действия

1. Установите ANIC.
2. Сохраните публичный IP-адрес ANIC в переменной оболочки:

```
$ IC_IP=<ваш_IP-адрес>
```

3. Сохраните HTTP-порт ANIC в переменной оболочки:

```
$ IC_HTTP_PORT=<номер порта>
```

5.15.2 Настройка JWT

1. Создайте Deployment и Service для приложения:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webapp
spec:
  replicas: 1
  selector:
    matchLabels:
      app: webapp
  template:
    metadata:
      labels:
        app: webapp
    spec:
      containers:
        - name: webapp
          image: angiesoftware/angie-hello:plain-text
          ports:
            - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: webapp-svc
spec:
  ports:
    - port: 80
      targetPort: 8080
      protocol: TCP
      name: http
  selector:
    app: webapp
```

Примените настройки:


```
$ kubectl apply -f webapp.yaml
```

- Создайте секрет с именем `jwk-secret`, который будет использоваться для проверки JWT:

```
apiVersion: v1
kind: Secret
metadata:
  name: jwk-secret
type: angie.software/jwk
data:
  jwk: eyJrZXlzlIjoKICAgIFt7CiAgICAgICAgIm5iOiJabUZ1ZEdGemRHbGphbmQwIiwKICAgICAgICAia3R5Ijoib2N0IiwKI
```

Примените настройки:

```
$ kubectl apply -f jwk-secret.yaml
```

- Создайте политику `jwt-policy`, которая будет ссылаться на `Secret` из предыдущего шага и разрешать запросы к веб-приложению только при наличии корректного JWT:

```
apiVersion: k8s.angie.software/v1
kind: Policy
metadata:
  name: jwt-policy
spec:
  jwt:
    realm: MyProductAPI
    secret: jwk-secret
    token: $http_token
```

Примените настройки:

```
$ kubectl apply -f jwt.yaml
```

- Настройте балансировку нагрузки. Создайте ресурс `VirtualServer` для веб-приложения:

```
apiVersion: k8s.angie.software/v1
kind: VirtualServer
metadata:
  name: webapp
spec:
  host: webapp.example.com
  policies:
    - name: jwt-policy
  upstreams:
    - name: webapp
      service: webapp-svc
      port: 80
  routes:
    - path: /
      action:
        pass: webapp
```

Примените настройки:

```
$ kubectl apply -f virtual-server.yaml
```

Обратите внимание, что `VirtualServer` ссылается на политику `jwt-policy`, созданную на предыдущем шаге.

5. Протестируйте конфигурацию.

Если попытаться получить доступ к приложению без JWT, ANIC отклонит запрос:

```
$ curl --resolve webapp.example.com:$IC_HTTP_PORT:$IC_IP http://webapp.example.
→com:$IC_HTTP_PORT/
```

Ответ:

```
<html>
<head><title>401 Authorization Required</title></head>
<body>
<center><h1>401 Authorization Required</h1></center>
<hr><center>Angie/1.8.1</center>
</body>
</html>
```

Если передать корректный JWT, запрос будет выполнен успешно:

```
$ curl --resolve webapp.example.com:$IC_HTTP_PORT:$IC_IP http://webapp.example.
→com:$IC_HTTP_PORT/ -H "token: `cat token.jwt`"
```

Ответ:

```
Server address: 172.17.0.3:8080
Server name: webapp-7c6d448df9-lcrx6
Date: 10/Sep/2020:18:20:03 +0000
URI: /
Request ID: db2c07ce640755ccbe9f666d16f85620
```

5.16 OIDC

OIDC (OpenID Connect) обеспечивает удобную аутентификацию пользователей через внешнего провайдера, используя безопасные токены для управления доступом в системе.

Политика OIDC настраивает ANIC как клиент (relying party) для аутентификации через OpenID Connect.

Например, следующая конфигурация использует `clientID myclient` и `clientSecret oidc-secret` для аутентификации через провайдера OpenID Connect `https://idp.example.com`:

```
oidc:
  clientID: myclient
  clientSecret: oidc-secret
  authEndpoint: https://idp.example.com/openid-connect/auth
  jwksURI: https://idp.example.com/openid-connect/certs
  tokenEndpoint: https://idp.example.com/openid-connect/token
  scope: openid+profile+email
  accessTokenEnable: true
```

Подробное описание параметров можно посмотреть [здесь](#).

5.16.1 Настройка аутентификации через OpenID Connect

Чтобы настроить аутентификацию через OpenID Connect, выполните следующие шаги:

1. Задайте аргумент командной строки `enable-oidc=true`.

В конфиге будут подключены три модуля:

- `load_module modules/nginx_http_js_module.so;`
- `load_module modules/nginx_http_auth_jwt_module.so;`
- `load_module modules/nginx_http_keyval_module.so;`

2. Добавьте секрет с ключом клиента. Ключ должен быть закодирован в Base64:

Список 1: `client-secret.yaml`

```
apiVersion: v1
kind: Secret
metadata:
  name: oidc-secret
type: angie.software/oidc
data:
  client-secret: <client_secret>
```

3. Примените секрет:

```
kubectl apply -f oidc/client-secret.yaml
```

4. Добавьте политику:

Список 2: `oidc.yaml`

```
apiVersion: k8s.angie.software/v1
kind: Policy
metadata:
  name: oidc-policy
spec:
  oidc:
    clientID: myclient
    clientSecret: oidc-secret
    authEndpoint: https://idp.example.com/openid-connect/auth
    jwksURI: https://idp.example.com/openid-connect/certs
    tokenEndpoint: https://idp.example.com/openid-connect/token
    scope: openid+profile+email
    accessTokenEnable: true
```

5. Примените политику:

```
kubectl apply -f oidc/oidc.yaml
```

6. После того как секрет и описание политики будут добавлены, примените политику для сервера или маршрута, сославшись на нее:

```
policies:
- name: oidc-policy
```

На данном этапе включение политики вызовет ошибку, т.к. не были заданы обязательные переменные, обеспечивающие валидацию токенов в процессе аутентификации OIDC:

- `$jwt_claim_iat`

- \$jwt_claim_iss
- \$jwt_claim_sub
- \$jwt_claim_aud

7. В спецификации VirtualServer задайте обязательные переменные в параметре *maps*:

```
maps:
- variable: $jwt_claim_iat
  source: $oidc_client
  parameters:
    - value: 'myclient'
      result: '80'
- variable: $jwt_claim_iss
  source: $oidc_client
  parameters:
    - value: 'myclient'
      result: 'PROVIDER_URL'
- variable: $jwt_claim_sub
  source: $oidc_client
  parameters:
    - value: 'myclient'
      result: 'myclient'
- variable: $jwt_claim_aud
  source: $oidc_client
  parameters:
    - value: 'myclient'
      result: 'myclient'
```

В конфигурацию VirtualServer будут добавлены следующие директивы:

```
map $oidc_client $jwt_claim_iat {
    myclient 80;
}

map $oidc_client $jwt_claim_iss {
    myclient PROVIDER_URL;
}

map $oidc_client $jwt_claim_sub {
    myclient myclient;
}

map $oidc_client $jwt_claim_aud {
    myclient myclient;
}
```

5.16.2 Полный пример конфигурации

```
apiVersion: k8s.angie.software/v1
kind: VirtualServer
metadata:
  name: test-echo
spec:
  host: test.example.com
  upstreams:
    - name: app-server-payload
```



```

    service: echoserver
    port: 8077
  routes:
    - path: /
      action:
        proxy:
          upstream: app-server-payload
      policies:
        - name: oidc-policy
  maps:
    - variable: $jwt_claim_iat
      source: $oidc_client
      parameters:
        - value: 'myclient'
          result: '80'
    - variable: $jwt_claim_iss
      source: $oidc_client
      parameters:
        - value: 'myclient'
          result: 'PROVIDER_URL'
    - variable: $jwt_claim_sub
      source: $oidc_client
      parameters:
        - value: 'myclient'
          result: 'myclient'
    - variable: $jwt_claim_aud
      source: $oidc_client
      parameters:
        - value: 'myclient'
          result: 'myclient'

```

5.16.3 Пример включения переменных map в зависимости от входного значения

Значения:

- default
- volatile
- include
- hostnames

```

maps:
  - variable: $result_var
    source: $host
    parameters:
      - value: 'default'
        result: 'default_value'
      - value: 'volatile'
        result: ''
      - value: 'include'
        result: '/dev/stdout'
      - value: 'example.com'
        result: '1'
      - value: '*.example.com'
        result: '1'

```


См. также директиву `map` в документации Angie.

5.17 TLS Passthrough

Функция TLS Passthrough позволяет ANIC принимать TLS-соединения на порту 443 и направлять их на соответствующие серверы-бэкенды без расшифровки. Маршрутизация осуществляется на основе SNI (Server Name Indication), что позволяет клиентам указывать имя сервера (например, `example.com`) во время SSL-рукопожатия. При этом ANIC продолжает обрабатывать обычный HTTPS-трафик на том же порту 443, завершая TLS-соединения с использованием сертификатов и ключей, заданных в ресурсах `Ingress` или `VirtualServer`.

Ниже приведен пример использования ресурса `TransportServer` для настройки балансировки нагрузки в режиме TLS Passthrough. В примере будет развернуто бэкенд-приложение (Secure App), прослушивающее TLS-трафик на порту 8443. ANIC будет маршрутизировать соединения к этому приложению с помощью `TransportServer`.

5.17.1 О Secure App

Secure App — это под с Angie (не путать с подом ANIC, который также использует Angie), настроенный на обслуживание HTTPS-трафика на порту 8443 для хоста `app.example.com`. Для терминации TLS используются самоподписанный TLS-сертификат и ключ. Приложение отвечает на HTTPS-запросы клиентов простым текстовым сообщением:

```
hello from pod <имя пода>
```

5.17.2 Предварительные действия

1. Установите ANIC.
2. Убедитесь, что развернуто определение пользовательского ресурса для `TransportServer`.
3. Запустите ANIC с параметрами `-enable-custom-resources` и `-enable-tls-passthrough`, чтобы включить поддержку TLS Passthrough.
4. Сохраните публичный IP-адрес ANIC в переменной оболочки:

```
$ IC_IP=<ваш_IP-адрес>
```

5. Сохраните HTTPS-порт ANIC в переменной оболочки:

```
$ IC_HTTPS_PORT=<номер порта>
```

5.17.3 Настройка TLS Passthrough

1. Разверните Secure App:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: secure-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: secure-app
```



```

template:
  metadata:
    labels:
      app: secure-app
  spec:
    containers:
      - name: secure-app
        image: angiesoftware/angie-hello:plain-text
        ports:
          - containerPort: 8443
        volumeMounts:
          - name: secret
            mountPath: /etc/angie/ssl
            readOnly: true
          - name: config-volume
            mountPath: /etc/angie/conf.d
    volumes:
      - name: secret
        secret:
          secretName: app-tls-secret
      - name: config-volume
        configMap:
          name: secure-config
---
apiVersion: v1
kind: Service
metadata:
  name: secure-app
spec:
  ports:
    - port: 8443
      targetPort: 8443
      protocol: TCP
      name: https
  selector:
    app: secure-app
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: secure-config
data:
  app.conf: |-
    server {
      listen 8443 ssl;
      listen [::]:8443 ssl;

      server_name app.example.com;

      ssl_certificate /etc/angie/ssl/tls.crt;
      ssl_certificate_key /etc/angie/ssl/tls.key;

      default_type text/plain;

      location / {
        return 200 "hello from pod $hostname\n";
      }

```



```

    }
  ---
  apiVersion: v1
  kind: Secret
  metadata:
    name: app-tls-secret
  data:
    tls.crt: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSOtLS0tCk1JSURGRENDQWZ3QONRQ3EzQWxhdnJiaWpqQU5CZ2txaGtpRz13MEJBUE
    tls.key: LS0tLS1CRUdJTiBQUklWQVRFIEtFWS0tLS0tCk1JSUV2Z01CQURBTk1na3Foa2lHOXcwQkFRRUZBQVNDQktnd2dnU2tBZQ

```

Примените настройки:

```
$ kubectl apply -f secure-app.yaml
```

2. Настройте балансировку нагрузки.

Создайте ресурс TransportServer:

```

apiVersion: k8s.angie.software/v1alpha1
kind: TransportServer
metadata:
  name: secure-app
spec:
  listener:
    name: tls-passthrough
    protocol: TLS_PASSTHROUGH
    host: app.example.com
    upstreams:
      - name: secure-app
        service: secure-app
        port: 8443
  action:
    pass: secure-app

```

Примените настройки:

```
$ kubectl apply -f transport-server-passthrough.yaml
```

3. Проверьте успешность применения конфигурации, просмотрев события TransportServer:

```
$ kubectl describe ts secure-app
```

Вывод команды может выглядеть так:

```

Events:
  Type    Reason             Age   From    Message
  ----    -
  Normal  AddedOrUpdated     9s    anic    Configuration for
default/secure-app was added or updated

```

4. Проверьте доступ к Secure App с помощью curl. Используйте параметр `--insecure`, чтобы отключить проверку сертификатов (так как используется самоподписанный сертификат), а также `--resolve`, чтобы указать IP-адрес и HTTPS-порт ANIC для домена `app.example.com`:

```

$ curl --resolve app.example.com:$IC_HTTPS_PORT:$IC_IP https://app.example.com:
$IC_HTTPS_PORT --insecure

```


Ожидаемый ответ:

```
hello from pod secure-app-d986bcf6b-jwm2s
```


ГЛАВА 6

Общие примеры

В этом разделе собраны примеры настройки и типовые конфигурации для общих примеров.

Пользовательские шаблоны

Пользовательский формат журнала (log-format)

Протокол PROXY

Wildcard-сертификат

Пример default-server-secret

6.1 Пользовательские шаблоны

Для изменения конфигурации Angie для ресурсов Ingress, ресурсов VirtualServer и основного файла конфигурации Angie можно использовать пользовательские шаблоны.

Пользовательские шаблоны ANIC настраиваются через *ConfigMap* с помощью следующих ключей:

- **main-template** — задает основной шаблон конфигурации Angie.
- **ingress-template** — задает шаблон конфигурации ANIC для ресурса Ingress.
- **virtualserver-template** — задает шаблон конфигурации ANIC для ресурса VirtualServer.

6.1.1 Пример

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: angie-config
  namespace: angie-ingress
data:
  main-template: |
    worker_processes {{.WorkerProcesses}};
    ...
    include /etc/angie/conf.d/*.conf;
  }
```



```
ingress-template: |
    {{range $upstream := .Upstreams}}
    upstream {{$upstream.Name}} {
        {{if $upstream.LBMethod }}{{$upstream.LBMethod}};{{end}}
        ...
    }{{end}}
virtualserver-template: |
    {{ range $u := .Upstreams }}
    upstream {{$u.Name}} {
        {{ if ne $u.UpstreamZoneSize "0" }}zone {{$u.Name}} {{$u.UpstreamZoneSize }};
        ...
    }
    {{ end }}
```

Примечание

- Шаблоны в примере обрезаны для краткости.
- Основные шаблоны `angie.tpl` и `angie.ingress.tpl` находятся по пути `internal/configs/version1`. Шаблон `VirtualServer` (`angie.virtualserver.tpl`) расположен по пути `internal/configs/version2`.

6.1.2 Диагностика ошибок

- Если пользовательский шаблон содержит ошибку, ANIC не запустится. Ошибка отобразится в журнале.

Пример ошибки:

```
Error updating Angie main template: template: angieTemplate:98: unexpected EOF
```

- Если некорректный пользовательский шаблон был добавлен после запуска ANIC, конфигурация не обновится. Ошибка отобразится в журнале, в событии, связанном с `ConfigMap`.

Пример ошибки:

```
Error when updating config from ConfigMap: Invalid angie configuration detected,
↳ not reloading
```

События `ConfigMap` можно просмотреть с помощью команды `kubectl describe -n anic configmap angie-config`.

Пример события с ошибкой:

```
Events:
  Type      Reason              Age             From             Message
  ----      -
  Normal    Updated             12s (x2 over 25s)  anic             Configuration from anic/
↳ angie-config was updated
  Warning   UpdatedWithError    10s             anic             Configuration from anic/
↳ angie-config was updated, but not applied: Error when parsing the main template:
↳ template: angieTemplate:98: unexpected EOF
  Warning   UpdatedWithError     8s             anic             Configuration from anic/
↳ angie-config was updated, but not applied: Error when writing main Config
```


6.2 Пользовательский формат журнала

Вы можете настроить пользовательский формат журнала (log-format) с помощью ресурса ConfigMap:

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: angie-config
  namespace: angie-ingress
data:
  log-format: '$remote_addr - $remote_user [$time_local] "$request" $status $grpc_
→status $body_bytes_sent "$http_referer" "$http_user_agent" "$http_x_forwarded_for"
→"$resource_name" "$resource_type" "$resource_namespace" "$service"'
```

В дополнение к встроенным переменным Angie можно использовать переменные, которые настраиваются в ANIC:

- `$resource_type` — тип ресурса Kubernetes, который обработал запрос клиента.
- `$resource_name` — имя ресурса Kubernetes, который обработал запрос клиента.
- `$resource_namespace` — пространство имен (namespace), в котором находится ресурс.
- `$service` — имя сервиса, на который был направлен клиентский запрос.
- `$grpc_status` — код статуса gRPC (при нормальной работе берется из трейлера HTTP/2 (grpc_status), возвращаемого бэкендом, при некоторых ошибках — из заголовка HTTP/2 (grpc_status), установленного бэкендом или Angie).

Примечание

Эти переменные доступны только для ресурсов Ingress, VirtualServer и VirtualServerRoute.

6.3 Протокол PROXY

Прокси-серверы и балансировщики нагрузки (например HAProxy или AWS ELB) могут передавать информацию о клиенте (IP-адрес и порт) следующему прокси или балансировщику с помощью протокола PROXY. Чтобы ANIC мог получить эту информацию, в ресурсе ConfigMap необходимо задать следующие параметры:

- `proxy-protocol` — должен быть установлен в значение `True`;
- `real-ip-header` — должен быть установлен в значение `proxy_protocol`;
- `set-real-ip-from` — IP-адрес или подсеть прокси или балансировщика, откуда будет приходить информация (подробнее см. `set_real_ip_from`).

Примечание

Протокол PROXY будет применяться ко всем ресурсам Ingress и VirtualServer. Ресурс TransportServer поддерживает протокол PROXY только при включенном TLS Passthrough.

6.3.1 Пример конфигурации

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: angie-config
data:
  proxy-protocol: "True"
  real-ip-header: "proxy_protocol"
  set-real-ip-from: "192.168.0.0/16"
```

В приведенном примере протокол PROXY настраивается через ресурс ConfigMap. Параметр `set-real-ip-from` установлен в `192.168.0.0/16` — это CIDR-подсеть прокси, стоящего перед ANIC в этом примере. Если вы хотите доверять всем IP-адресам, можно задать значение `0.0.0.0/0`. После создания ресурса ConfigMap IP-адрес клиента будет доступен через переменную `$remote_addr` в конфигурации Angie.

По умолчанию ANIC запоминает значение `$remote_addr` и передает его бэкенду в заголовке X-Real-IP. Стандартный формат журнала Angie по умолчанию: `'$remote_addr - $remote_user [$time_local] "$request" $status $body_bytes_sent "$http_referer" "$http_user_agent"'`.

6.4 Wildcard-сертификат

Wildcard-сертификат упрощает настройку терминирования TLS, если необходимо использовать один и тот же TLS-сертификат в нескольких ресурсах Ingress и VirtualServer в разных пространствах имен. Wildcard-сертификат позволяет централизованно управлять TLS-сертификатами для множества доменов без дублирования секрета в каждом ресурсе.

Обычно такой сертификат выдается для поддомена (например, `*.example.com`), а хосты ресурсов Ingress и VirtualServer включают этот поддомен (например `foo.example.com`, `bar.example.com`).

Важно

Предварительно необходимо запустить ANIC с аргументом командной строки `-wildcard-tls-secret` со значением TLS-секрета, содержащего wildcard-сертификат и ключ. ANIC поддерживает только один wildcard-секрет.

```
-wildcard-tls-secret=angie-ingress/wildcard-tls-secret
```

6.4.1 Пример

В примерах ниже показана настройка терминирования TLS для ресурса Ingress с хостом `foo.example.com` и ресурса VirtualServer с хостом `bar.example.com`.

Пример для Ingress из пространства имен `foo`:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: foo
  namespace: foo
spec:
  ingressClassName: angie
  tls:
  - hosts:
    - foo.example.com
```



```
rules:
- host: foo.example.com
  http:
    paths:
    - path: /
      pathType: Prefix
      backend:
        service:
          name: foo-service
          port:
            number: 80
```

Пример для VirtualServer из пространства имен **bar**:

```
apiVersion: k8s.angie.software/v1
kind: VirtualServer
metadata:
  name: bar
  namespace: bar
spec:
  host: bar.example.com
  tls:
    secret: ""
  upstreams:
  - name: bar
    service: bar-service
    port: 80
  routes:
  - path: /
    action:
      pass: bar
```

В примерах выше конкретный TLS-секрет не указан:

- в ресурсе Ingress отсутствует поле **secret** в блоке **tls**;
- в ресурсе VirtualServer поле **secret** пустое (**""**).

В этом случае будет использоваться wildcard-секрет, указанный с помощью параметра **-wildcard-tls-secret** при запуске ANIC.

6.5 Пример default-server-secret

Ниже приведен пример **default-server-secret**:

```
apiVersion: v1
kind: Secret
metadata:
  name: default-server-secret
  namespace: angie-ingress
type: kubernetes.io/tls
data:
  tls.crt: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSB0tLS0tCk1JSUN2akNDQWFZQ0NRREFPRj10THNhWFhEU5CZ2txaGtpRz13MEJBUXNGQUU...
  tls.key: LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSU1FcEFJQkFBS0NBUEVBdi91RWM4b1JkMHUvZXVJTjHNFK1RYZUp...
  ↪LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSB0tLS0tCk1JSUN2akNDQWFZQ0NRREFPRj10THNhWFhEU5CZ2txaGtpRz13MEJBUXNGQUU...
  ↪LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSU1FcEFJQkFBS0NBUEVBdi91RWM4b1JkMHUvZXVJTjHNFK1RYZUp...
```


ГЛАВА 7

Известные проблемы и решения

7.1 Ошибка "proxy_busy_buffers_size" must be less than the size of all "proxy_buffers" minus one buffer

Сообщение указывает на ошибку в конфигурации из-за неправильного соотношения значений параметров `proxy_buffer_size` и `proxy_buffers`.

Параметр	Описание	Значение по умолчанию
<code>proxy_buffer_size</code>	Задаёт размер основного буфера для обработки данных. Значение <code>proxy_buffer_size</code> может быть не более чем в два раза больше размера <code>proxy_buffers</code> .	4k 8k
<code>proxy_buffers</code>	Задаёт количество и размер дополнительных буферов. При значении 8 8k общий размер будет равен $8 \times 8 = 64k$.	8 4k 8k
<code>proxy_busy_buffers</code>	Ограничивает суммарный размер буферов, которые могут быть заняты для отправки ответа. Значение должно быть меньше, чем общий размер всех <code>proxy_buffers</code> минус размер одного буфера. Если значение <code>proxy_buffers</code> равно 8 8k, то значение <code>proxy_busy_buffers_size</code> должно быть меньше, чем $8 \times 8 - 8$, т.е. меньше 56k.	8k 16k

Ошибка может возникнуть, если значение `proxy_buffer_size` было изменено с помощью аннотации на большее и, таким образом, было нарушено соотношение параметров. Для решения проблемы необходимо с помощью аннотации установить размер для `proxy_buffers`.

Например:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    angie.software/proxy-buffers: '8 32k'
    angie.software/proxy-buffer-size: '64k'
```



```
name: test-echo
namespace: echoserver
spec:
  ingressClassName: angie
  rules:
  - host: test.example.com
    http:
      paths:
      - backend:
          service:
            name: echoserver
            port:
              number: 8077
          pathType: ImplementationSpecific
```


ГЛАВА 8

Права на интеллектуальную собственность

Документация на программный продукт Angie Ingress Controller (ANIC) является интеллектуальной собственностью ООО «Веб-Сервер».

Copyright © 2025, ООО «Веб-Сервер». Все права защищены.