ANIC

Версия 1.0.0

Описание функциональных характеристик

сент. 11, 2023

Оглавление

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.

1.1 Общие сведения

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

ANIC развертывается 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	3.17	x86_64, arm64
Alt Linux	10	x86_64, arm64
Debian	11	x86_64, arm64

ГЛАВА 2

Как работает ANIC

Этот документ объясняет, как работает Angie Ingress Controller (ANIC).

Мы предполагаем, что читатель знаком с основными концепциями Kubernetes, такими как Pod, Deployment, Service и Endpoints.

2.1 Что такое Ingress Controller?

Ingress Controller - это компонент в кластере Kubernetes, который настраивает балансировщик нагрузки HTTP в соответствии с ресурсами Ingress, созданными пользователем кластера.

> Чтобы узнать больше о ресурсах Ingress, обратитесь к официальной документации Kubernetes.

Этот документ относится конкретно к Angie Ingress Controller, также называемому *Ingress Controller* или *ANIC*, который построен на возможностях Angie.

2.2 Ingress Controller на высоком уровне

Давайте начнем с общего изучения ANIC. Рассмотрим пример того, как ANIC предоставляет клиентам в Интернете доступ к двум веб-приложениям, запущенным в кластере Kubernetes.

В схеме участвует:

- Кластер *Kubernetes*.
- Пользователи кластера: *администратор*, *пользователь A* и *пользователь B*, которые используют кластер через *API Kubernetes*.
- *Клиенты A* и *клиенты B*, которые подключаются к *приложениям A* и *приложениям B*, развернутым соответствующими пользователями.
- *ANIC*, развернутый администратором в Pod в пространстве имен *angie-ingress* и настроенный с помощью ресурса *ConfigMap angie-ingress*. Для простоты мы изобразили только один Pod ANIC, однако *администратор* обычно развертывает по крайней мере два Pod для обеспечения избыточности. *ANIC* использует *API Kubernetes* для получения последних ресурсов Ingress, созданных в кластере, а затем настраивает *Angie* в соответствии с этими ресурсами.

- *Приложение A* с двумя Pod, развернутыми в *пространстве имен A пользователем A*. Чтобы предоставить доступ приложению к его клиентам (*клиентам A*) через хост `a.example.com` , *пользователь A* создает *ресурс Ingress A*.
- *Приложение B* с одним Pod, развернутым в *пространстве имен B пользователем B*. Чтобы предоставить доступ к приложению его клиентам (*клиентам B*) через хост `b.example.com`, *пользователь B* создает *ресурс VirtualServer B*.
- *Общедоступная конечная точка*, которая находится перед Pod-ами *ANIC*. Обычно это TCP-балансировщик нагрузки (облачный, программный или аппаратный) или комбинация такого балансировщика нагрузки с сервисом NodePort. *Клиенты A* и *клиенты B* подключаются к своим приложениям через *общедоступную конечную точку*.

Для простоты не учтены многие необходимые ресурсы Kubernetes, такие как Deployment и Service, которые администратору и пользователям также необходимо создать.

Далее исследуем Pod ANIC.

2.3 Pod Ingress Controller

Pod ANIC состоит из одного контейнера, который, в свою очередь, включает в себя следующее:

- *Процесс ANIC*, который настраивает Angie в соответствии с Ingress и другими ресурсами, созданными в кластере.
- *Главный процесс Angie*, который управляет рабочими процессами Angie.
- *Рабочие процессы Angie*, которые обрабатывают клиентский трафик и балансируют нагрузку на серверные приложения.

В приведенном ниже нумерованном списке описано каждое соединение с указанием его типа в фигурных скобках:

1. (HTTP) *Prometheus* извлекает метрики ANIC и Angie через конечную точку HTTP, которую предоставляет *ANIC*.

   Примечание: *Prometheus* не требуется для ANIC, и эту конечную точку можно отключить.

2. (HTTPS) *ANIC* обращается к *API Kubernetes*, чтобы получить последние версии ресурсов в кластере, и выполняет запись в API для обновления статусов обрабатываемых ресурсов и выдачи событий.
3. (HTTP) *Kubelet* проверяет готовность *ANIC* (значение по умолчанию `: 8081/angie-ready`), чтобы определить готовность Pod ANIC.
4. (Файловый ввод-вывод) Когда *ANIC* запускается, он считывает из файловой системы *шаблоны конфигурации*, необходимые для генерации конфигурации. Шаблоны расположены в каталоге `/` контейнера и имеют расширение `.tmpl`.
5. (Файловый ввод-вывод) *ANIC* записывает в свои потоки *stdout* и *stderr* журналы, которые собираются средой выполнения контейнеров.
6. (Файловый ввод-вывод) *ANIC* генерирует *конфигурацию* Angie на основе ресурсов, созданных в кластере и записывает ее в файловую систему в папке `/etc/angie`. Файлы конфигурации имеют расширение `.conf`.
7. (Файловый ввод-вывод) *ANIC* записывает *TLS-сертификаты* и *ключи* из всех секретов TLS, на которые ссылаются ресурсы Ingress и другие ресурсы, в файловую систему.
8. (HTTP) *ANIC* извлекает метрики Angie через UNIX-сокет `unix:/var/lib/angie/angie-status.socket` и преобразует их в формат Prometheus, используемый в #1.

9. (HTTP) Чтобы убедиться в успешности перезагрузки конфигурации, *ANIC* проверяет, что по крайней мере у одного *рабочего процесса Angie* есть новая конфигурация. Для этого *ANIC* проверяет конкретную конечную точку через UNIX-сокет `unix:/var/lib/angie/angie-config-version.sock`.
10. (N/A) Чтобы запустить Angie, *ANIC* запускает команду `angie`, которая запускает *главный процесс Angie*.
11. (Сигнал) Чтобы перезагрузить Angie, *ANIC* выполняет команду `angie -s reload`, которая проверяет конфигурацию и отправляет сигнал перезагрузки *главному процессу Angie*.
12. (Сигнал) Чтобы выключить Angie, *ANIC* выполняет команду `angie -s quit`, которая отправляет сигнал плавного выключения *главному процессу Angie*.
13. (Файловый ввод-вывод) *Главный процесс Angie* отправляет журнальный вывод в свои потоки *stdout* и *stderr*, которые собираются средой выполнения контейнеров.
14. (Файловый ввод-вывод) *Главный процесс Angie* считывает *TLS-сертификат и ключи*, указанные в конфигурации, при запуске или перезагрузке.
15. (Файловый ввод-вывод) *Главный процесс Angie* считывает *конфигурационные файлы* при запуске или во время перезагрузки.
16. (Сигнал) *Главный процесс Angie* управляет жизненным *циклом рабочих процессов Angie*. Он создает рабочие процессы с новой конфигурацией и отключает процессы со старой.
17. (Файловый ввод-вывод) *Рабочий процесс Angie* отправляет журнальный вывод в свои потоки *stdout* и *stderr*, которые собираются средой выполнения контейнеров.
18. (UDP) *Рабочий процесс Angie* отправляет журналы задержки ответов HTTP-апстрима в формате Syslog через UNIX-сокет `/var/lib/angie/angie-syslog.sock` в *ANIC*. В свою очередь, *ANIC* анализирует и преобразует эти журналы в метрики Prometheus.
19. (HTTP, HTTPS, TCP, UDP) *Клиент* отправляет и получает трафик от любого из *рабочих процессов Angie* через порты 80 и 443 и любые дополнительные порты, открытые в ресурсе GlobalConfiguration.
20. (HTTP, HTTPS, TCP, UDP) *Рабочий процесс Angie* отправляет трафик на *бэкенды* и получает трафик от них.
21. (HTTP) *Администратор* может подключиться к Angie stub_status, используя порт 8080, через *рабочий процесс Angie*. **Примечание**. По умолчанию Angie разрешает подключения только от `localhost`.

2.4 Процесс Ingress Controller

В этом разделе рассматривается архитектура процесса ANIC, включая следующие вопросы:

- Как ANIC обрабатывает новый ресурс Ingress, созданный пользователем.
- Краткое описание того, как ANIC работает и как он соотносится с контроллерами Kubernetes.
- Различные компоненты процесса ANIC.

2.4.1 Обработка нового ресурса Ingress

Ниже рассказано, как ANIC обрабатывает новый ресурс Ingress. Для простоты мы представляем главный и рабочий процессы Angie в виде единого блока *Angie*. Также обратите внимание, что ресурсы VirtualServer и VirtualServerRoute обрабатываются аналогично.

Обработка нового ресурса Ingress включает в себя следующие шаги:

1. *Пользователь* создает новый ресурс Ingress.
2. Процесс ANIC поддерживает *кэш* ресурсов в кластере. *Кэш* содержит только те ресурсы, которые интересуют ANIC, такие как Ingress. *Кэш* синхронизируется с API Kubernetes, отслеживая изменения в ресурсах.
3. Как только в *кэше* появляется новый ресурс Ingress, он уведомляет *контур управления* об измененном ресурсе.
4. *Контур управления* получает последнюю версию ресурса Ingress из *кэша*. Поскольку ресурс Ingress ссылается на другие ресурсы, такие как секреты TLS, *контур управления* также получает последние версии любых ресурсов, на которые ведут такие ссылки.
5. *Контур управления* генерирует TLS-сертификаты и ключи из секретов TLS и записывает их в файловую систему.
6. *Контур управления* генерирует и записывает *конфигурационные файлы* Angie, которые соответствуют ресурсу Ingress, и записывает их в файловую систему.
7. *Контур управления* перезагружает *Angie* и ожидает успешной перезагрузки *Angie*. В ходе перезагрузки:
   - *Angie* считывает *TLS-сертификаты и ключи*.
   - *Angie* считывает *конфигурационные файлы*.
8. *Контур управления* генерирует событие для ресурса Ingress и обновляет его статус. Если перезагрузка завершается неудачей, событие будет содержать сообщение об ошибке.

2.4.2 Ingress Controller — это контроллер Kubernetes

Основываясь на примере из предыдущего раздела, мы можем обобщить принципы работы Ingress:

ANIC постоянно обрабатывает как новые ресурсы, так и изменения в существующих ресурсах кластера. В результате конфигурация Angie остается актуальной для ресурсов кластера.

ANIC является примером контроллера Kubernetes: ANIC запускает контур управления, который гарантирует, что Angie всегда будет настроен в соответствии с желаемым состоянием (это ресурсы Ingress и другие ресурсы).

Желаемое состояние сосредоточено в следующих встроенных и пользовательских (CR) ресурсах Kubernetes:

- Конфигурация балансировки нагрузки уровня 7:
  - Ресурсы Ingress
  - Ресурсы VirtualServer (CR)
  - Ресурсы VirtualServerRoute (CR)
- Политики уровня 7:
  - Ресурсы Policy (CR)
- Конфигурация балансировки нагрузки уровня 4:
  - Ресурсы TransportServer (CR)
- Обнаружение сервисов:

  - Ресурсы Service
  - Ресурсы Endpoint
  - Ресурсы Pod
- Конфигурация секретов:
  - Ресурсы Secret
- Глобальная конфигурация:
  - Ресурс ConfigMap (только один ресурс)
  - Ресурс GlobalConfiguration (CR, только один ресурс)

В следующем разделе мы рассмотрим различные компоненты процесса ANIC.

2.5 Компоненты процесса Ingress Controller

В этом разделе мы опишем компоненты процесса ANIC и то, как они взаимодействуют, включая следующие вопросы:

1. То, как ANIC следит за изменениями ресурсов.
2. Основные компоненты контура управления ANIC.
3. Как эти компоненты обрабатывают изменение ресурса.
4. Несколько дополнительных компонентов, которые имеют решающее значение для обработки изменений.

ANIC написан на go и в значительной степени зависит от клиента Go для Kubernetes.

2.5.1 Кэши ресурсов

В разделе *Обработка нового ресурса Ingress* мы упоминали, что ANIC поддерживает кэш ресурсов в кластере, который синхронизируется с API Kubernetes, отслеживая изменения в ресурсах. Мы также упоминали, что как только кэш обновляется, он уведомляет контур управления об измененном ресурсе.

Кэш на самом деле представляет собой набор *информеров*. Далее рассказано, как изменения в ресурсах обрабатываются ANIC.

- Для каждого типа ресурса, который отслеживает ANIC, создается информер. *Информер* включает в себя *хранилище*, в котором хранятся ресурсы этого типа. Чтобы синхронизировать это *хранилище* с последними версиями ресурсов в кластере, *информер* использует *API-интерфейсы наблюдения и перечисления Kubernetes* для этого типа ресурсов.
- Когда в кластере происходит изменение (например, создается новый ресурс), *информер* обновляет свое *хранилище* и вызывает обработчики для этого *информера*.
- ANIC регистрирует обработчики для каждого *информера*. В большинстве случаев *обработчик* создает запись для затронутого ресурса в *рабочей очереди*, где элемент рабочей очереди включает тип ресурса, его пространство имен и название.
- *Рабочая очередь* всегда пытается освободить саму себя: если в ее начале есть элемент, очередь удалит его и отправит его *контроллеру*, используя функцию обратного вызова.
- *Контроллер* является основным компонентом в ANIC, который и реализует контур управления. Описание компонентов см. в разделе *Контур управления*. Пока достаточно знать, что для обработки элемента рабочей очереди *контроллер* получает последнюю версию ресурса из *хранилища*, перенастраивает *Angie* в соответствии с ресурсом, обновляет статус ресурса и отправляет событие через *API Kubernetes*.

2.5.2 Контур управления

В этом разделе рассматриваются основные компоненты ANIC, из которых состоит контур управления:

- Контроллер:
  - Запускает контур управления ANIC.
  - Создает экземпляры *информеров*, *обработчиков*, *рабочей очереди* и дополнительных вспомогательных компонентов.
  - Включает метод синхронизации (см. следующий раздел), который вызывает *рабочая очередь* для обработки измененного ресурса.
  - Передает измененные ресурсы в *конфигуратор* для перенастройки Angie.
- Конфигуратор:
  - Генерирует файлы конфигурации Angie, ключи TLS и сертификаты на основе ресурса Kubernetes.
  - Использует *менеджер* для записи сгенерированных файлов и перезагрузки Angie.
- Менеджер:
  - Управляет жизненным циклом Angie (запуск, перезагрузка, завершение работы).
  - Управляет конфигурационными файлами, ключами TLS и сертификатами.

Вспомогательные компоненты

Есть два дополнительных вспомогательных компонента, имеющих решающее значение для обработки изменений: *Конфигурация* и *Локальное хранилище секретов*.

Конфигурация

Конфигурация содержит последнее допустимое состояние ресурсов конфигурации балансировки нагрузки ANIC, как то: ресурсы Ingress, VirtualServer, VirtualServerRoute, TransportServer и GlobalConfiguration.

Конфигурация поддерживает операции добавления (для добавления или обновления) и удаления ресурсов. Когда вы добавляете, обновляете или удаляете ресурс в конфигурации, она делает следующее:

1. Проверяет объект (в случае добавления или обновления)
2. Вычисляет изменения в затронутых ресурсах, которые необходимо передать в конфигурацию Angie, возвращая изменения вызывающей стороне.

Например, когда вы добавляете новый ресурс Ingress, *конфигурация* возвращает изменение, требующее от ANIC добавления конфигурации для этого ресурса в конфигурационные файлы Angie. Другой пример: если вы сделаете существующий ресурс Ingress недействительным, *конфигурация* вернет изменение, требующее от ANIC удалить конфигурацию для этого ресурса из конфигурационных файлов Angie.

Кроме того, *конфигурация* гарантирует, что только один ресурс Ingress, VirtualServer или TransportServer (TLS Passthrough) содержит определенный хост (например, mysite.example.com), и только один ресурс TransportServer (TCP, UDP) содержит определенный прослушиватель (например, порт 53 для UDP). Это гарантирует, что в конфигурации Angie не произойдет коллизий между хостом и прослушивателем.

В конечном счете, ANIC гарантирует, что конфигурация Angie в файловой системе отражает состояние объектов в *конфигурации* в любой момент времени.

Локальное хранилище секретов

Локальное хранилище секретов содержит допустимые секретные ресурсы и синхронизирует с ними соответствующие файлы в файловой системе. Секреты используются для хранения сертификатов и ключей TLS (тип `kubernetes.io/tls`), центров сертификации, а также клиентских секретов поставщика OIDC.

Когда *контроллер* обрабатывает изменение в ресурсе конфигурации, таком как Ingress, он создает расширенную версию ресурса, которая включает зависимости, такие как секреты, необходимые для генерации конфигурации Angie. *Локальное хранилище секретов* позволяет *контроллеру* получить ссылку на файловую систему для секрета с помощью ключа секрета (пространство имен/имя).

2.6 Перезагрузка Angie

В следующем разделе рассматривается перезагрузка Angie в целом и, в частности, ее реализация в ANIC.

2.6.1 Перезагрузка в целом

Перезагрузка Angie необходима для применения новой конфигурации и включает в себя следующие действия:

1. Администратор отправляет сигнал HUP (зависание) главному процессу Angie, чтобы запустить перезагрузку.
2. Главный процесс завершает работу рабочих процессов со старой конфигурацией и запускает рабочие процессы с новой конфигурацией.
3. Администратор проверяет, что перезагрузка успешно завершена.

Примечание: Обратитесь к документации Angie для получения более подробной информации о перезагрузке.

Как выполнить перезагрузку

Двоичный файл Angie (`angie`) поддерживает операцию перезагрузки с параметром `-s reload`. Когда вы используете эту опцию:

1. Процесс проверяет новую конфигурацию Angie и завершает работу, если она недействительна, выводя сообщения об ошибках в stderr.
2. Он посылает сигнал HUP главному процессу Angie и завершает работу.

В качестве альтернативы вы можете отправить сигнал HUP непосредственно главному процессу Angie.

Как убедиться в успехе перезагрузки

Команда `angie -s reload` не дожидается завершения перезагрузки Angie. В результате именно администратор несет ответственность за подтверждение ее успешности. Есть несколько вариантов:

- Проверьте, создал ли главный процесс новые рабочие процессы. Например, запустив `ps` или прочитав файловую систему `/proc`.
- Отправьте HTTP-запрос в Angie, и если ответит новый рабочий процесс, вы будете знать, что Angie успешно перезагрузился.

Примечание: Для этого требуется дополнительная настройка Angie.

Перезагрузка занимает некое время, обычно не менее 200 мс. Это время зависит от размера конфигурации, количества TLS-сертификатов и ключей, включенных модулей, подробностей конфигурации и доступных ресурсов ЦП.

Потенциальные проблемы

В большинстве случаев, если команда `angie -s reload` завершается успешно, перезагрузка также будет успешной. В редких случаях перезагрузка завершается неудачей, и главный процесс Angie добавит сообщение об ошибке в журнал ошибок. Например:

```
2022/07/09 00:56:42 [emerg] 1353#1353: limit_req "one" uses the
"$remote_addr" key while previously it used the "$binary_remote_addr" key
```

Операция выполняется плавно; перезагрузка не приводит к потере трафика Angie. Однако частые перезагрузки могут привести к повышенной загрузке памяти и потенциальной остановке Angie с ошибкой OOM (нехватка памяти), что приведет к потере трафика. Это может произойти, если вы 1) проксируете трафик, который использует долгоживущие соединения (например, Websockets, gRPC) и 2) часто перезагружаете конфигурацию. В этом случае вы можете столкнуться с несколькими поколениями завершающих работу рабочих процессов Angie (старые рабочие процессы Angie не будут завершаться до тех пор, пока все соединения не будут прерваны либо клиентами, либо бэкендами, если только вы не настроите worker_shutdown_timeout, что заставит старые рабочие процессы завершать работу после тайм-аута). В конечном счете все эти рабочие процессы могут исчерпать доступную системную память.

Поскольку как старые, так и новый рабочие процессы Angie сосуществуют во время перезагрузки, она может привести к резкому увеличению использования памяти вплоть до двукратного. Из-за нехватки доступной памяти главный процесс Angie может лишиться возможности создавать новые рабочие процессы.

ГЛАВА 3

Отчеты о состоянии ресурсов

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:
    1. Определенный пользователем адрес, указанный в ключе ConfigMap `external-status-address`.
    2. 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.

3.2 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                                ␣
↪                                PORTS      AGE
    mysite Valid   mysite.example.com       ae430f41a1a0042908655abcdefghijkl-
↪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
```

3.2.1 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`
`ExternalI`	Список внешних конечных точек, для которых хосты ресурса являются общедоступными.	*[]externalEndpoint*

Следующее поле отображается только в состоянии VirtualServerRoute:

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

3.2.2 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:

1. Если вы хотите, чтобы ANIC сообщал о внешних конечных точках, определите источник для внешнего адреса

Примечание: Остальные поля будут включаться в отчет и без настроенного внешнего адреса). Это может быть:

---

   1. Определенный пользователем адрес, указанный в ключе ConfigMap external-status-address.
   2. 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 VirtualServer и VirtualServerRoute.

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
```

3.3.1 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`
`Reaso`	Причина последнего обновления.	`string`
`Messa`	Дополнительная информация о состоянии.	`string`

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
```

3.4.1 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`
`Reasc`	Причина последнего обновления.	`string`
`Messa`	Дополнительная информация о состоянии.	`string`

ГЛАВА 4

Пользовательские шаблоны

ANIC использует шаблоны для генерации конфигурации Angie для ресурсов Ingress, ресурсов VirtualServer и основного файла конфигурации Angie. Вы можете настроить собственные шаблоны и применить их с помощью ConfigMap.