Menu

Сетевые взаимодействия

Введение

Введение

Преимущества

Сценарии применения

Ограничения использования

Архитектура

Понимание Kube-OVN

Компоненты upstream OVN/OVS

Основной контроллер и агент

Инструменты мониторинга, эксплуатации и расширения

Понимание ALB

Основные компоненты

Быстрый старт

Общие понятия ALB

Взаимосвязь между ALB, ALB Instance, Frontend/FT, Rule, Ingress и Project

ALB Leader

Дополнительные ресурсы:

Понимание MetalLB

Терминология

Принципы высокой доступности в MetalLB

Алгоритм MetalLB для выбора узлов-хозяев VIP

Пулы внешних адресов и количество узлов

Дополнительные ресурсы

Основные понятия

Auth

Основная концепция

Быстрый старт

Связанные аннотации Ingress

forward-auth

basic-auth

CR

Специальная аннотация Ingress для ALB

Другие функции, связанные с аутентификацией в Ingress-Nginx

Примечание: несовместимые моменты с Ingress-Nginx

Устранение неполадок

Совместимость аннотаций ingress-nginx

Основные понятия

Поддерживаемые аннотации ingress-nginx

TCP/HTTP Keepalive

Основная концепция

CRD

ModSecurity

Терминология

Процедура эксплуатации

Связанные объяснения

Пример конфигурации

Сравнение различных методов Ingress

Для L4 (TCP/UDP) трафика

Для L7 (HTTP/HTTPS) трафика

HTTP Redirect

Основная концепция

CRD

Аннотация Ingress

Редирект на уровне порта

Редирект на уровне правила

L4/L7 Таймаут

Основная концепция

CRD

Что означает таймаут

Аннотация Ingress

Таймаут на уровне порта

GatewayAPI

OTel

Терминология

Предварительные требования

Процедура

Связанные операции

Дополнительные сведения

Пример конфигурации

Руководства

Создание сервисов

Создание Ingress

Настройка Gateway

Создание доменного имени

Создание сертификатов

Создание пула внешних IP-адресов

Создание BGP-пиров

Настройка подсетей

Настройка сетевых политик

Создание Admin Network Policies

Примечания

Создание AdminNetworkPolicy или BaselineAdminNetworkPolicy через веб-консоль

Создание AdminNetworkPolicy или BaselineAdminNetworkPolicy через CLI

Дополнительные ресурсы

Настройка сетевых политик кластера

Примечания

Процедура

Как сделать

Развертывание высокодоступного VIP для ALB

Метод 1: Использование внутренней маршрутизации типа LoadBalancer для предоставления VIP

Метод 2: Использование внешнего устройства балансировщика нагрузки для предоставления VIP

Soft Data Center LB Solution (Alpha)

Предварительные требования

Процедура

Проверка

Подготовка физической сети Kube-OVN Underlay

Инструкции по использованию

Объяснение терминологии

Требования к среде

Пример конфигурации

Автоматическое взаимное подключение подсетей Underlay и Overlay

Процедура

Использование OAuth Proxy с ALB

Overview

Procedure

Result

Создание GatewayAPI Gateway

Развертывание MetalLB

Установка Pod Security Policies в режим Privileged

Настройка балансировщика нагрузки

Как правильно выделять ресурсы CPU и памяти

Проброс IPv6-трафика на IPv4-адреса внутри кластера

Метод настройки

Проверка результата

Calico Network поддерживает шифрование WireGuard

Статус установки

Терминология

Примечания

Требования

Процедура

Проверка результата

Kube-OVN Overlay Network поддерживает шифрование IPsec

Терминология

Примечания

Предварительные требования

Процедура

ALB Monitoring

Terminology

Procedure

Monitoring Metrics

Устранение неполадок

Как решить проблемы межузловой коммуникации в ARM-средах?

Определение причины ошибки

Введение

Контейнерная сеть — это комплексное сетевое решение, разработанное для облачно-нативных приложений, обеспечивающее беспрепятственную восточно-западную коммуникацию внутри кластеров и эффективное управление северо-южным трафиком через внешние сети, при этом предоставляя необходимые сетевые функции. Она состоит из следующих основных компонентов:

- Container Network Interfaces (CNI) для управления восточно-западным трафиком внутри кластера.
- Ingress Gateway Controller ALB для управления HTTPS входящим трафиком.
- MetalLB для обработки сервисов типа LoadBalancer.
- Кроме того, обеспечивает надежные функции сетевой безопасности и шифрования для гарантии защищенной коммуникации.

Содержание

Преимущества

Контейнерная сеть предлагает следующие ключевые преимущества:

- **Гибкое управление сетью**

Поддерживая несколько CNI, контейнерная сеть поддерживает режимы overlay, underlay и маршрутизации, обеспечивая гибкость для адаптации к разнообразным сетевым средам. Также она предлагает тонкое распределение IP-адресов и надежное управление исходящим трафиком. Будучи командой-основателем Kube-OVN, мы обладаем обширным практическим опытом в создании и поддержке сетей большого масштаба, обеспечивая надежное и производительное соединение.

- **Изоляция, мультиарендность и гибкость API для Ingress Gateway**

  С помощью оператора ALB можно создавать и управлять несколькими экземплярами ALB в одном кластере. Каждый арендатор может иметь выделенную группу экземпляров ALB в качестве ingress gateway, обеспечивая эффективную изоляцию и управление ресурсами. Кроме того, пользователи могут гибко выбирать между Ingress и Gateway API в зависимости от своих предпочтений и операционных требований, обеспечивая беспрепятственное управление трафиком и повышенную гибкость. Будучи командой-основателем ALB, мы гарантируем надежное и масштабируемое решение.

- **Комплексная сетевая безопасность**

  Контейнерная сеть предоставляет многоуровневую систему безопасности для защиты на всех уровнях. На уровне CNI поддерживаются различные модели политик безопасности, включая NetworkPolicy и AdminNetworkPolicy, для реализации тонкого контроля доступа к сети. Для безопасной передачи данных сеть включает надежное шифрование трафика. На уровне Ingress Gateway предоставляются продвинутые механизмы безопасности, такие как TLS termination и поддержка ModSecurity, обеспечивая всестороннюю защиту внешних приложений. Благодаря встроенному применению сетевых политик, шифрованию и мониторингу трафика обеспечивается защита от несанкционированного доступа и соблюдение стандартов безопасности.

Сценарии применения

Контейнерная сеть особенно подходит для следующих сценариев:

- **Управление восточно-западным трафиком**

Использование CNI для эффективной коммуникации между pod внутри кластера с поддержкой как overlay, так и underlay сетевых режимов для удовлетворения различных требований развертывания.

- **Контроль северо-южного трафика**

  Использование ALB в качестве Ingress Gateway Controller для управления внешним HTTPS трафиком с гибким выбором API и возможностями мультиарендной изоляции для разных команд.

- **Экспонирование сервисов Load Balancer**

  Применение MetalLB для обеспечения высокой доступности сервисов типа LoadBalancer, позволяя надежный внешний доступ к сервисам кластера через виртуальные IP-адреса.

- **Сетевая безопасность и шифрование**

  Реализация комплексной безопасности с помощью NetworkPolicy, AdminNetworkPolicy и шифрования трафика для обеспечения защищенной коммуникации по всей сетевой инфраструктуре.

Ограничения использования

Несмотря на широкие функциональные возможности, следует учитывать следующие ограничения:

- **Требование к underlay сети**

  Некоторые возможности underlay сети, такие как Kube-OVN Underlay Subnet, Egress IP и MetalLB, требуют поддержки базовой L2 сети. Эти функции не могут использоваться в публичных облачных провайдерах и некоторых виртуализированных средах, таких как AWS и GCP.

Благодаря универсальному дизайну и комплексному набору функций, контейнерная сеть позволяет организациям создавать, масштабировать и управлять безопасными, надежными и высокопроизводительными контейнеризированными приложениями.

Menu

Архитектура

Понимание Kube-OVN

Компоненты upstream OVN/OVS

Основной контроллер и агент

Инструменты мониторинга, эксплуатации и расширения

Понимание ALB

Основные компоненты

Быстрый старт

Общие понятия ALB

Взаимосвязь между ALB, ALB Instance, Frontend/FT, Rule, Ingress и Project

ALB Leader

Дополнительные ресурсы:

Понимание MetalLB

Терминология

Принципы высокой доступности в MetalLB

Алгоритм MetalLB для выбора узлов-хозяев VIP

Пулы внешних адресов и количество узлов

Дополнительные ресурсы

Понимание Kube-OVN

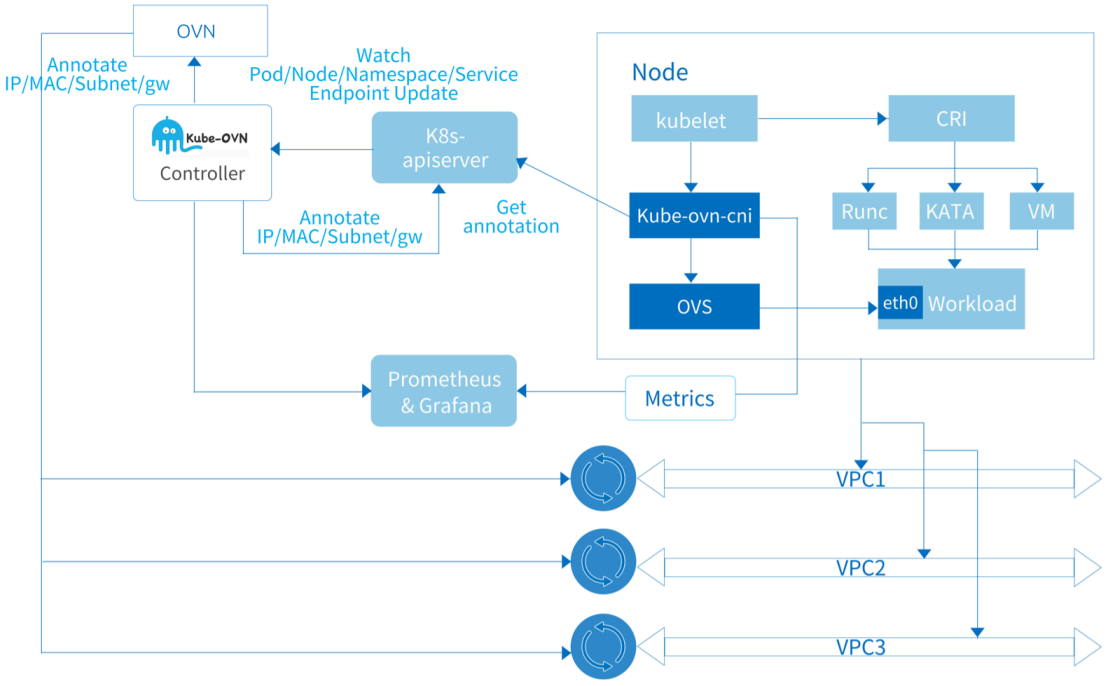
В этом документе описывается общая архитектура Kube-OVN, функциональность каждого компонента и их взаимодействие.

В целом, Kube-OVN служит мостом между Kubernetes и OVN, объединяя проверенные SDN с Cloud Native. Это означает, что Kube-OVN не только реализует сетевые спецификации в Kubernetes, такие как CNI, Service и NetworkPolicy, но и приносит множество возможностей SDN в облачную нативную среду, таких как логические коммутаторы, логические маршрутизаторы, VPC, шлюзы, QoS, ACL и зеркалирование трафика.

Kube-OVN также поддерживает хорошую открытость для интеграции с множеством технологических решений, таких как Cilium, Submariner, Prometheus, KubeVirt и др.

Компоненты Kube-OVN можно условно разделить на три категории.

- Компоненты upstream OVN/OVS.
- Основной контроллер и агент.
- Инструменты мониторинга, эксплуатации и расширения.



Содержание

Компоненты upstream OVN/OVS

Данный тип компонентов происходит из сообщества OVN/OVS с конкретными модификациями для сценариев использования Kube-OVN. OVN/OVS — это зрелая SDN-система для управления виртуальными машинами и контейнерами, и мы настоятельно рекомендуем пользователям, заинтересованным в реализации Kube-OVN, сначала ознакомиться с ovn-architecture(7)↗, чтобы понять, что такое OVN и как с ним интегрироваться. Kube-OVN использует northbound-интерфейс OVN для создания и координации виртуальных сетей и отображения сетевых концепций в Kubernetes.

Все компоненты, связанные с OVN/OVS, упакованы в образы и готовы к запуску в Kubernetes.

ovn-central

Deployment `ovn-central` запускает компоненты контрольной плоскости OVN, включая `ovn-nb`, `ovn-sb` и `ovn-northd`.

- `ovn-nb`: Сохраняет конфигурацию виртуальной сети и предоставляет API для управления виртуальной сетью. `kube-ovn-controller` в основном взаимодействует с `ovn-nb` для настройки виртуальной сети.
- `ovn-sb`: Хранит таблицу логических потоков, сгенерированную из логической сети `ovn-nb`, а также фактическое состояние физической сети каждого узла.
- `ovn-northd`: преобразует виртуальную сеть из `ovn-nb` в таблицу логических потоков в `ovn-sb`.

Несколько экземпляров `ovn-central` синхронизируют данные через протокол Raft для обеспечения высокой доступности.

ovs-ovn

`ovs-ovn` запускается как DaemonSet на каждом узле, внутри Pod работают `openvswitch`, `ovsdb` и `ovn-controller`. Эти компоненты выступают агентами для `ovn-central`, преобразуя таблицы логических потоков в реальные сетевые конфигурации.

Основной контроллер и агент

Эта часть является ядром Kube-OVN, служит мостом между OVN и Kubernetes, связывая две системы и переводя сетевые концепции между ними. Большинство основных функций реализованы в этих компонентах.

kube-ovn-controller

Этот компонент выполняет трансляцию всех ресурсов внутри Kubernetes в ресурсы OVN и выступает в роли контрольной плоскости всей системы Kube-OVN. `kube-ovn-controller` слушает события по всем ресурсам, связанным с сетевой функциональностью, и обновляет логическую сеть внутри OVN на основе изменений ресурсов. Основные ресурсы, за которыми ведется наблюдение, включают:

Pod, Service, Endpoint, Node, NetworkPolicy, VPC, Subnet, Vlan, ProviderNetwork.

На примере события создания Pod: `kube-ovn-controller` слушает событие создания Pod, выделяет адрес с помощью встроенной функции IPAM в памяти, и вызывает `ovn-`

`central` для создания логических портов, статических маршрутов и возможных правил ACL. Далее `kube-ovn-controller` записывает назначенный адрес и информацию о подсети, такую как CIDR, шлюз, маршрут и т.д., в аннотацию Pod. Эту аннотацию затем читает `kube-ovn-cni` и использует для настройки локальной сети.

kube-ovn-cni

Этот компонент запускается на каждом узле как DaemonSet, реализует интерфейс CNI и управляет локальным OVS для настройки локальной сети.

DaemonSet копирует бинарный файл `kube-ovn` на каждую машину как инструмент взаимодействия между `kubelet` и `kube-ovn-cni`. Этот бинарный файл отправляет соответствующий CNI-запрос в `kube-ovn-cni` для дальнейших операций. По умолчанию бинарный файл копируется в каталог `/opt/cni/bin`.

`kube-ovn-cni` настраивает конкретную сеть для выполнения соответствующих операций с трафиком, основные задачи включают:

1. Настройка `ovn-controller` и `vswitchd`.
2. Обработка запросов CNI Add/Del:
    2.1. Создание или удаление пары veth и привязка или отвязка к портам OVS.
    2.2. Настройка портов OVS.
    2.3. Обновление правил iptables/ipset/route на хосте.
3. Динамическое обновление QoS сети.
4. Создание и настройка NIC `ovn0` для соединения сети контейнеров и сети хоста.
5. Настройка NIC хоста для реализации Vlan/Underlay/EIP.
6. Динамическая настройка межкластерных шлюзов.

Инструменты мониторинга, эксплуатации и расширения

Эти компоненты предоставляют средства мониторинга, диагностики, инструменты эксплуатации и внешние интерфейсы для расширения основных сетевых возможностей Kube-OVN и упрощения повседневных операций и обслуживания.

kube-ovn-speaker

Этот компонент — DaemonSet, работающий на узлах с определенной меткой, который публикует маршруты во внешний мир, позволяя внешнему доступу к контейнерам напрямую через IP Pod.

kube-ovn-pinger

Этот компонент — DaemonSet, работающий на каждом узле, собирает информацию о состоянии OVS, качестве сети узла, задержках в сети и т.д.

kube-ovn-monitor

Этот компонент собирает информацию о состоянии OVN и метрики мониторинга.

kubectl-ko

Этот компонент — плагин kubectl, который позволяет быстро выполнять распространённые операции.

Понимание ALB

ALB (Another Load Balancer) — это Kubernetes Gateway на базе OpenResty с многолетним опытом эксплуатации от Alauda.

Содержание

ALB Instance

ALB-Operator

Frontend (сокращённо FT)

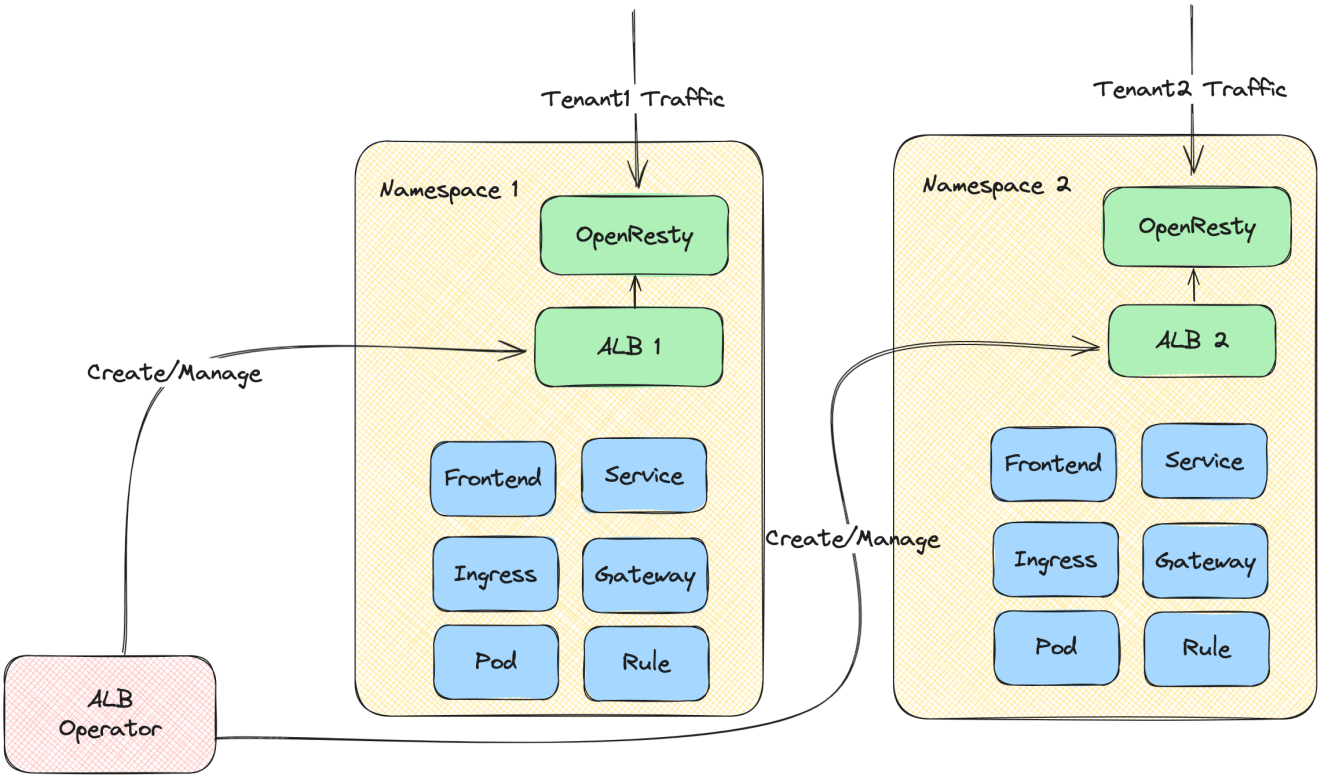
RULE

ALB Leader

Project

Дополнительные ресурсы:

Основные компоненты



- **ALB Operator**: оператор, управляющий жизненным циклом экземпляров ALB. Он отвечает за наблюдение за CR ALB и создание, обновление экземпляров ALB для разных арендаторов.
- **ALB Instance**: экземпляр ALB включает OpenResty, который выступает в роли data plane, и Go контроллер как control plane. Go контроллер отслеживает различные CR (Ingress, Gateway, Rule и др.) и преобразует их в ALB-специфичные DSL правила. OpenResty затем использует эти DSL правила для сопоставления и обработки входящих запросов.

Быстрый старт

Развертывание ALB Operator

1. Создайте кластер.
2.

```
 helm repo add alb https://alauda.github.io/alb/;helm repo update;helm search repo|grep alb
```

3.

```
 helm install alb-operator alb/alauda-alb2
```

Развертывание экземпляра ALB

```
cat <<EOF | kubectl apply -f -
apiVersion: crd.alauda.io/v2beta1
kind: ALB2
metadata:
    name: alb-demo
    namespace: kube-system
spec:
    address: "172.20.0.5"  # IP-адрес узла, на котором развернут alb
    type: "nginx"
    config:
        networkMode: host
        loadbalancerName: alb-demo
        projects:
        - ALL_ALL
        replicas: 1
EOF
```

Запуск демонстрационного приложения

```
cat <<EOF | kubectl apply -f -
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-world
  labels:
    k8s-app: hello-world
spec:
  replicas: 1
  selector:
    matchLabels:
      k8s-app: hello-world
  template:
    metadata:
      labels:
        k8s-app: hello-world
    spec:
      terminationGracePeriodSeconds: 60
      containers:
      - name: hello-world
        image: docker.io/crccheck/hello-world:latest
        imagePullPolicy: IfNotPresent
---
apiVersion: v1
kind: Service
metadata:
  name: hello-world
  labels:
    k8s-app: hello-world
spec:
  ports:
  - name: http
    port: 80
    targetPort: 8000
  selector:
    k8s-app: hello-world
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: hello-world
spec:
  rules:
```

```
    - http:
        paths:
        - path: /
          pathType: Prefix
          backend:
            service:
              name: hello-world
              port:
                number: 80
  EOF
```

Теперь вы можете получить доступ к приложению через `curl http://${ip}`

Общие понятия ALB

Ниже определены общие понятия в ALB.

Auth

Auth — это механизм, выполняющий аутентификацию до того, как запрос достигнет фактического сервиса. Он позволяет обрабатывать аутентификацию на уровне ALB единообразно, без необходимости реализовывать логику аутентификации в каждом бэкенд-сервисе.

Подробнее об ALB Auth.

Network Mode

Экземпляр ALB может быть развернут в двух режимах: host network mode и container network mode.

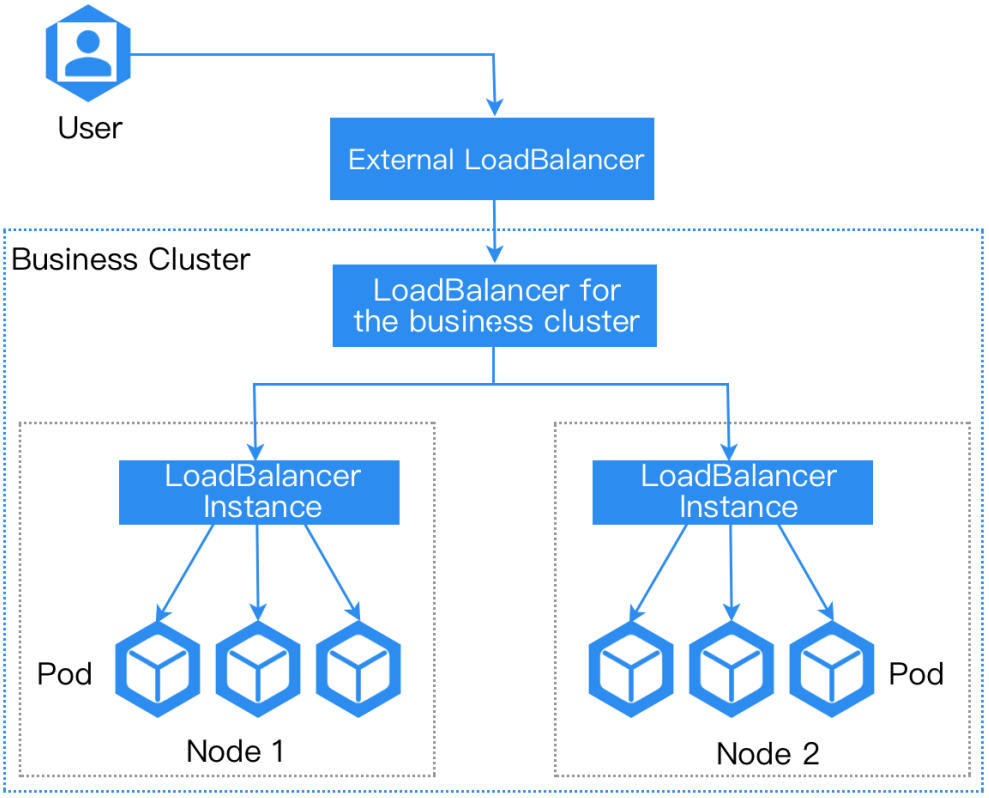
Host Network Mode

Использует сетевой стек узла напрямую, разделяя IP-адрес и порт с узлом.

В этом режиме экземпляр балансировщика нагрузки напрямую привязывается к порту узла, без проброса портов или иной контейнерной сетевой инкапсуляции.

NOTE

Чтобы избежать конфликтов портов, на одном узле разрешается развертывать только один экземпляр ALB.



В режиме host-network экземпляр ALB по умолчанию слушает все сетевые интерфейсы узла.

Преимущества:

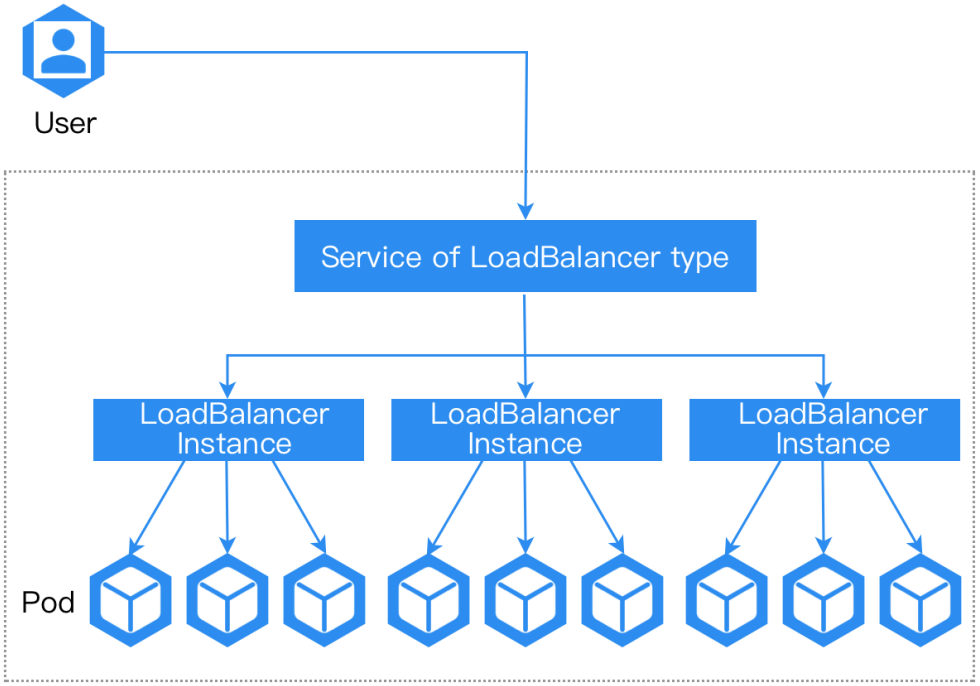
1. Максимальная сетевая производительность.
2. Доступ по IP-адресу узла.

Недостатки:

1. На одном узле разрешён только один экземпляр ALB.
2. Возможны конфликты портов с другими процессами.

Container Network Mode

В отличие от host network mode, container network mode разворачивает ALB с использованием контейнерной сети.



Преимущества:

1. Поддержка нескольких экземпляров ALB на одном узле.
2. ALB интегрирован с MetalLB, который может предоставлять VIP для ALB.
3. Отсутствие конфликтов портов с другими процессами.

Недостатки:

1. Немного более низкая производительность.
2. Доступ к ALB осуществляется через сервис LoadBalancer.

Frontend

Определён ресурс frontend (сокращённо ft), который используется для объявления всех портов, на которых должен слушать ALB.

Каждый frontend соответствует порту прослушивания на балансировщике нагрузки (LB). Frontend связывается с ALB через метки.

```yaml
apiVersion: crd.alauda.io/v1
kind: Frontend
metadata:
  labels:
    alb2.cpaas.io/name: alb-demo (1)
  name: alb-demo-00080 (2)
  namespace: cpaas-system
spec:
  backendProtocol: "http"
  certificate_name: "" (3)
  port: 80
  protocol: http (4)
  serviceGroup: (5)
    services:
      - name: hello-world
        namespace: default
        port: 80
        weight: 100 (6)
```

(1) Обязательно, указывает экземпляр ALB, которому принадлежит этот Frontend.

(2) Формат `$alb_name-$port`.

(3) Формат `$secret_ns/$secret_name`.

(4) Протокол самого Frontend.

- `http|https|grpc|grpcs` для l7 прокси.
- `tcp|udp` для l4 прокси.

(5) Для l4 прокси `serviceGroup` обязателен. Для l7 прокси `serviceGroup` является опциональным. При поступлении запроса ALB сначала пытается сопоставить его с правилами, связанными с этим Frontend. Если запрос не соответствует ни одному правилу, ALB перенаправляет его на дефолтный `serviceGroup`, указанный в конфигурации Frontend.

(6) Конфигурация `weight` применяется для алгоритмов планирования Round Robin и Weighted Round Robin.

NOTE

ALB слушает ingress и автоматически создаёт `Frontend` или Rule. Поле `source` определяется следующим образом:

2.1. `spec.source.type` в настоящее время поддерживает только `ingress`.

2.2. `spec.source.name` — имя ingress.

2.3. `spec.source.namespace` — namespace ingress.

Дополнительные ресурсы

- L4/L7 timeout
- Keepalive

Rules

Определён ресурс rule, который описывает, как экземпляр alb должен обрабатывать запросы 7-го уровня.

С помощью Rule можно настроить сложные шаблоны сопоставления и распределения трафика. При поступлении трафика он сопоставляется с внутренними правилами и выполняется соответствующая переадресация, а также предоставляются дополнительные функции, такие как cors, url rewrite и др.

```yaml
apiVersion: crd.alauda.io/v1
kind: Rule
metadata:
  labels:
    alb2.cpaas.io/frontend: alb-demo-00080 (1)
    alb2.cpaas.io/name: alb-demo (2)
  name: alb-demo-00080-test
  namespace: kube-system
spec:
  backendProtocol: "" (3)
  certificate_name: "" (4)
  dslx:
    - type: METHOD
      values:
        - - EQ
          - POST
    - type: URL
      values:
        - - STARTS_WITH
          - /app-a
        - - STARTS_WITH
          - /app-b
    - type: PARAM
      key: group
      values:
        - - EQ
          - vip
    - type: HOST
      values:
        - - ENDS_WITH
          - .app.com
    - type: HEADER
      key: LOCATION
      values:
        - - IN
          - east-1
          - east-2
    - type: COOKIE
      key: uid
      values:
        - - EXIST
    - type: SRC_IP
      values:
```

```
        - - RANGE
          - "1.1.1.1"
          - "1.1.1.100"
    enableCORS: false
    priority: 4 5
    serviceGroup: 6
      services:
        - name: hello-world
          namespace: default
          port: 80
          weight: 100
```

1. Обязательно, указывает Frontend, которому принадлежит это правило.
2. Обязательно, указывает ALB, которому принадлежит это правило.
3. Аналогично Frontend.
4. Аналогично Frontend.
5. Чем меньше число, тем выше приоритет.
6. Аналогично Frontend.

dslx

dslx — это предметно-ориентированный язык, используемый для описания критериев сопоставления.

Например, следующее правило сопоставляет запрос, который удовлетворяет всем перечисленным условиям:

- url начинается с /app-a или /app-b
- метод POST
- параметр url с именем group равен vip
- хост соответствует *.app.com
- заголовок LOCATION равен east-1 или east-2
- присутствует cookie с именем uid
- IP-адрес источника находится в диапазоне 1.1.1.1-1.1.1.100

```
dslx:
  - type: METHOD
    values:
      - - EQ
        - POST
  - type: URL
    values:
      - - STARTS_WITH
        - /app-a
      - - STARTS_WITH
        - /app-b
  - type: PARAM
    key: group
    values:
      - - EQ
        - vip
  - type: HOST
    values:
      - - ENDS_WITH
        - .app.com
  - type: HEADER
    key: LOCATION
    values:
      - - IN
        - east-1
        - east-2
  - type: COOKIE
    key: uid
    values:
      - - EXIST
  - type: SRC_IP
    values:
      - - RANGE
        - "1.1.1.1"
        - "1.1.1.100"
```

Изоляция проектов

Для rule по умолчанию действует изоляция по проектам: каждый пользователь видит только правила своего проекта.

Режим проекта

ALB может использоваться несколькими проектами, которые могут управлять этим ALB. Все порты ALB видны этим проектам.

Режим портового проекта

Порт ALB может принадлежать разным проектам. Такой режим называется Port Project Mode. Администратор задаёт диапазон портов для каждого проекта. Пользователи проекта могут создавать порты только в пределах этого диапазона и видеть только эти порты.

Взаимосвязь между ALB, ALB Instance, Frontend/FT, Rule, Ingress и Project

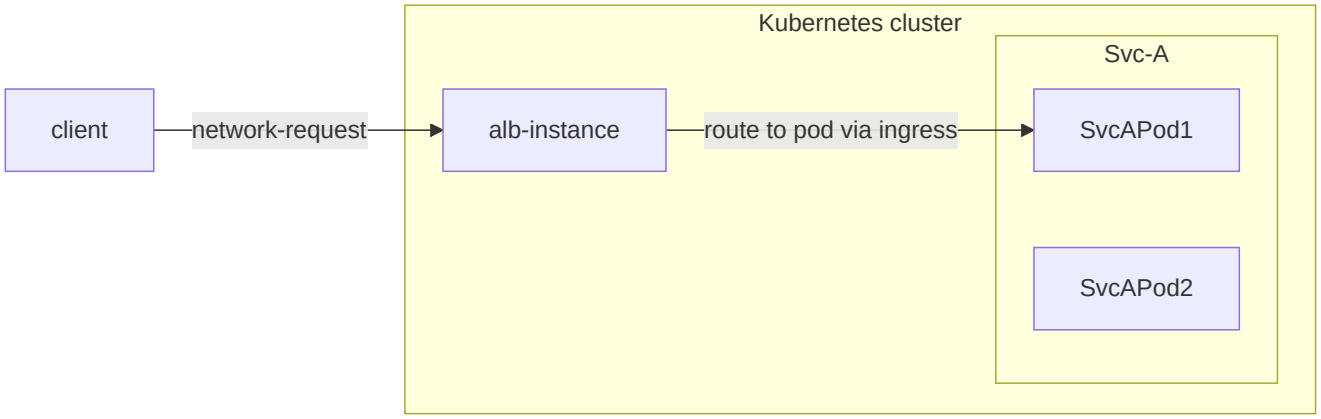
LoadBalancer — ключевой компонент современных облачно-нативных архитектур, выступающий в роли интеллектуального маршрутизатора и балансировщика трафика.

Чтобы понять, как работает ALB в Kubernetes кластере, нужно разобраться в нескольких основных понятиях и их взаимосвязях:

- Сам ALB
- Frontend (FT)
- Правила (Rules)
- Ресурсы Ingress
- Проекты

Эти компоненты совместно обеспечивают гибкое и мощное управление трафиком.

Далее описано, как эти понятия взаимодействуют и какую роль играют в цепочке обработки запросов. Подробные описания каждого понятия будут рассмотрены в отдельных статьях.



В цепочке обработки запросов:

1. Клиент отправляет HTTP/HTTPS/другой протокол запрос, который в итоге **попадает на pod ALB**, и pod (экземпляр ALB) начинает обрабатывать этот запрос.
2. Экземпляр ALB находит правило, которое может сопоставиться с этим запросом.
3. При необходимости модифицирует/перенаправляет/переписывает запрос согласно правилу.
4. Находит и выбирает IP pod из сервисов, указанных в правиле, и пересылает запрос на pod.

Ingress

Ingress — ресурс Kubernetes, описывающий, какой запрос должен быть направлен в какой сервис.

Ingress Controller

Программа, которая понимает ресурс Ingress и проксирует запросы в сервис.

ALB

ALB — это Ingress controller.

В Kubernetes кластере мы используем ресурс `alb2` для управления ALB. Вы можете использовать `kubectl get alb2 -A` для просмотра всех ALB в кластере.

ALB создаются пользователями вручную. Каждый ALB имеет свой IngressClass. При создании Ingress можно указать поле `.spec.ingressClassName`, чтобы определить, какой Ingress controller должен обрабатывать этот Ingress.

ALB Instance

ALB также представляет собой Deployment (набор pod-ов), работающий в кластере. Каждый pod называется экземпляром ALB.

Каждый экземпляр ALB обрабатывает запросы независимо, но все экземпляры разделяют Frontend (FT), Rule и другие конфигурации, принадлежащие одному ALB.

ALB-Operator

ALB-Operator — стандартный компонент, развернутый в кластере, оператор для ALB. Он создаёт/обновляет/удаляет Deployment и другие связанные ресурсы для каждого ALB согласно ресурсу ALB.

Frontend (сокращённо FT)

FT — ресурс, определённый самим ALB. Он представляет порты прослушивания экземпляра ALB.

FT может создаваться ALB-Leader или пользователем вручную.

Случаи создания FT ALB-Leader:

1. Если у Ingress есть сертификат, создаётся FT 443 (HTTPS).
2. Если у Ingress нет сертификата, создаётся FT 80 (HTTP).

RULE

RULE — ресурс, определённый самим ALB. Он выполняет ту же роль, что и Ingress, но более специфичен. RULE уникально связан с FT.

RULE может создаваться ALB-Leader или пользователем вручную.

Случаи создания RULE ALB-Leader:

1. Синхронизация Ingress в RULE.

ALB Leader

Из нескольких экземпляров ALB выбирается один лидер. Лидер отвечает за:

1. Преобразование Ingress в Rules. Для каждого пути в Ingress создаётся Rule.
2. Создание необходимых FT для Ingress. Например, если у Ingress есть сертификат, создаётся FT 443 (HTTPS), если нет — FT 80 (HTTP).

Project

С точки зрения ALB, Project — это набор namespace.

В ALB можно настроить один или несколько Projects. При преобразовании Ingress в Rules ALB Leader игнорирует Ingress из namespace, не входящих в Project.

Дополнительные ресурсы:

- Настройка Load Balancer

Понимание MetalLB

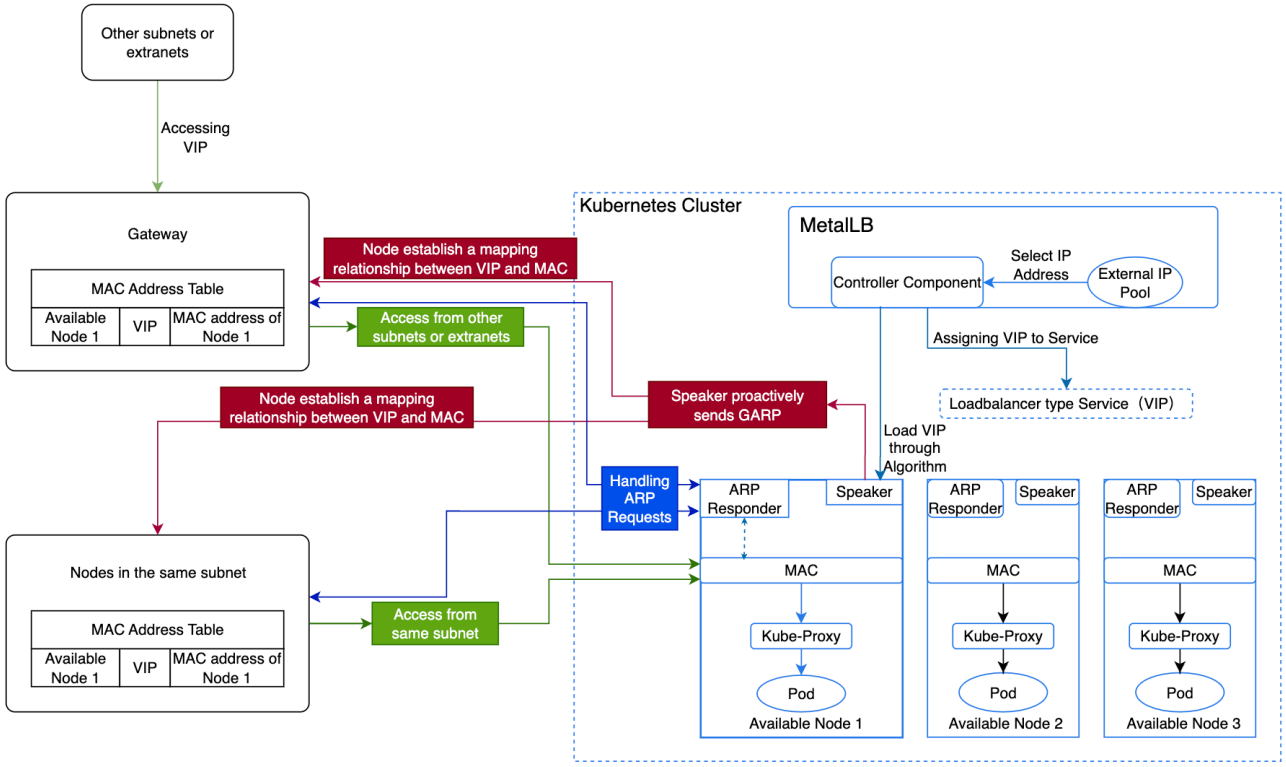
Содержание

Терминология

Термин	Описание
VIP	Виртуальный IP-адрес (VIP) — это IP-адрес, назначаемый MetalLB для внутренней маршрутизации типа LoadBalancer, обеспечивающий единый точку доступа для внешнего трафика к сервисам внутри кластера.
ARP	Протокол разрешения адресов (ARP) используется для сопоставления IP-адресов сетевого уровня с MAC-адресами канального уровня.
GARP	Gratuitous ARP (GARP) — это специальный ARP-запрос, используемый для информирования других узлов в сети о привязке IP-адреса к MAC-адресу. В отличие от обычных ARP-запросов,

Термин	Описание
	GARP не ожидает ответов, а активно рассылает информацию по сети.
ARP Responder	Компонент MetalLB, отвечающий на ARP-запросы, сопоставляя VIP с MAC-адресом узла. Когда узлу необходимо связаться с VIP, он отправляет ARP-запросы для получения MAC-адреса, соответствующего VIP. Каждый доступный узел имеет ARP Responder, который отвечает на эти запросы, сопоставляя VIP с MAC-адресом узла.
Controller	Компонент MetalLB, который динамически выделяет VIP из пула внешних адресов для внутренней маршрутизации типа LoadBalancer. Controller отслеживает события создания и удаления внутренних маршрутов в кластере для выделения или освобождения VIP по мере необходимости.
Speaker	Компонент MetalLB, который на основе политик или алгоритмов определяет, должны ли узлы размещать VIP и отправлять GARP. Он обеспечивает определённый уровень баланса между узлами, и при недоступности одного узла другие узлы могут взять на себя VIP и отправить GARP, обеспечивая тем самым высокую доступность.

Принципы высокой доступности в MetalLB



По умолчанию платформа использует ARP-режим MetalLB, а конкретный процесс реализации и принципы следующие:

- Компонент Controller MetalLB выбирает IP-адрес из пула внешних адресов и выделяет его для внутренней маршрутизации типа LoadBalancer в качестве VIP.
- MetalLB выбирает доступный узел для размещения VIP на основе алгоритма, который затем перенаправляет трафик.
- Компонент Speaker на этом узле активно отправляет GARP, устанавливая связь между VIP и MAC-адресом на всех узлах.
  - Узлы в одной подсети, узнав о сопоставлении VIP с MAC-адресом доступного узла, будут напрямую взаимодействовать с этим узлом при обращении к VIP.
  - Узлы в разных подсетях сначала направляют трафик на шлюз своей подсети, который затем перенаправляет трафик на узел, размещающий VIP.
- При сбое этого узла MetalLB выбирает другой доступный узел для размещения VIP, обеспечивая тем самым высокую доступность.
- Попав на узел, Kube-Proxy перенаправляет трафик соответствующему Pod.

Алгоритм MetalLB для выбора узлов-хозяев VIP

MetalLB хеширует все доступные узлы, соответствующие пулу внешних адресов, вместе с VIP и сортирует их по определённому алгоритму, выбирая первым доступным узлом для размещения VIP.

Пулы внешних адресов и количество узлов

Создайте пул внешних адресов и добавьте доступные узлы. Все доступные узлы поддерживают **резервные** отношения, то есть только узел, размещающий VIP, может перенаправлять трафик, и он должен обрабатывать весь трафик для VIP из пула внешних адресов.

Формула расчёта

Формула: **Количество пулов внешних адресов = ceil(n-vip / n-node)**, где ceil — округление вверх.

Примечание: При использовании виртуальных машин количество виртуальных машин = Количество пулов внешних адресов * n. Здесь n должно быть больше 2, при этом допускается отказ не более одного узла.

- n-vip: количество VIP.
- n-node: количество VIP, которое может обслуживать один узел.

Пример применения

Если в компании 10 VIP, и каждый доступный узел может обслуживать 5 VIP, при этом допускается отказ одного узла, как компании спланировать количество пулов внешних адресов и доступных узлов?

Анализ:

Требуется всего два пула внешних адресов и четыре доступных узла.

- Каждый доступный узел может обслуживать максимум 5 VIP, значит один пул внешних адресов может вместить 5 VIP, следовательно, для 10 VIP требуется два пула внешних адресов.
- При допущении отказа одного узла каждый пул должен включать один узел, размещающий VIP, и один резервный узел, что даёт по два доступных узла для каждого из двух пулов внешних адресов.

Дополнительные ресурсы

- Создание пула внешних IP-адресов
- Создание BGP-пиров

Menu

Основные понятия

Auth

Основная концепция

Быстрый старт

Связанные аннотации Ingress

forward-auth

basic-auth

CR

Специальная аннотация Ingress для ALB

Другие функции, связанные с аутентификацией в Ingress-Nginx

Примечание: несовместимые моменты с Ingress-Nginx

Устранение неполадок

Совместимость аннотаций ingress-nginx

Основные понятия

Поддерживаемые аннотации ingress-nginx

TCP/HTTP Keepalive

Основная концепция

CRD

ModSecurity

Терминология

Процедура эксплуатации

Связанные объяснения

Пример конфигурации

Сравнение различных методов Ingress

Для L4 (TCP/UDP) трафика

Для L7 (HTTP/HTTPS) трафика

HTTP Redirect

Основная концепция

CRD

Аннотация Ingress

Редирект на уровне порта

Редирект на уровне правила

L4/L7 Таймаут

Основная концепция

CRD

Что означает таймаут

Аннотация Ingress

Таймаут на уровне порта

GatewayAPI

OTel

Auth

Содержание

Основная концепция

Что такое Auth

Auth — это механизм, который выполняет аутентификацию до того, как запрос попадёт к реальному сервису. Он позволяет обрабатывать аутентификацию на уровне ALB единообразно, без необходимости реализовывать логику аутентификации в каждом backend-сервисе.

Поддерживаемые методы аутентификации

ALB поддерживает два основных метода аутентификации:

1. **Forward Auth (Внешняя аутентификация)**
   - Отправка запроса во внешний сервис аутентификации для проверки личности пользователя
   - Сценарии применения: требуется сложная логика аутентификации, например OAuth, SSO и т.д.
   - Рабочий процесс:

1. Запрос пользователя поступает на ALB
2. ALB пересылает информацию для аутентификации в сервис аутентификации
3. Сервис аутентификации возвращает результат проверки
4. На основе результата аутентификации принимается решение о допуске к backend-сервису

2. **Basic Auth (Базовая аутентификация)**

- Простой механизм аутентификации на основе имени пользователя и пароля
- Сценарии применения: простое управление доступом, защита среды разработки
- Рабочий процесс:

1. Запрос пользователя поступает на ALB
2. ALB проверяет имя пользователя и пароль в запросе
3. Сравнивает с настроенной информацией для аутентификации
4. Если проверка успешна, запрос перенаправляется к backend-сервису

Способы настройки Auth

1. **Глобальная аутентификация**

- Настраивается на уровне ALB, применяется ко всем сервисам
- Настраивается в ALB или FT CR

2. **Аутентификация на уровне пути**

- Настраивается на конкретном пути Ingress
- Настраивается на конкретном Rule
- Может переопределять глобальную конфигурацию аутентификации

3. **Отключение аутентификации**

- Отключение аутентификации для конкретного пути
- Настраивается в Ingress с аннотацией: `alb.ingress.cpaas.io/auth-enable: "false"`
- Настраивается в Rule с помощью CR

Обработка результата аутентификации

- Успешная аутентификация: запрос будет перенаправлен к backend-сервису
- Ошибка аутентификации: возвращается ошибка 401 unauthorized
- Можно настроить поведение перенаправления после ошибки аутентификации (применимо к Forward Auth)

Быстрый старт

Настройка Basic Auth с ALB

Развёртывание ALB

```
cat <<EOF | kubectl apply -f -
apiVersion: crd.alauda.io/v2
kind: ALB2
metadata:
  name: auth
  namespace: cpaas-system
spec:
  config:
    networkMode: container
    projects:
    - ALL_ALL
    replicas: 1
    vip:
      enableLbSvc: false
  type: nginx
EOF
export ALB_IP=$(kubectl get pods -n cpaas-system -l service_name=alb2-auth -o
jsonpath='{.items[*].status.podIP}');echo $ALB_IP
```

Настройка Secret и Ingress

```
# echo "Zm9vOiRhcHIxJHFJQ05aNjFRJDJpb29pSlZVQU1tcHJxMjU4L0NoUDE=" | base64 -d #
foo:$apr1$qICNZ61Q$2iooiJVUAMmprq258/ChP1
# openssl passwd -apr1 -salt qICNZ61Q bar # $apr1$qICNZ61Q$2iooiJVUAMmprq258/ChP1

kubectl apply -f - <<'END'
apiVersion: v1
kind: Secret
metadata:
  name: auth-file
type: Opaque
data:
  auth: Zm9vOiRhcHIxJHFJQ05aNjFRJDJpb29pSlZVQU1tcHJxMjU4L0NoUDE=
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: auth-file
  annotations:
    "nginx.ingress.kubernetes.io/auth-type": "basic"
    "nginx.ingress.kubernetes.io/auth-secret": "default/auth-file"
    "nginx.ingress.kubernetes.io/auth-secret-type": "auth-file"
spec:
  rules:
  - http:
      paths:
      - path: /app-file
        pathType: Prefix
        backend:
          service:
            name: app-server
            port:
              number: 80
END
```

Проверка

```
# echo "Zm9vOiJhYXIi" | base64 -d # foo:bar
curl -v -X GET -H "Authorization: Basic Zm9vOmJhcg==" http://$ALB_IP:80/app-file #
должен вернуть 200
# неправильный пароль
curl -v -X GET -H "Authorization: Basic XXXXOmJhcg==" http://$ALB_IP:80/app-file #
должен вернуть 401
```

Связанные аннотации Ingress

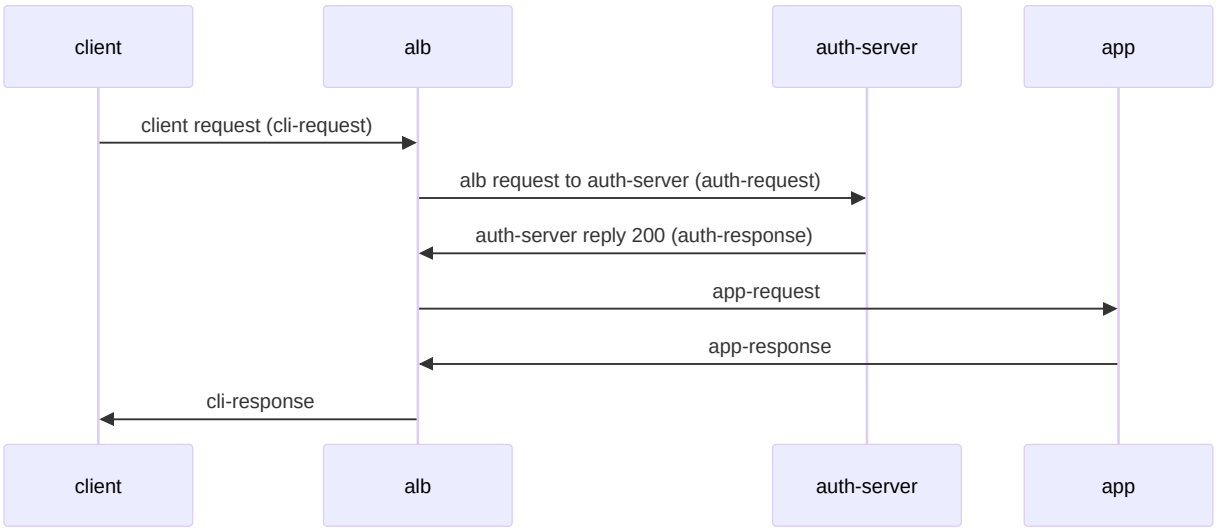
Ingress-nginx определяет ряд аннотаций для настройки конкретных деталей процесса аутентификации. Ниже приведён список аннотаций, поддерживаемых ALB, где "v" означает поддержку, а "x" — отсутствие поддержки.

	support	type	note
forward-auth			forward auth путём отправки http-запроса
nginx.ingress.kubernetes.io/auth-url	v	string	
nginx.ingress.kubernetes.io/auth-method	v	string	
nginx.ingress.kubernetes.io/auth-signin	v	string	
nginx.ingress.kubernetes.io/auth-signin-redirect-param	v	string	
nginx.ingress.kubernetes.io/auth-response-headers	v	string	
nginx.ingress.kubernetes.io/auth-proxy-set-headers	v	string	

	support	type	note
nginx.ingress.kubernetes.io/auth-request-redirect	v	string	
nginx.ingress.kubernetes.io/auth-always-set-cookie	v	boolean	
nginx.ingress.kubernetes.io/auth-snippet	x	string	
basic-auth			аутентификация по имени пользователя и паролю через secret
nginx.ingress.kubernetes.io/auth-realm	v	string	
nginx.ingress.kubernetes.io/auth-secret	v	string	
nginx.ingress.kubernetes.io/auth-secret-type	v	string	
nginx.ingress.kubernetes.io/auth-type	-	"basic" or "digest"	basic: поддерживается apr1 **digest: не поддерживается**
auth-cache			
nginx.ingress.kubernetes.io/auth-cache-key	x	string	
nginx.ingress.kubernetes.io/auth-cache-duration	x	string	
auth-keepalive			keepalive при отправке запроса.

	support	type	note
			Настройка keepalive через ряд аннотаций
nginx.ingress.kubernetes.io/auth-keepalive	x	number	
nginx.ingress.kubernetes.io/auth-keepalive-share-vars	x	"true" or "false"	
nginx.ingress.kubernetes.io/auth-keepalive-requests	x	number	
nginx.ingress.kubernetes.io/auth-keepalive-timeout	x	number	
auth-tls↗			при https-запросе дополнительная проверка сертификата
nginx.ingress.kubernetes.io/auth-tls-secret	x	string	
nginx.ingress.kubernetes.io/auth-tls-verify-depth	x	number	
nginx.ingress.kubernetes.io/auth-tls-verify-client	x	string	
nginx.ingress.kubernetes.io/auth-tls-error-page	x	string	
nginx.ingress.kubernetes.io/auth-tls-pass-certificate-to-upstream	x	"true" or "false"	
nginx.ingress.kubernetes.io/auth-tls-match-cn	x	string	

forward-auth



Связанные аннотации:

- nginx.ingress.kubernetes.io/auth-url
- nginx.ingress.kubernetes.io/auth-method
- nginx.ingress.kubernetes.io/auth-signin
- nginx.ingress.kubernetes.io/auth-signin-redirect-param
- nginx.ingress.kubernetes.io/auth-response-headers
- nginx.ingress.kubernetes.io/auth-proxy-set-headers
- nginx.ingress.kubernetes.io/auth-request-redirect
- nginx.ingress.kubernetes.io/auth-always-set-cookie

Эти аннотации описывают изменения, вносимые в auth-request, app-request и cli-response на диаграмме выше.

Конфигурация связанных аннотаций

auth-url

URL для auth-request, значение может быть переменной.

auth-method

Метод для auth-request.

auth-proxy-set-headers

Значение — ссылка на ConfigMap в формате `ns/name`. По умолчанию все заголовки из cli-request отправляются auth-server. Дополнительные заголовки можно настроить через proxy_set_header. По умолчанию отправляются следующие заголовки:

```
X-Original-URI          $request_uri;
X-Scheme                $pass_access_scheme;
X-Original-URL          $scheme://$http_host$request_uri;
X-Original-Method       $request_method;
X-Sent-From             "alb";
X-Real-IP               $remote_addr;
X-Forwarded-For         $proxy_add_x_forwarded_for;
X-Auth-Request-Redirect $request_uri;
```

Конфигурация аннотаций, связанных с app-request

auth-response-headers

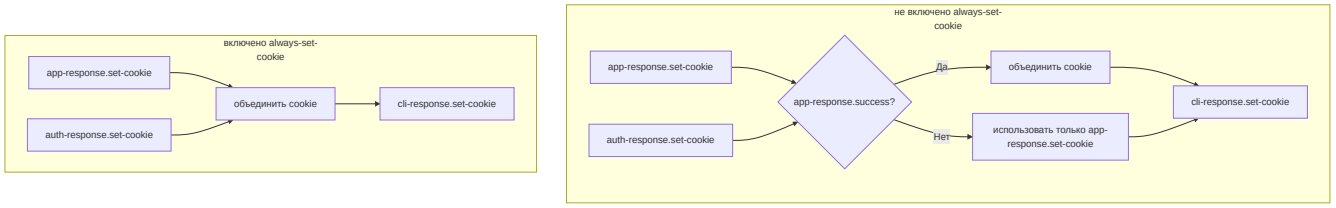
Значение — строка с разделёнными запятыми заголовками, позволяющая перенести конкретные заголовки из auth-response в app-request. пример:

```
nginx.ingress.kubernetes.io/auth-response-headers: Remote-User,Remote-Name
```

Когда ALB инициирует app-request, он включит Remote-User и Remote-Name из заголовков auth-response.

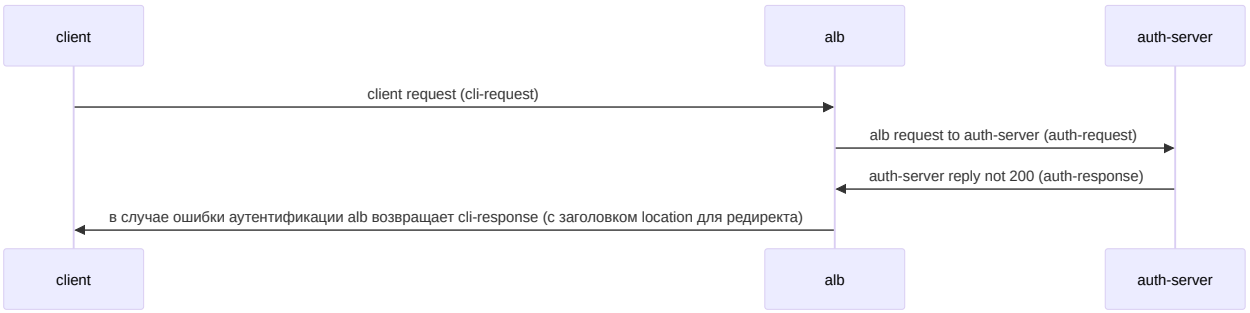
Обработка cookie

auth-response и app-response могут устанавливать cookie. По умолчанию, только если app-response.success, cookie из auth-response.set-cookie будут объединены с cli-response.set-cookie.



Конфигурация, связанная с Redirect sign

Когда auth-server возвращает 401, можно установить заголовок redirect в cli-response, чтобы браузер перенаправился на url, указанный в auth-signin, для прохождения проверки.



auth-signin

Значение — url, указывает заголовок location в cli-response.

auth-signin-redirect-param

Имя параметра запроса в signin-url, по умолчанию rd. Если signin-url не содержит параметр с именем, указанным в `auth-signin-redirect-param`, alb автоматически добавит этот параметр. Значение параметра будет установлено в `$pass_access_scheme://$http_host$escaped_request_uri`, чтобы сохранить исходный URL запроса.

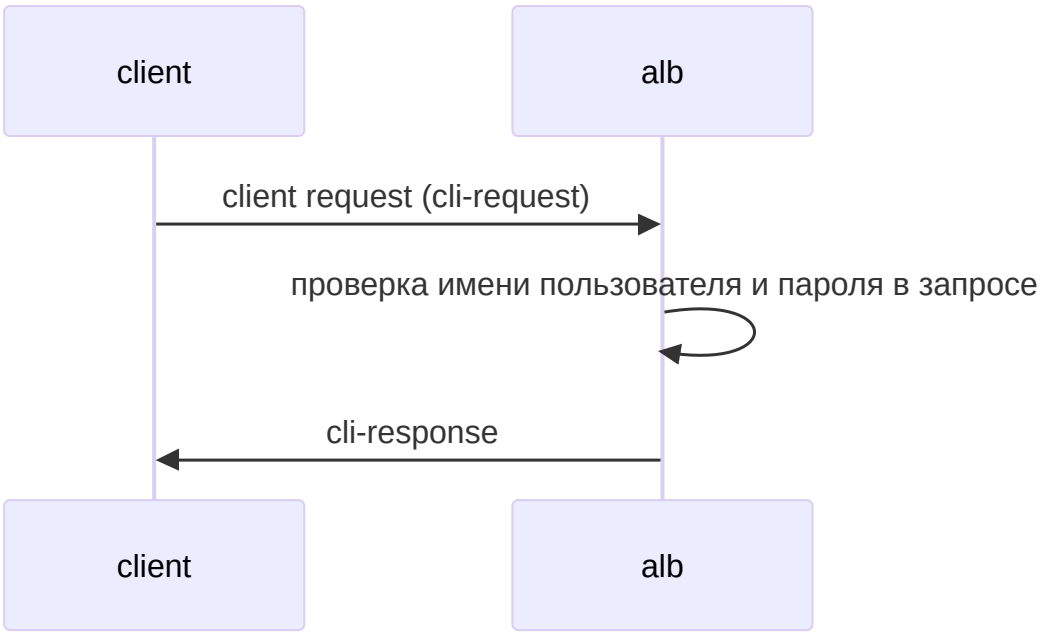
auth-request-redirect

Устанавливает заголовок `x-auth-request-redirect` в auth-request.

---

basic-auth

basic-auth — процесс аутентификации, описанный в RFC 7617↗. Процесс взаимодействия следующий:



auth-realm

описание защищённой области↗ Это значение realm в заголовке `WWW-Authenticate` cli-response. WWW-Authenticate: Basic realm="$realm"

auth-type

Тип схемы аутентификации, в настоящее время поддерживается только basic

auth-secret

Ссылка на secret с именем пользователя и паролем, формат ns/name

auth-secret-type

Secret поддерживает два типа:

1. auth-file: данные secret содержат только один ключ "auth", значение — строка в формате Apache htpasswd. Например:

```
data:
  auth: "user1:$apr1$xyz..."
```

2. auth-map: данные secret, где каждый ключ — имя пользователя, а соответствующее значение — хэш пароля (без имени пользователя в формате htpasswd). Например:

```
data:
  user1: "$apr1$xyz...."
  user2: "$apr1$abc...."
```

Примечание: в настоящее время поддерживаются только хэши паролей в формате htpasswd, сгенерированные алгоритмом apr1.

CR

В ALB CR добавлены конфигурационные параметры, связанные с auth, которые можно настроить в ALB/Frontend/Rule CR. Во время выполнения ALB преобразует аннотации на Ingress в правила.

```yaml
auth:
  # Конфигурация базовой аутентификации
  basic:
    #  string; соответствует nginx.ingress.kubernetes.io/auth-type: basic
    auth_type: "basic"
    #  string; соответствует nginx.ingress.kubernetes.io/auth-realm
    realm: "Restricted Access"
    #  string; соответствует nginx.ingress.kubernetes.io/auth-secret
    secret: "ns/name"
    #  string; соответствует nginx.ingress.kubernetes.io/auth-secret-type
    secret_type: "auth-map|auth-file"
  # Конфигурация Forward аутентификации
  forward:
    #  boolean; соответствует nginx.ingress.kubernetes.io/auth-always-set-cookie
    always_set_cookie: true
    #  string; соответствует nginx.ingress.kubernetes.io/auth-proxy-set-headers
    auth_headers_cm_ref: "ns/name"
    #  string; соответствует nginx.ingress.kubernetes.io/auth-request-redirect
    auth_request_redirect: "/login"
    #  string; соответствует nginx.ingress.kubernetes.io/auth-method
    method: "GET"
    #  string; соответствует nginx.ingress.kubernetes.io/auth-signin
    signin: "/signin"
    #  string; соответствует nginx.ingress.kubernetes.io/auth-signin-redirect-param
    signin_redirect_param: "redirect_to"
    #  []string; соответствует nginx.ingress.kubernetes.io/auth-response-headers
    upstream_headers:
      - "X-User-ID"
      - "X-User-Name"
      - "X-User-Email"
    #  string; соответствует nginx.ingress.kubernetes.io/auth-url
    url: "http://auth-service/validate"
```

Auth поддерживает настройку в:

- Alb CR в `.spec.config.auth`
- Frontend CR в `.spec.config.auth`
- Rule CR в `.spec.config.auth`

Порядок наследования: Alb > Frontend > Rule. Если дочерний CR не настроен, используется конфигурация родительского CR.

Специальная аннотация Ingress для ALB

При обработке Ingress ALB определяет приоритет на основе префикса аннотации. Приоритет от высокого к низкому:

- `index.$rule_index-$path_index.alb.ingress.cpaas.io`
- `alb.ingress.cpaas.io`
- `nginx.ingress.kubernetes.io`

Это позволяет решить проблему совместимости с ingress-nginx и задать конфигурацию auth для конкретного пути Ingress.

Auth-Enable

```
alb.ingress.cpaas.io/auth-enable: "false"
```

Новая аннотация, добавленная ALB, используется для указания, включена ли функция аутентификации для Ingress.

Другие функции, связанные с аутентификацией в Ingress-Nginx

Global-Auth

В ingress-nginx можно задать глобальную аутентификацию через ConfigMap. Это эквивалентно настройке auth для всех Ingress. В ALB можно настроить auth в ALB2 и FT CR. Правила под ними наследуют эти настройки.

No-Auth-Locations

В ALB можно отключить функцию auth для этого Ingress, настроив аннотацию: `alb.ingress.cpaas.io/auth-enable: "false"` в Ingress.

Примечание: несовместимые моменты с Ingress-Nginx

1. Не поддерживается auth-keepalive
2. Не поддерживается auth-snippet
3. Не поддерживается auth-cache
4. Не поддерживается auth-tls
5. Basic-auth поддерживает только basic, digest не поддерживается
6. Basic-auth basic поддерживает только алгоритм apr1, bcrypt, sha256 и другие не поддерживаются

Устранение неполадок

1. Проверьте логи контейнера Nginx в pod ALB
2. Проверьте заголовок `X-ALB-ERR-REASON` в ответе

Совместимость аннотаций ingress-nginx

Содержание

Основные понятия

ingress-nginx — это широко используемый Ingress Controller в Kubernetes, который определяет множество аннотаций для реализации различных функций, выходящих за рамки официального определения ingress.

Поддерживаемые аннотации ingress-nginx

Название	тип	Поддержка (v — поддерживается, x — не поддерживается, o — частично поддерживается или может быть реализовано через конфигурацию)
nginx.ingress.kubernetes.io/app-root	string	x

Название	тип	Поддержка (v — поддерживается, x — не поддерживается, o — частично поддерживается или может быть реализовано через конфигурацию)
nginx.ingress.kubernetes.io/affinity	cookie	o ingress не поддерживает. alb rule может настроить cookie hash
nginx.ingress.kubernetes.io/use-regex	bool	
nginx.ingress.kubernetes.io/affinity-mode	"balanced" или "persistent"	o ingress не поддерживает. alb rule может настроить session persistence
nginx.ingress.kubernetes.io/affinity-canary-behavior	"sticky" или "legacy"	o ingress не поддерживает. alb rule может настроить session persistence
nginx.ingress.kubernetes.io/auth-realm	string	v auth
nginx.ingress.kubernetes.io/auth-secret	string	v auth
nginx.ingress.kubernetes.io/auth-secret-type	string	v auth
nginx.ingress.kubernetes.io/auth-type	"basic" или "digest"	v auth
nginx.ingress.kubernetes.io/auth-tls-secret	string	x
nginx.ingress.kubernetes.io/auth-tls-verify-depth	number	x

Название	тип	Поддержка (v — поддерживается, x — не поддерживается, o — частично поддерживается или может быть реализовано через конфигурацию)
nginx.ingress.kubernetes.io/auth-tls-verify-client	string	x
nginx.ingress.kubernetes.io/auth-tls-error-page	string	x
nginx.ingress.kubernetes.io/auth-tls-pass-certificate-to-upstream	"true" или "false"	x
nginx.ingress.kubernetes.io/auth-tls-match-cn	string	x
nginx.ingress.kubernetes.io/auth-url	string	v
nginx.ingress.kubernetes.io/auth-cache-key	string	x
nginx.ingress.kubernetes.io/auth-cache-duration	string	x
nginx.ingress.kubernetes.io/auth-keepalive	number	x
nginx.ingress.kubernetes.io/auth-keepalive-share-vars	"true" или "false"	x
nginx.ingress.kubernetes.io/auth-keepalive-requests	number	x
nginx.ingress.kubernetes.io/auth-keepalive-timeout	number	x
nginx.ingress.kubernetes.io/auth-proxy-set-headers	string	v

Название	тип	Поддержка (v — поддерживается, x — не поддерживается, o — частично поддерживается или может быть реализовано через конфигурацию)
nginx.ingress.kubernetes.io/auth-snippet	string	x
nginx.ingress.kubernetes.io/enable-global-auth	"true" или "false"	o auth
nginx.ingress.kubernetes.io/backend-protocol	string	v
nginx.ingress.kubernetes.io/canary	"true" или "false"	x
nginx.ingress.kubernetes.io/canary-by-header	string	x
nginx.ingress.kubernetes.io/canary-by-header-value	string	x
nginx.ingress.kubernetes.io/canary-by-header-pattern	string	x
nginx.ingress.kubernetes.io/canary-by-cookie	string	x
nginx.ingress.kubernetes.io/canary-weight	number	x
nginx.ingress.kubernetes.io/canary-weight-total	number	x
nginx.ingress.kubernetes.io/client-body-buffer-size	string	x

Название	тип	Поддержка (v — поддерживается, x — не поддерживается, o — частично поддерживается или может быть реализовано через конфигурацию)
nginx.ingress.kubernetes.io/configuration-snippet	string	x
nginx.ingress.kubernetes.io/custom-http-errors	[]int	x
nginx.ingress.kubernetes.io/custom-headers	string	o
nginx.ingress.kubernetes.io/default-backend	string	o можно использовать default-backend ingress
nginx.ingress.kubernetes.io/enable-cors	"true" или "false"	v
nginx.ingress.kubernetes.io/cors-allow-origin	string	v
nginx.ingress.kubernetes.io/cors-allow-methods	string	v
nginx.ingress.kubernetes.io/cors-allow-headers	string	v
nginx.ingress.kubernetes.io/cors-expose-headers	string	x
nginx.ingress.kubernetes.io/cors-allow-credentials	"true" или "false"	x
nginx.ingress.kubernetes.io/cors-max-age	number	x

Название	тип	Поддержка (v — поддерживается, x — не поддерживается, o — частично поддерживается или может быть реализовано через конфигурацию)
nginx.ingress.kubernetes.io/force-ssl-redirect	"true" или "false"	v redirect
nginx.ingress.kubernetes.io/from-to-www-redirect	"true" или "false"	x
nginx.ingress.kubernetes.io/http2-push-preload	"true" или "false"	x
nginx.ingress.kubernetes.io/limit-connections	number	x
nginx.ingress.kubernetes.io/limit-rps	number	x
nginx.ingress.kubernetes.io/global-rate-limit	number	x
nginx.ingress.kubernetes.io/global-rate-limit-window	duration	x
nginx.ingress.kubernetes.io/global-rate-limit-key	string	x
nginx.ingress.kubernetes.io/global-rate-limit-ignored-cidrs	string	x
nginx.ingress.kubernetes.io/permanent-redirect	string	v redirect
nginx.ingress.kubernetes.io/permanent-redirect-code	number	v redirect

Название	тип	Поддержка (v — поддерживается, x — не поддерживается, o — частично поддерживается или может быть реализовано через конфигурацию)
nginx.ingress.kubernetes.io/temporal-redirect	string	v redirect
nginx.ingress.kubernetes.io/preserve-trailing-slash	"true" или "false"	x
nginx.ingress.kubernetes.io/proxy-body-size	string	x
nginx.ingress.kubernetes.io/proxy-cookie-domain	string	x
nginx.ingress.kubernetes.io/proxy-cookie-path	string	x
nginx.ingress.kubernetes.io/proxy-connect-timeout	number	v timeout
nginx.ingress.kubernetes.io/proxy-send-timeout	number	v timeout
nginx.ingress.kubernetes.io/proxy-read-timeout	number	v timeout
nginx.ingress.kubernetes.io/proxy-next-upstream	string	x
nginx.ingress.kubernetes.io/proxy-next-upstream-timeout	number	x
nginx.ingress.kubernetes.io/proxy-next-upstream-tries	number	x

Название	тип	Поддержка (v — поддерживается, x — не поддерживается, o — частично поддерживается или может быть реализовано через конфигурацию)
nginx.ingress.kubernetes.io/proxy-request-buffering	string	x
nginx.ingress.kubernetes.io/proxy-redirect-from	string	x
nginx.ingress.kubernetes.io/proxy-redirect-to	string	x
nginx.ingress.kubernetes.io/proxy-http-version	"1.0" или "1.1"	x
nginx.ingress.kubernetes.io/proxy-ssl-secret	string	x
nginx.ingress.kubernetes.io/proxy-ssl-ciphers	string	x
nginx.ingress.kubernetes.io/proxy-ssl-name	string	x
nginx.ingress.kubernetes.io/proxy-ssl-protocols	string	x
nginx.ingress.kubernetes.io/proxy-ssl-verify	string	x
nginx.ingress.kubernetes.io/proxy-ssl-verify-depth	number	x
nginx.ingress.kubernetes.io/proxy-ssl-server-name	string	x
nginx.ingress.kubernetes.io/enable-rewrite-log	"true" или "false"	x

Название	тип	Поддержка (v — поддерживается, x — не поддерживается, o — частично поддерживается или может быть реализовано через конфигурацию)
nginx.ingress.kubernetes.io/rewrite-target	URI	v
nginx.ingress.kubernetes.io/satisfy	string	x
nginx.ingress.kubernetes.io/server-alias	string	x
nginx.ingress.kubernetes.io/server-snippet	string	x
nginx.ingress.kubernetes.io/service-upstream	"true" или "false"	x
nginx.ingress.kubernetes.io/session-cookie-change-on-failure	"true" или "false"	x
nginx.ingress.kubernetes.io/session-cookie-conditional-samesite-none	"true" или "false"	x
nginx.ingress.kubernetes.io/session-cookie-domain	string	x
nginx.ingress.kubernetes.io/session-cookie-expires	string	x
nginx.ingress.kubernetes.io/session-cookie-max-age	string	x
nginx.ingress.kubernetes.io/session-cookie-name	string	x
nginx.ingress.kubernetes.io/session-cookie-path	string	x

Название	тип	Поддержка (v — поддерживается, x — не поддерживается, o — частично поддерживается или может быть реализовано через конфигурацию)
nginx.ingress.kubernetes.io/session-cookie-samesite	string	x
nginx.ingress.kubernetes.io/session-cookie-secure	string	x
nginx.ingress.kubernetes.io/ssl-redirect	"true" или "false"	v
nginx.ingress.kubernetes.io/ssl-passthrough	"true" или "false"	x
nginx.ingress.kubernetes.io/stream-snippet	string	x
nginx.ingress.kubernetes.io/upstream-hash-by	string	x
nginx.ingress.kubernetes.io/x-forwarded-prefix	string	x
nginx.ingress.kubernetes.io/load-balance	string	x
nginx.ingress.kubernetes.io/upstream-vhost	string	v
nginx.ingress.kubernetes.io/denylist-source-range	CIDR	o можно добиться похожего эффекта через modsecurity
nginx.ingress.kubernetes.io/whitelist-source-range	CIDR	o можно добиться похожего эффекта через modsecurity

Название	тип	Поддержка (v — поддерживается, x — не поддерживается, o — частично поддерживается или может быть реализовано через конфигурацию)
nginx.ingress.kubernetes.io/proxy-buffering	string	x
nginx.ingress.kubernetes.io/proxy-buffers-number	number	x
nginx.ingress.kubernetes.io/proxy-buffer-size	string	x
nginx.ingress.kubernetes.io/proxy-max-temp-file-size	string	x
nginx.ingress.kubernetes.io/ssl-ciphers	string	x
nginx.ingress.kubernetes.io/ssl-prefer-server-ciphers	"true" или "false"	x
nginx.ingress.kubernetes.io/connection-proxy-header	string	x
nginx.ingress.kubernetes.io/enable-access-log	"true" или "false"	o по умолчанию включен access_log, формат фиксирован
nginx.ingress.kubernetes.io/enable-opentelemetry	"true" или "false"	v otel
nginx.ingress.kubernetes.io/opentelemetry-trust-incoming-span	"true" или "false"	v otel
nginx.ingress.kubernetes.io/enable-modsecurity	bool	v modsecurity

Название	тип	Поддержка (v — поддерживается, x — не поддерживается, o — частично поддерживается или может быть реализовано через конфигурацию)
nginx.ingress.kubernetes.io/enable-owasp-core-rules	bool	v modsecurity
nginx.ingress.kubernetes.io/modsecurity-transaction-id	string	v modsecurity
nginx.ingress.kubernetes.io/modsecurity-snippet	string	v modsecurity
nginx.ingress.kubernetes.io/mirror-request-body	string	x
nginx.ingress.kubernetes.io/mirror-target	string	x
nginx.ingress.kubernetes.io/mirror-host	string	x

TCP/HTTP Keepalive

Содержание

Основная концепция

1. ALB поддерживает конфигурацию keepalive на уровне порта. Она может быть настроена на frontend.
2. Keepalive работает **между клиентом и ALB, а не между ALB и backend**.
3. Реализуется через конфигурацию Nginx, и Nginx **требует и автоматически перезагружается** при изменении конфигурации.
4. TCP keepalive и HTTP keepalive — это два разных понятия:

    1. **TCP keepalive** — это функция TCP-протокола, которая периодически отправляет probe-пакеты для проверки, жива ли связь, когда нет передачи данных. Это помогает обнаруживать и очищать мертвые соединения.

    2. **HTTP keepalive** (также известный как persistent connections) позволяет нескольким HTTP-запросам использовать одно и то же TCP-соединение, избегая накладных расходов на установку новых соединений. Это улучшает производительность за счёт снижения задержек и использования ресурсов.

CRD

```
keepalive:
  properties:
    http:
      description: Downstream L7 keepalive
      properties:
        header_timeout:
          description: Keepalive header timeout. Default is not set.
          type: string
        requests:
          description: Keepalive requests. Default is 1000.
          type: integer
        timeout:
          description: Keepalive timeout. Default is 75s.
          type: string
      type: object
    tcp:
      description: TCPKeepAlive defines TCP keepalive parameters (SO_KEEPALIVE)
      properties:
        count:
          description: The TCP_KEEPCNT socket option.
          type: integer
        idle:
          description: The TCP_KEEPIDLE socket option.
          type: string
        interval:
          description: The TCP_KEEPINTVL socket option.
          type: string
      type: object
  type: object
```

Её можно настроить только на Frontend в `.spec.config.keepalive`.

ModSecurity

ModSecurity — это открытый Web Application Firewall (WAF), предназначенный для защиты веб-приложений от вредоносных атак. Он поддерживается сообществом с открытым исходным кодом и поддерживает различные языки программирования и веб-серверы. Платформа Load Balancer (ALB) поддерживает настройку ModSecurity, позволяя создавать индивидуальные конфигурации на уровне Ingress.

Содержание

Терминология

Термин	Объяснение
owasp-core-rules	OWASP Core Rule Set — это набор правил с открытым исходным кодом, используемый для обнаружения и предотвращения распространённых атак на веб-приложения.

Процедура эксплуатации

Настройте ModSecurity, добавив аннотации в YAML-файл соответствующего ресурса или настроив CR.

Метод первый: Добавление аннотаций

Добавьте следующие аннотации в поле metadata.annotations соответствующего YAML-файла для настройки ModSecurity.

- **Аннотации, совместимые с Ingress-Nginx**

Аннотация	Тип	Применимый объект	Объясне
nginx.ingress.kubernetes.io/enable-modsecurity	bool	Ingress	Включить ModSecu
nginx.ingress.kubernetes.io/enable-owasp-core-rules	bool	Ingress	Включить OWASP C Rule Set.
nginx.ingress.kubernetes.io/modsecurity-transaction-id	string	Ingress	Использу для идентиф уникальн транзакц каждого запроса, помогает логирова отладке.
nginx.ingress.kubernetes.io/modsecurity-snippet	string	Ingress, ALB, FT, Rule	Позволяе пользова вставлят собствен

Аннотация	Тип	Применимый объект	Объясне
			конфигур ModSecu для удовлетв специфи требован безопасн

- **Специальные аннотации ALB**

Аннотация	Тип	Применимый объект	Объяснение
alb.modsecurity.cpaas.io/use-recommend	bool	Ingress	Включить или отключить рекомендуемые правила ModSecurity; установите `true` для применения предопределённого набора правил безопасности.
alb.modsecurity.cpaas.io/cmref	string	Ingress	Ссылка на конкретные конфигурации, например, можно загрузить пользовательские настройки безопасности, указав путь ссылки на ConfigMap (`$ns/$name#$section`)

Метод второй: Настройка CR

1. Откройте файл конфигурации ALB, FT или Rule, который необходимо настроить.
2. Добавьте следующие поля в spec.config по необходимости.

```
{ "modsecurity": {
      "enable": true, # Включить или отключить ModSecurity
      "transactionId": "$xx", # Использовать ID из Nginx
      "useCoreRules": true, # Добавить modsecurity_rules_file /etc/nginx/owasp-
modsecurity-crs/nginx-modsecurity.conf
      "useRecommend": true, # Добавить modsecurity_rules_file
/etc/nginx/modsecurity/modsecurity.conf
      "cmRef": "$ns/$name#$section", # Добавить конфигурацию из ConfigMap
    } }
```

3. Сохраните и примените файл конфигурации.

Связанные объяснения

Переопределение

Если ModSecurity не настроен в Rule, будет предпринята попытка найти конфигурацию в FT; если в FT конфигурация отсутствует, будет использована конфигурация из ALB.

Пример конфигурации

В следующем примере развёртывается ALB с именем `waf-alb` и демонстрационное backend-приложение с именем `hello`. Кроме того, развёртывается Ingress с именем `ing-waf-enable`, который определяет маршрут `/waf-enable` и настраивает правила ModSecurity. Любой запрос, содержащий параметр запроса `test`, значение которого включает строку `test`, будет заблокирован.

```
cat <<EOF | kubectl apply -f -
apiVersion: crd.alauda.io/v2
kind: ALB2
metadata:
  name: waf-alb
  namespace: cpaas-system
spec:
  config:
    loadbalancerName: waf-alb
    projects:
      - ALL_ALL
    replicas: 1
  type: nginx
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/enable-modsecurity: "true"
    nginx.ingress.kubernetes.io/modsecurity-transaction-id: "$request_id"
    nginx.ingress.kubernetes.io/modsecurity-snippet: |
      SecRuleEngine On
      SecRule ARGS:test "@contains test" "id:1234,deny,log"
  name: ing-waf-enable
spec:
  ingressClassName: waf-alb
  rules:
    - http:
        paths:
          - backend:
              service:
                name: hello
                port:
                  number: 80
            path: /waf-enable
            pathType: ImplementationSpecific
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ing-waf-normal
spec:
  ingressClassName: waf-alb
```

```
  rules:
    - http:
        paths:
          - backend:
              service:
                name: hello
                port:
                  number: 80
            path: /waf-not-enable
            pathType: ImplementationSpecific
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello
spec:
  replicas: 1
  selector:
    matchLabels:
      service.cpaas.io/name: hello
      service_name: hello
  template:
    metadata:
      labels:
        service.cpaas.io/name: hello
        service_name: hello
    spec:
      containers:
        - name: hello-world
          image: docker.io/hashicorp/http-echo
          imagePullPolicy: IfNotPresent
---
apiVersion: v1
kind: Service
metadata:
  name: hello
spec:
  internalTrafficPolicy: Cluster
  ipFamilies:
    - IPv4
  ipFamilyPolicy: SingleStack
  ports:
    - name: http
      port: 80
```

```
      protocol: TCP
      targetPort: 5678
  selector:
    service_name: hello
  sessionAffinity: None
  type: ClusterIP
EOF
```

Сравнение различных методов Ingress

Платформа Alauda Container Platform поддерживает несколько спецификаций ingress-трафика в экосистеме Kubernetes.
В этом документе проводится их сравнение (Service, Ingress, Gateway API и ALB Rule), чтобы помочь пользователям сделать правильный выбор.

Содержание

Для L4 (TCP/UDP) трафика

Сервисы типа LoadBalancer, Gateway API и ALB Rules могут все обеспечивать внешний доступ к L4 трафику. Здесь мы рекомендуем использовать подход с сервисом типа LoadBalancer.
И Gateway API, и ALB Rules реализованы через ALB, который является прокси в пространстве пользователя, и их производительность значительно снижается при обработке L4 трафика по сравнению с сервисами типа LoadBalancer.

Для L7 (HTTP/HTTPS) трафика

Хотя Ingress, GatewayAPI и ALB Rules все могут обеспечивать внешний доступ к L7 трафику, они отличаются по возможностям и моделям изоляции.

Ingress

Ingress — это стандартная спецификация, принятая сообществом Kubernetes, и рекомендуется для использования по умолчанию.
Ingress обрабатывается экземплярами ALB, которыми управляет администратор платформы.

GatewayAPI

GatewayAPI предоставляет более гибкий режим изоляции, однако он менее зрелый, чем Ingress.
Используя GatewayAPI, разработчик может создавать собственные изолированные экземпляры ALB для обработки правил GatewayAPI.
Поэтому, если вам нужно делегировать создание и управление экземплярами ALB разработчикам, следует выбрать GatewayAPI.

ALB Rule

ALB Rule (Load Balancer в UI) предоставляет наиболее гибкие правила сопоставления трафика и максимальные возможности. Фактически, и Ingress, и GatewayAPI реализуются путем преобразования их в ALB Rules.
Однако ALB Rule сложнее, чем Ingress и GatewayAPI, и не является стандартным API сообщества. Поэтому мы рекомендуем использовать его только в тех случаях, когда Ingress и GatewayAPI не удовлетворяют вашим требованиям.

HTTP Redirect

Содержание

Основная концепция

HTTP redirect — это функция, предоставляемая ALB. Она напрямую возвращает HTTP-код 30x для запроса, который соответствует правилу. Заголовок Location используется для указания клиенту перенаправиться на новый URL.

ALB поддерживает настройку редиректа на уровне порта и правила.

CRD

```
redirect:
  properties:
    code:
      type: integer
    host:
      type: string
    port:
      type: integer
    prefix_match:
      type: string
    replace_prefix:
      type: string
    scheme:
      type: string
    url:
      type: string
  type: object
```

Редирект может быть настроен на:

- Frontend: `.spec.config.redirect`
- Rule: `.spec.config.redirect`

Аннотация Ingress

Аннотация	Описание
nginx.ingress.kubernetes.io/permanent-redirect	Соответствует URL в CR, по умолчанию устанавливает код 301
nginx.ingress.kubernetes.io/permanent-redirect-code	Соответствует code в CR
nginx.ingress.kubernetes.io/temporal-redirect	Соответствует URL в CR, по умолчанию устанавливает код 302

Аннотация	Описание
nginx.ingress.kubernetes.io/temporal-redirect-code	Соответствует code в CR
nginx.ingress.kubernetes.io/ssl-redirect	Соответствует scheme в CR, по умолчанию устанавливает scheme HTTPS
nginx.ingress.kubernetes.io/force-ssl-redirect	Соответствует scheme в CR, по умолчанию устанавливает scheme HTTPS

SSL-Redirect

1. SSL-redirect и force-ssl-redirect отличаются тем, что SSL-redirect действует только если у ingress есть сертификат для соответствующего домена, тогда как force-ssl-redirect действует независимо от наличия сертификата.
2. Для HTTPS-портов, если настроен только SSL-redirect, редирект не будет установлен.

Редирект на уровне порта

Когда редирект настроен на уровне порта, *все запросы* к этому порту будут перенаправлены согласно конфигурации редиректа.

Редирект на уровне правила

Когда редирект настроен на уровне правила, запросы, соответствующие этому правилу, будут перенаправлены согласно конфигурации редиректа.

L4/L7 Таймаут

Содержание

Основная концепция

L4/L7 таймаут — это функция, предоставляемая ALB. Она используется для настройки времени таймаута для L4/L7 прокси.

Таймаут реализован с помощью Lua-скрипта, и при его изменении **не требуется перезагрузка** Nginx.

CRD

```
timeout:
  properties:
    proxy_connect_timeout_ms:
      type: integer
    proxy_read_timeout_ms:
      type: integer
    proxy_send_timeout_ms:
      type: integer
  type: object
```

Конфигурация может быть задана в:

- Frontend: `.spec.config.timeout`
- Rule: `.spec.config.timeout`

Что означает таймаут

Существует три типа таймаутов:

1. **proxy_connect_timeout_ms**: Определяет время таймаута для установления соединения с upstream-сервером. Если соединение не будет установлено в течение этого времени, запрос завершится с ошибкой.
2. **proxy_read_timeout_ms**: Определяет время таймаута для чтения ответа от upstream-сервера. Таймаут устанавливается между двумя последовательными операциями чтения, а не на весь ответ целиком. Если в течение этого времени данные не поступают, соединение закрывается.
3. **proxy_send_timeout_ms**: Определяет время таймаута для отправки запроса upstream-серверу. Аналогично таймауту чтения, он устанавливается между двумя последовательными операциями записи. Если в течение этого времени данные не могут быть отправлены, соединение закрывается.

Аннотация Ingress

Аннотация	Описание
nginx.ingress.kubernetes.io/proxy-connect-timeout	Соответствует proxy_connect_timeout_ms в CRD
nginx.ingress.kubernetes.io/proxy-read-timeout	Соответствует proxy_read_timeout_ms в CRD
nginx.ingress.kubernetes.io/proxy-send-timeout	Соответствует proxy_send_timeout_ms в CRD

Таймаут на уровне порта

Вы можете настроить таймаут непосредственно на порту, который используется как таймаут L4.

Menu

GatewayAPI

GatewayAPI ↗ — это новый стандарт для Kubernetes ingress.

ALB также поддерживает GatewayAPI. Каждый ресурс Gateway будет преобразован в ресурс ALB.

Listener и Router будут обрабатываться непосредственно в ALB. Они не будут преобразованы в `Frontend` и `Rule`.

OTel

OpenTelemetry (OTel) — это проект с открытым исходным кодом, направленный на предоставление независимого от поставщика стандарта для сбора, обработки и экспорта телеметрических данных в распределённых системах, таких как архитектуры микросервисов. Он помогает разработчикам проще анализировать производительность и поведение программного обеспечения, что облегчает диагностику и устранение проблем в приложениях.

Содержание

Терминология

Термин	Объяснение
Trace	Данные, отправляемые на OTel Server, представляющие собой набор связанных событий или операций, используемых для отслеживания потока запросов в распределённых системах; каждый Trace состоит из нескольких Spans.
Span	Независимая операция или событие внутри Trace, включающая время начала, продолжительность и другую релевантную информацию.
OTel Server	Сервер OTel, способный принимать и хранить данные Trace, например Jaeger, Prometheus и др.
Jaeger	Система распределённого трассирования с открытым исходным кодом, используемая для мониторинга и отладки архитектур микросервисов, поддерживающая интеграцию с OpenTelemetry.
Attributes	Пары ключ-значение, прикреплённые к Trace или Span для предоставления дополнительной контекстной информации. Включают Resource Attributes и Span Attributes; см. Attributes для подробностей.
Sampler	Компонент стратегии, определяющий, следует ли сэмплировать и отправлять Trace. Можно настроить различные стратегии сэмплирования, например полное сэмплирование, пропорциональное и др.
ALB (Another Load Balancer)	Программное или аппаратное устройство, распределяющее сетевые запросы между доступными узлами в кластере; балансировщик нагрузки (ALB), используемый на платформе, является программным балансировщиком уровня 7, который можно настроить для мониторинга трафика с помощью OTel. ALB поддерживает отправку Trace на указанный Collector и позволяет использовать различные стратегии сэмплирования; также поддерживает настройку отправки Trace на уровне Ingress.
FT (Frontend)	Конфигурация порта для ALB, задающая настройки на уровне порта.

Термин	Объяснение
Rule	Правила маршрутизации на порту (FT), используемые для сопоставления конкретных маршрутов.
HotROD (Rides on Demand)	Пример приложения, предоставляемый Jaeger для демонстрации использования распределённого трассирования; подробности см. в Hot R.O.D. - Rides on Demand ↗.
hotrod-with-proxy	Указывает адреса внутренних микросервисов HotROD через переменные окружения; подробности см. в hotrod-with-proxy ↗.

Предварительные требования

- **Убедитесь, что существует работоспособный ALB**: Создайте или используйте существующий ALB, имя которого в данном документе заменено на `<otel-alb>`. Инструкции по созданию ALB см. в разделе Creating Load Balancer.
- **Убедитесь, что имеется адрес сервера отчётов данных OTel**: Этот адрес далее будет называться `<jaeger-server>`.

Процедура

Обновление конфигурации ALB

1. На Master-узле кластера с помощью CLI выполните команду для редактирования конфигурации ALB.

```
kubectl edit alb2 -n cpaas-system <otel-alb> # Замените <otel-alb> на фактическое имя ALB
```

2. Добавьте следующие поля в раздел `spec.config`.

```
otel:
  enable: true
  exporter:
    collector:
      address: "<jaeger-server>" # Замените <jaeger-server> на фактический адрес сервера отчётов OTel
      request_timeout: 1000
```

Пример готовой конфигурации:

```
spec:
  address: 192.168.1.1
  config:
    otel:
     enable: true
     exporter:
       collector:
         address: "http://jaeger.default.svc.cluster.local:4318"
         request_timeout: 1000
    antiAffinityKey: system
    defaultSSLCert: cpaas-system/cpaas-system
    defaultSSLStrategy: Both
    gateway:
    ...
type: nginx
```

3. Выполните команду для сохранения изменений. После обновления ALB по умолчанию будет включён OpenTelemetry, и вся информация о Trace запросов будет отправляться на Jaeger Server.

```
:wq
```

Связанные операции

Настройка OTel в Ingress

- **Включение или отключение OTel на Ingress**

  Настройка включения или отключения OTel на Ingress позволяет лучше контролировать и отлаживать поток запросов приложений, выявляя узкие места производительности или ошибки, отслеживая запросы при их прохождении между различными сервисами.

  Процедура

  Добавьте следующую конфигурацию в поле metadata.annotations Ingress:

  ```
  nginx.ingress.kubernetes.io/enable-opentelemetry: "true"
  ```

  Объяснение параметра:

  - **nginx.ingress.kubernetes.io/enable-opentelemetry**: При значении `true` контроллер Ingress включает функциональность OpenTelemetry при обработке запросов через этот Ingress, то есть информация о Trace запросов будет собираться и отправляться. При значении `false` или отсутствии этой аннотации сбор и отправка Trace не выполняются.

- **Включение или отключение доверия OTel на Ingress**

  OTel Trust определяет, доверяет ли Ingress и использует ли Trace информацию (например, trace ID) из входящих запросов.

  Процедура

  Добавьте следующую конфигурацию в поле metadata.annotations Ingress:

  ```
  nginx.ingress.kubernetes.io/opentelemetry-trust-incoming-span: "true"
  ```

  Объяснение параметра:

  - **nginx.ingress.kubernetes.io/opentelemetry-trust-incoming-span**: При значении `true` Ingress продолжает использовать уже существующую Trace информацию, что помогает сохранять согласованность в межсервисном трассировании, позволяя полностью отслеживать и анализировать всю цепочку запросов в системе распределённого трассирования. При значении `false` для запроса

генерируется новая информация трассировки, что может привести к тому, что запрос будет рассматриваться как часть новой цепочки трассировки после прохождения через Ingress, прерывая непрерывность межсервисного трассирования.

- **Добавление различных конфигураций OTel на Ingress**

  Эта настройка позволяет кастомизировать поведение OTel и метод экспорта данных для разных ресурсов Ingress, обеспечивая тонкий контроль над стратегией трассирования или целями каждого сервиса.

  Процедура

  Добавьте следующую конфигурацию в поле metadata.annotations Ingress:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    alb.ingress.cpaas.io/otel: >
      {
         "enable": true,
         "exporter": {
             "collector": {
                 "address": "<jaeger-server>", # Замените <jaeger-server> на
фактический адрес сервера отчётов OTel, например "address":
"http://128.0.0.1:4318"
                 "request_timeout": 1000
             }
         }
      }
```

  Объяснение параметров:

  - **exporter**: Определяет, как собранные данные Trace отправляются в OTel Collector (сервер отчётов OTel).
  - **address**: Адрес OTel Collector.
  - **request_timeout**: Таймаут запроса.

Использование OTel в приложениях

Ниже приведена полная структура конфигурации OTel, которая может использоваться для определения включения и использования функций OTel в приложениях.

На Master-узле кластера с помощью CLI выполните команду для получения полной структуры конфигурации OTel.

```
kubectl get crd alaudaloadbalancer2.crd.alauda.io -o json|jq
".spec.versions[2].schema.openAPIV3Schema.properties.spec.properties.config.properties.otel"
```

Результат:

```
{
    "otel": {
        "enable": true
    }
    "exporter": {
        "collector": {
            "address": ""
          },
    },
    "flags": {
        "hide_upstream_attrs": false
        "notrust_incoming_span": false
        "report_http_request_header": false
        "report_http_response_header": false
    },
    "sampler": {
        "name": "",
        "options": {
            "fraction": ""
            "parent_name": ""
          },
      },
 }
```

Объяснение параметров:

Параметр	Описание
otel.enable	Включение функциональности OTel.
exporter.collector.address	Адрес сервера отчётов OTel, поддерживает протоколы http/https и доменные имена.
flags.hide_upstream_attrs	Отчёт об информации о правилах upstream.
flag.notrust_incoming_span	Доверие и использование Trace информации OTel (например, trace ID) из входящих запросов.
flags.report_http_request_header	Отчёт заголовков HTTP-запроса.
flags.report_http_response_header	Отчёт заголовков HTTP-ответа.
sampler.name	Название стратегии сэмплирования; подробности см. в разделе Sampling Strategies.
sampler.options.fraction	Доля сэмплирования.
sampler.options.parent_name	Родительская стратегия для стратегий parent_base.

Наследование

По умолчанию, если в ALB настроены определённые параметры OTel, а FT не настроен, FT наследует параметры от ALB как собственные; то есть FT наследует конфигурацию ALB, а Rule может наследовать конфигурации как от ALB, так и от FT.

- **ALB**: Конфигурация на уровне ALB обычно глобальная и по умолчанию. Здесь можно настроить глобальные параметры, например адреса Collector, которые будут унаследованы нижележащими FT и Rule.
- **FT**: FT может наследовать конфигурации от ALB, то есть параметры OTel, не настроенные на FT, будут использоваться из ALB. При этом FT можно детализировать, например, выборочно включать или отключать OTel на FT без влияния на другие FT или глобальные настройки ALB.

- **Rule**: Rule может наследовать конфигурации как от ALB, так и от FT. При этом Rule также можно детализировать, например, конкретное правило может не доверять входящей Trace информации OTel или изменять стратегии сэмплирования.

Процедура

Настройте поле `spec.config.otel` в YAML-файлах ALB, FT и Rule для добавления конфигураций, связанных с OTel.

Дополнительные сведения

Стратегии сэмплирования

Параметр	Объяснение
always on	Всегда отправлять все данные трассировки.
always off	Никогда не отправлять данные трассировки.
traceid-ratio	Решение об отправке принимается на основе `traceid`. Формат `traceparent` — `xx-traceid-xx-flag`, где первые 16 символов `traceid` представляют 32-битное шестнадцатеричное число. Если это число меньше `fraction`, умноженного на 4294967295 (то есть (2^32-1)), данные отправляются.
parent-base	Решение об отправке принимается на основе флага в traceparent запроса. При флаге 01 данные отправляются; например: `curl -v "http://$ALB_IP/" -H 'traceparent: 00-xx-xx-01'`; при флаге 02 данные не отправляются; например: `curl -v "http://$ALB_IP/" -H 'traceparent: 00-xx-xx-02'`.

Атрибуты

- **Resource Attributes**

  Эти атрибуты отправляются по умолчанию.

Параметр	Описание
hostname	Имя хоста Pod ALB
service.name	Имя ALB
service.namespace	Пространство имён, где расположен ALB
service.type	По умолчанию ALB
service.instance.id	Имя Pod ALB

- **Span Attributes**
  - Атрибуты, отправляемые по умолчанию:

Параметр	Описание
http.status_code	Код состояния
http.request.resend_count	Количество повторных попыток
alb.rule.rule_name	Имя правила, с которым совпал этот запрос
alb.rule.source_type	Тип правила, с которым совпал запрос, в настоящее время только Ingress
alb.rule.source_name	Имя Ingress
alb.rule.source_ns	Пространство имён, где расположен Ingress

  - Атрибуты, отправляемые по умолчанию, но которые можно исключить, изменяя поле flag.hide_upstream_attrs:

Параметр	Описание
alb.upstream.svc_name	Имя Service (внутреннего маршрута), на который перенаправляется трафик
alb.upstream.svc_ns	Пространство имён Service (внутреннего маршрута), на который перенаправляется трафик

Параметр	Описание
alb.upstream.peer	IP-адрес и порт Pod, на который перенаправляется трафик

- Атрибуты, не отправляемые по умолчанию, но которые можно включить, изменяя поле flag.report_http_request_header:

Параметр	Описание
`**http.request.header.<header>**`	Заголовок запроса

- Атрибуты, не отправляемые по умолчанию, но которые можно включить, изменяя поле flag.report_http_response_header:

Параметр	Описание
`**http.response.header.<header>**`	Заголовок ответа

Пример конфигурации

Ниже приведена YAML-конфигурация, которая разворачивает ALB и использует Jaeger в качестве сервера OTel, с Hotrod-proxy в качестве демонстрационного бэкенда. Настройка правил Ingress позволяет при обращении клиентов к ALB направлять трафик на HotROD. Кроме того, взаимодействие между внутренними микросервисами HotROD также маршрутизируется через ALB.

1. Сохраните следующий YAML в файл с именем all.yaml.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hotrod
spec:
  replicas: 1
  selector:
    matchLabels:
      service.cpaas.io/name: hotrod
      service_name: hotrod
  template:
    metadata:
      labels:
        service.cpaas.io/name: hotrod
        service_name: hotrod
    spec:
      containers:
        - name: hotrod
          env:
            - name: PROXY_PORT
              value: "80"
            - name: PROXY_ADDR
              value: "otel-alb.default.svc.cluster.local:"
            - name: OTEL_EXPORTER_OTLP_ENDPOINT
              value: "http://jaeger.default.svc.cluster.local:4318"
          image: theseedoaa/hotrod-with-proxy:latest
          imagePullPolicy: IfNotPresent
          command: ["/bin/hotrod", "all", "-v"]
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: hotrod-frontend
spec:
  ingressClassName: otel-alb
  rules:
    - http:
        paths:
          - backend:
              service:
                name: hotrod
                port:
                  number: 8080
```

```
            path: /dispatch
            pathType: ImplementationSpecific
          - backend:
              service:
                name: hotrod
                port:
                  number: 8080
            path: /frontend
            pathType: ImplementationSpecific
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: hotrod-customer
spec:
  ingressClassName: otel-alb
  rules:
    - http:
        paths:
          - backend:
              service:
                name: hotrod
                port:
                  number: 8081
            path: /customer
            pathType: ImplementationSpecific
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: hotrod-route
spec:
  ingressClassName: otel-alb
  rules:
    - http:
        paths:
          - backend:
              service:
                name: hotrod
                port:
                  number: 8083
            path: /route
            pathType: ImplementationSpecific
---
```

```
apiVersion: v1
kind: Service
metadata:
  name: hotrod
spec:
  internalTrafficPolicy: Cluster
  ipFamilies:
    - IPv4
  ipFamilyPolicy: SingleStack
  ports:
    - name: frontend
      port: 8080
      protocol: TCP
      targetPort: 8080
    - name: customer
      port: 8081
      protocol: TCP
      targetPort: 8081
    - name: router
      port: 8083
      protocol: TCP
      targetPort: 8083
  selector:
    service_name: hotrod
  sessionAffinity: None
  type: ClusterIP
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jaeger
spec:
  replicas: 1
  selector:
    matchLabels:
      service.cpaas.io/name: jaeger
      service_name: jaeger
  template:
    metadata:
      labels:
        service.cpaas.io/name: jaeger
        service_name: jaeger
    spec:
      containers:
```

```
        - name: jaeger
          env:
            - name: LOG_LEVEL
              value: debug
          image: jaegertracing/all-in-one:1.58.1
          imagePullPolicy: IfNotPresent
      hostNetwork: true
      tolerations:
        - operator: Exists
---
apiVersion: v1
kind: Service
metadata:
  name: jaeger
spec:
  internalTrafficPolicy: Cluster
  ipFamilies:
    - IPv4
  ipFamilyPolicy: SingleStack
  ports:
    - name: http
      port: 4318
      protocol: TCP
      targetPort: 4318
  selector:
    service_name: jaeger
  sessionAffinity: None
  type: ClusterIP
---
apiVersion: crd.alauda.io/v2
kind: ALB2
metadata:
  name: otel-alb
spec:
  config:
    loadbalancerName: otel-alb
    otel:
      enable: true
      exporter:
        collector:
          address: "http://jaeger.default.svc.cluster.local:4318"
          request_timeout: 1000
    projects:
      - ALL_ALL
```

```
    replicas: 1
    resources:
      alb:
        limits:
          cpu: 200m
          memory: 2Gi
        requests:
          cpu: 50m
          memory: 128Mi
      limits:
        cpu: "1"
        memory: 1Gi
      requests:
        cpu: 50m
        memory: 128Mi
  type: nginx
```

2. В CLI выполните команду для развертывания Jaeger, ALB, HotROD и всех необходимых CR для тестирования.

```
kubectl apply ./all.yaml
```

3. Выполните команду для получения адреса доступа к Jaeger.

```
export JAEGER_IP=$(kubectl get po -A -o wide |grep jaeger | awk '{print $7}');echo
"http://$JAEGER_IP:16686"
```

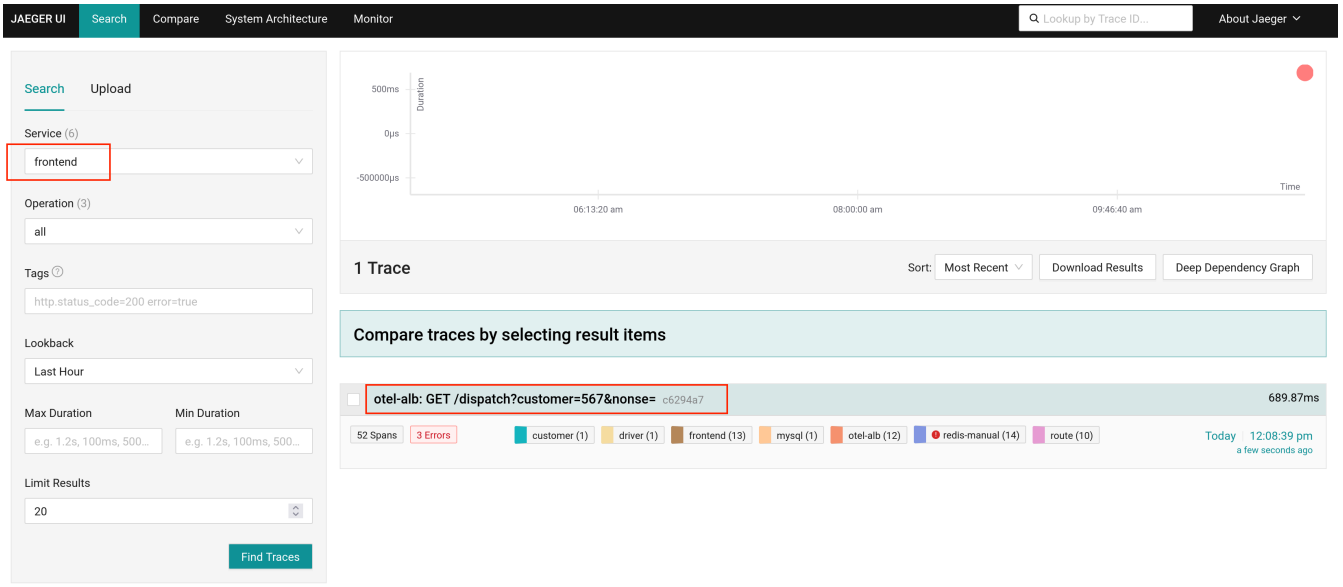
4. Выполните команду для получения адреса доступа к otel-alb.

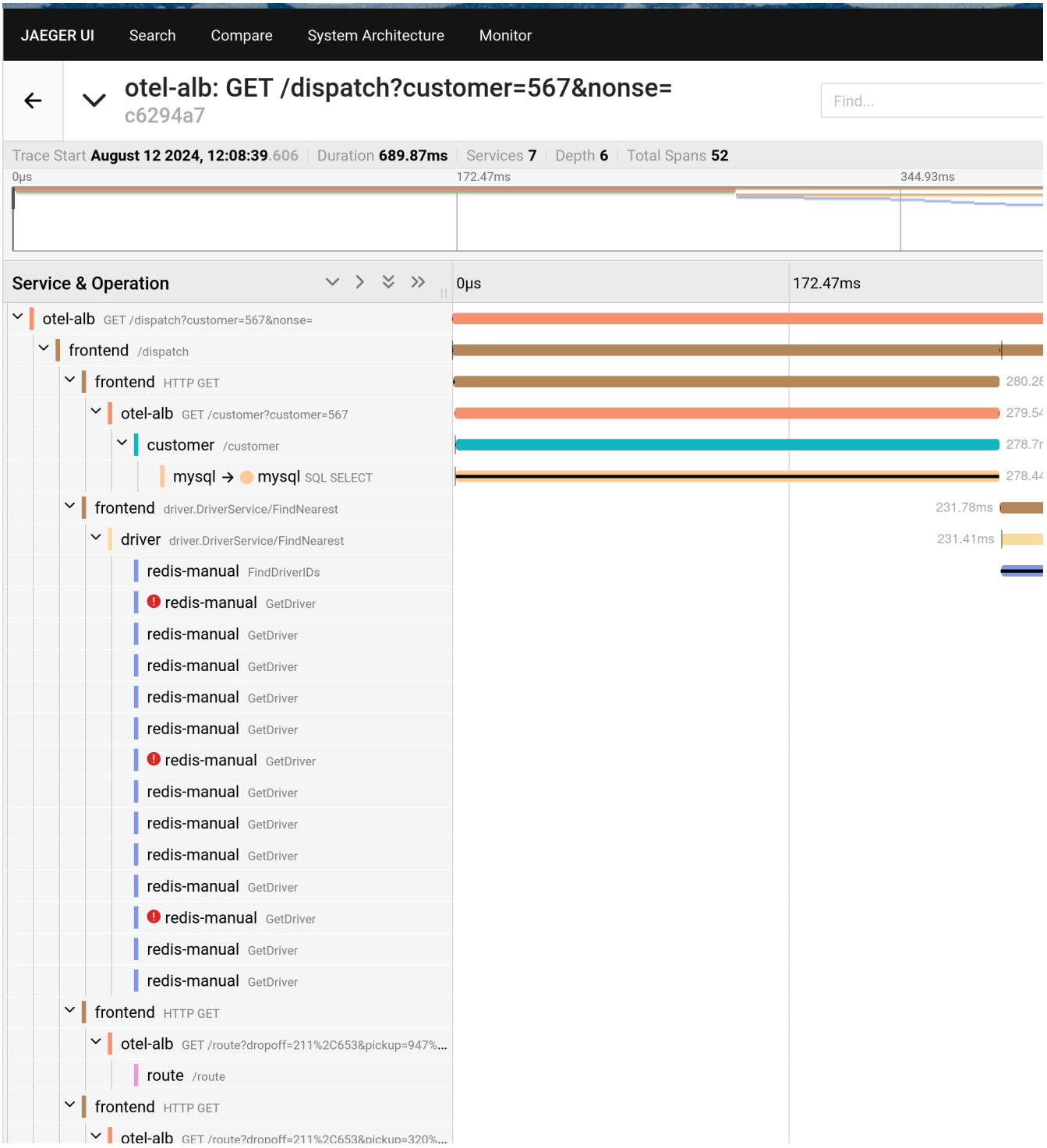
```
export ALB_IP=$(kubectl get po -A -o wide|grep otel-alb | awk '{print $7}');echo
$ALB_IP
```

5. Выполните команду для отправки запроса к HotROD через ALB. ALB при этом отправит Trace в Jaeger.

```
curl -v "http://<$ALB_IP>:80/dispatch?customer=567&nonse=" # Замените <$ALB_IP> в
команде на адрес доступа otel-alb, полученный на предыдущем шаге
```

6. Откройте адрес доступа Jaeger, полученный в Шаге 3, чтобы просмотреть результаты.




JAEGER UI
Search
Compare
System Architecture
Monitor
otel-alb: GET /dispatch?customer=567&nonse=
c6294a7
Find...
Trace Start August 12 2024, 12:08:39.606
Duration 689.87ms
Services 7
Depth 6
Total Spans 52
0µs
172.47ms
344.93ms
Service & Operation
0µs
172.47ms
otel-alb GET /dispatch?customer=567&nonse=
frontend /dispatch
frontend HTTP GET
otel-alb GET /customer?customer=567
customer /customer
mysql → mysql SQL SELECT
frontend driver.DriverService/FindNearest
231.78ms
driver driver.DriverService/FindNearest
231.41ms
redis-manual FindDriverIDs
redis-manual GetDriver
redis-manual GetDriver
redis-manual GetDriver
redis-manual GetDriver
redis-manual GetDriver
redis-manual GetDriver
redis-manual GetDriver
redis-manual GetDriver
redis-manual GetDriver
redis-manual GetDriver
redis-manual GetDriver
redis-manual GetDriver
redis-manual GetDriver
frontend HTTP GET
otel-alb GET /route?dropoff=211%2C653&pickup=947%...
route /route
frontend HTTP GET
otel-alb GET /route?dropoff=211%2C653&pickup=320%...

Menu

Руководства

Создание сервисов

Зачем нужен Service

Пример Service типа ClusterIP:

Headless Services

Создание сервиса через веб-консоль

Создание сервиса через CLI

Пример: Доступ к приложению внутри кластера

Пример: Доступ к приложению вне кластера

Пример: Service типа ExternalName

Аннотации для Service типа LoadBalancer

Создание Ingress

Метод реализации

Предварительные требования

Пример Ingress:

Создание Ingress через веб-консоль

Создание Ingress через CLI

Настройка Gateway

Создание доменного имени

Создание сертификатов

Создание пула внешних IP-адресов

Создание BGP-пиров

Настройка подсетей

Настройка сетевых политик

Создание Admin Network Policies

Настройка сетевых политик кластера

Создание сервисов

В Kubernetes Service — это способ экспонирования сетевого приложения, которое работает в виде одного или нескольких Pod в вашем кластере.

Содержание

Зачем нужен Service

1. У Pod есть собственные IP-адреса, но:

- IP Pod нестабильны (меняются при пересоздании Pod).

- Прямой доступ к Pod становится ненадежным.

2. Service решает эту проблему, предоставляя:

- Стабильный IP и DNS-имя.
- Автоматическое балансирование нагрузки на соответствующие Pod.

Пример Service типа ClusterIP:

```yaml
# simple-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  type: ClusterIP (1)
  selector: (2)
    app.kubernetes.io/name: MyApp
  ports:
    - protocol: TCP
      port: 80 (3)
      targetPort: 80 (4)
```

(1) Доступные значения type и их поведение: `ClusterIP`, `NodePort`, `LoadBalancer`, `ExternalName`

(2) Набор Pod, на которые нацелен Service, обычно определяется селектором, который вы задаёте.

(3) Порт Service.

(4) Привязка `targetPort` Service к `containerPort` Pod. Также можно ссылаться на `port.name` в контейнере Pod.

Headless Services

Иногда не требуется балансировка нагрузки и единый IP Service. В таком случае можно создать так называемые headless Services:

```
spec:
  clusterIP: None
```

Headless Services полезны, когда:

- Нужно обнаруживать отдельные IP Pod, а не только один IP сервиса.
- Требуются прямые подключения к каждому Pod (например, для баз данных типа Cassandra или StatefulSets).
- Используются StatefulSets, где каждый Pod должен иметь стабильное DNS-имя.

Создание сервиса через веб-консоль

1. Перейдите в **Container Platform**.
2. В левой навигационной панели выберите **Network** > **Services**.
3. Нажмите **Create Service**.
4. Следуйте инструкциям для настройки соответствующих параметров.

Параметр	Описание
Virtual IP Address	Если включено, для этого Service будет выделен ClusterIP, который можно использовать для обнаружения сервиса внутри кластера. Если отключено, будет создан headless Service, который обычно используется для **StatefulSet**.
Type	• **ClusterIP**: Экспонирует Service на внутреннем IP кластера. При выборе этого значения Service доступен только внутри кластера.

Параметр	Описание
	• **NodePort**: Экспонирует Service на IP каждого Node на статическом порту (NodePort). • **ExternalName**: Отображает Service на содержимое поля externalName (например, на hostname api.foo.bar.example). • **LoadBalancer**: Экспонирует Service внешне с помощью внешнего балансировщика нагрузки. Kubernetes не предоставляет собственный компонент балансировки нагрузки; вы должны обеспечить его самостоятельно или интегрировать кластер с облачным провайдером.
Target Component	• **Workload**: Service будет перенаправлять запросы на **конкретный** workload, который соответствует меткам, например, `project.cpaas.io/name: projectname` и `service.cpaas.io/name: deployment-name`. • **Virtualization**: Service будет перенаправлять запросы на **конкретную** виртуальную машину или группу виртуальных машин. • **Label Selector**: Service будет перенаправлять запросы на **определённый тип** workload с указанными метками, например, `environment: release`.
Port	Используется для настройки сопоставления портов для этого Service. В следующем примере другие Pod внутри кластера могут обращаться к этому Service через виртуальный IP (если включён) и TCP порт *80*; запросы будут перенаправлены на внешний TCP порт *6379* или *redis* Pod целевого компонента. • **Protocol**: Протокол, используемый Service, поддерживаются: `TCP`, `UDP`, `HTTP`, `HTTP2`, `HTTPS`, `gRPC`. • **Service Port**: Номер порта, который Service экспонирует внутри кластера, то есть Port, например, *80*.

Параметр	Описание
	• **Container Port**: Целевой порт (или имя), на который маппится service port, то есть targetPort, например, *6379* или *redis*. • **Service Port Name**: Генерируется автоматически. Формат: `<protocol>-<service port>-<container port>`, например: *tcp-80-6379* или *tcp-80-redis*.
Session Affinity	Сессионная аффинити на основе IP-адреса источника (ClientIP). Если включено, все запросы с одного IP-адреса будут направляться на один и тот же сервер при балансировке нагрузки, что гарантирует обработку запросов от одного клиента одним сервером.

5. Нажмите **Create**.

Создание сервиса через CLI

```
kubectl apply -f simple-service.yaml
```

Создать сервис на основе существующего ресурса deployment `my-app`.

```
kubectl expose deployment my-app \
  --port=80 \
  --target-port=8080 \
  --name=test-service \
  --type=NodePort \
  -n p1-1
```

Пример: Доступ к приложению внутри кластера

```yaml
# access-internal-demo.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.25
          ports:
            - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: nginx-clusterip
spec:
  type: ClusterIP
  selector:
    app: nginx
  ports:
    - port: 80
      targetPort: 80
```

1. Примените этот YAML:

```
kubectl apply -f access-internal-demo.yaml
```

2. Запустите другой Pod:

```
kubectl run test-pod --rm -it --image=busybox -- /bin/sh
```

3. Доступ к сервису `nginx-clusterip` из Pod `test-pod`:

```
wget -qO- http://nginx-clusterip
# или используя DNS-записи, созданные автоматически Kubernetes: <service-name>.
<namespace>.svc.cluster.local
wget -qO- http://nginx-clusterip.default.svc.cluster.local
```

Вы должны увидеть HTML-ответ с текстом типа "Welcome to nginx!".

Пример: Доступ к приложению вне кластера

```yaml
# access-external-demo.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.25
          ports:
            - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: nginx-nodeport
spec:
  type: NodePort
  selector:
    app: nginx
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30080
```

1. Примените этот YAML:

```
kubectl apply -f access-external-demo.yaml
```

2. Проверка Pod:

```
kubectl get pods -l app=nginx -o wide
```

3. curl к Service:

```
curl http://{NodeIP}:{nodePort}
```

Вы должны увидеть HTML-ответ с текстом типа "Welcome to nginx!".

Разумеется, можно также получить доступ к приложению из вне кластера, создав Service типа LoadBalancer.

Примечание: Пожалуйста, предварительно настройте сервис LoadBalancer.

```yaml
# access-external-demo-with-loadbalancer.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.25
          ports:
            - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: nginx-lb-service
spec:
  type: LoadBalancer
  selector:
    app: nginx
  ports:
    - port: 80
      targetPort: 80
```

1. Примените этот YAML:

```
kubectl apply -f access-external-demo-with-loadbalancer.yaml
```

2. Получите внешний IP-адрес:

```
kubectl get svc nginx-lb-service
```

```
NAME            TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)        AGE
nginx-service   LoadBalancer   10.0.2.57       34.122.45.100   80:30005/TCP   30s
```

`EXTERNAL-IP` — это адрес, по которому вы можете получить доступ из браузера.

```
curl http://34.122.45.100
```

Вы должны увидеть HTML-ответ с текстом типа "Welcome to nginx!".

Если EXTERNAL-IP равен `pending`, значит сервис LoadBalancer в данный момент не развернут в кластере.

Пример: Service типа ExternalName

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-external-service
  namespace: default
spec:
  type: ExternalName
  externalName: example.com
```

1. Примените этот YAML:

```
kubectl apply -f external-service.yaml
```

2. Попробуйте разрешить имя внутри Pod в кластере:

```
kubectl run test-pod --rm -it --image=busybox -- sh
```

затем:

```
nslookup my-external-service.default.svc.cluster.local
```

Вы увидите, что имя разрешается в `example.com`.

Аннотации для Service типа LoadBalancer

AWS EKS Cluster

Для подробного описания аннотаций LoadBalancer Service в EKS, пожалуйста, обратитесь к Annotation Usage Documentation ↗ .

Ключ	Значение	Описание
service.beta.kubernetes.io/aws-load-balancer-type	external: Использовать официальный AWS LoadBalancer Controller.	Определяет контроллер для типа LoadBalancer. **Примечание**: Пожалуйста, заранее свяжитесь с администратором платформы для развертывания AWS LoadBalancer Controller.
service.beta.kubernetes.io/aws-load-balancer-nlb-target-type	• instance: Трафик будет направляться на	Определяет, как трафик достигает Pod.

Ключ	Значение	Описание
	Pod через NodePort. • ip: Трафик направляется напрямую на Pod (кластер должен использовать Amazon VPC CNI).	
service.beta.kubernetes.io/aws-load-balancer-scheme	• internal: Частная сеть. • internet-facing: Публичная сеть.	Определяет, использовать ли частную или публичную сеть.
service.beta.kubernetes.io/aws-load-balancer-ip-address-type	• IPv4 • dualstack	Определяет поддерживаемый стек IP-адресов.

Huawei Cloud CCE Cluster

Для подробного описания аннотаций LoadBalancer Service в CCE, пожалуйста, обратитесь к Annotation Usage Documentation↗ .

Ключ	
kubernetes.io/elb.id	

Ключ	

kubernetes.io/elb.autocreate	Пример: `{"type":"public","bandwidth_name":"cce-bandwidt`
	`1551163379627","bandwidth_chargemode":"bandwidth","bandwic`
	`["cn-north-4b"],"l4_flavor_name":"L4_flavor.elb.s1.small"}`
	
	Примечание: Пожалуйста, сначала ознакомьтесь с ин
	параметры примера.

kubernetes.io/elb.subnet-id	
kubernetes.io/elb.class	• union: Общая балансировка нагрузки. • performance: Эксклюзивная балансировка нагрузки

Ключ	
kubernetes.io/elb.enterpriseID	

Azure AKS Cluster

Для подробного описания аннотаций LoadBalancer Service в AKS, пожалуйста, обратитесь к Annotation Usage Documentation↗ .

Ключ	Значение	Описание
service.beta.kubernetes.io/azure-load-balancer-internal	• true: Частная сеть. • false: Публичная сеть.	Определяет, использовать ли частную или публичную сеть.

Google GKE Cluster

Для подробного описания аннотаций LoadBalancer Service в GKE, пожалуйста, обратитесь к Annotation Usage Documentation↗ .

Ключ	Значение	Описание
networking.gke.io/load-balancer-type	Internal	Определяет использование частной сети.

Ключ	Значение	Описание
loud.google.com/l4-rbs	enabled	По умолчанию публичный. Если этот параметр настроен, трафик будет направляться напрямую на Pod.

Создание Ingress

Правила Ingress (Kubernetes Ingress) открывают HTTP/HTTPS маршруты снаружи кластера для внутренней маршрутизации (Kubernetes Service), что позволяет контролировать внешний доступ к вычислительным компонентам.

Создайте Ingress для управления внешним HTTP/HTTPS доступом к Service.

WARNING

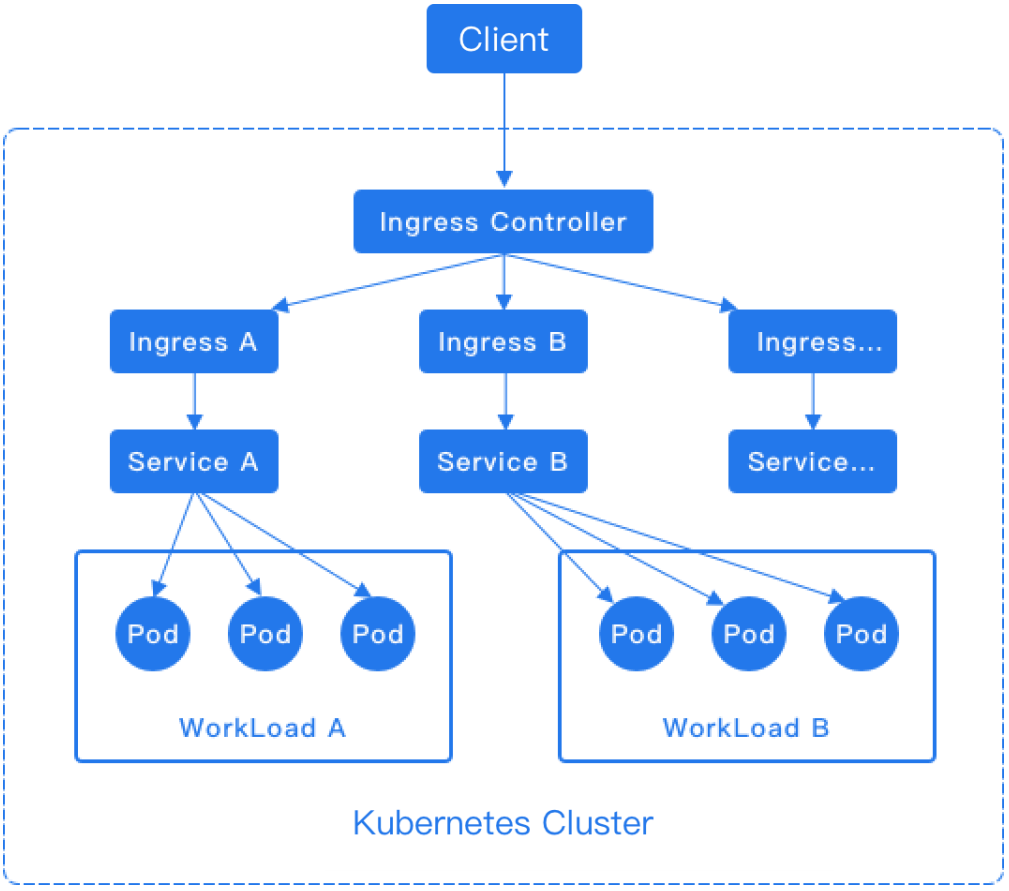
При создании нескольких ingress в одном и том же namespace разные ingress **НЕ ДОЛЖНЫ** иметь одинаковые **Domain**, **Protocol** и **Path** (то есть дублирование точек доступа не допускается).

Содержание

Метод реализации

Правила Ingress зависят от реализации Ingress Controller, который отвечает за отслеживание изменений в Ingress и Service. После создания нового правила Ingress

внутри Ingress Controller автоматически создаётся правило переадресации, соответствующее правилу Ingress. Когда Ingress Controller получает запрос, он сопоставляет правило переадресации с правилом Ingress и распределяет трафик по указанным внутренним маршрутам, как показано на схеме ниже.



NOTE

Для протокола HTTP Ingress поддерживает только порт 80 в качестве внешнего порта. Для протокола HTTPS Ingress поддерживает только порт 443 в качестве внешнего порта. Балансировщик нагрузки платформы автоматически добавит порты 80 и 443 для прослушивания.

Быстрый старт

Далее мы используем community-версию Ingress-NGINX, чтобы продемонстрировать, как получить доступ к вашему приложению с помощью контроллера NGINX.

1. Разверните контроллер `Ingress-NGINX`.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.12.2/deploy/static/provider/cloud/deploy.yaml
```

После выполнения этой команды автоматически создаются следующие ресурсы:

Вид	Имя	Описание
`Namespace`	`ingress-nginx`	Ресурсы для изоляции контроллеров
`ServiceAccount`	`ingress-nginx`	Сервисный аккаунт для контроллера
`ClusterRole`	`ingress-nginx`	Разрешения на уровне кластера
`ClusterRoleBinding`	`ingress-nginx`	Привязка ClusterRole к SA
`ConfigMap`	`ingress-nginx-controller`	Настройка поведения контроллера (например, уровни логирования, таймаут прокси и т.д.)
`ValidatingWebhookConfig`	`ingress-nginx-admission`	Вебхук для проверки легитимности конфигурации Ingress (опционально)
`Service` (TCP/UDP)	`ingress-nginx-controller`	Тип по умолчанию `LoadBalancer`, можно изменить на `NodePort`.
`Deployment`	`ingress-nginx-controller`	
`Pod`	`ingress-nginx-controller-xxx`	
`Role` / `RoleBinding`	связанные с admission	Поддержка вебхука
`Job`	`ingress-nginx-admission-create`	Регистрация вебхука

Если вы хотите изменить адрес реестра по умолчанию, можно скачать YAML-файл с помощью `curl`, изменить его и затем применить.

```
curl -O https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.12.2/deploy/static/provider/cloud/deploy.yaml
```

Ожидайте запуска Pod `ingress-nginx-controller-xxx`.

2. Локальное тестирование

- Создайте простой веб-сервер и связанный с ним сервис:

```
kubectl create deployment demo --image=nginx --port=80
kubectl expose deployment demo
```

- Создайте ресурс ingress. В этом примере используется хост, который сопоставляется с `localhost`:

```
kubectl create ingress demo-localhost --class=nginx \
  --rule="demo.local/*=demo:80"
```

- Пробросьте локальный порт к контроллеру ingress:

```
kubectl port-forward --namespace=ingress-nginx service/ingress-nginx-controller 8080:80
```

- Получите доступ к вашему развертыванию с помощью curl:

```
curl --resolve demo.local:8080:127.0.0.1 http://demo.local:8080
```

Примечание: Этот параметр временно разрешает доменное имя demo.local в IP 127.0.0.1 и используется на порту 8080. При посещении http://demo.local:8080 ↗ вы фактически посещаете http://127.0.0.1:8080 ↗. С другой стороны, следует настроить `hosts`:

```
echo "127.0.0.1 demo.local" | sudo tee -a /etc/hosts
```

В итоге вы должны увидеть HTML-ответ с текстом вроде "Welcome to nginx!".

После этого вы можете получить доступ к сайту по адресу `http://demo.local:8080/`.

INFO

Тип по умолчанию для `ingress-nginx-controller` — `LoadBalancer`. Если поле `EXTERNAL-IP` показывает `pending`, это означает, что вашему Kubernetes кластеру не удалось создать балансировщик нагрузки.

Если вы интегрируетесь с провайдером, который поддерживает указание IP-адресов балансировщика нагрузки для Service через (специфичные для провайдера) аннотации, рекомендуется использовать этот способ.

3. Онлайн-тестирование

Когда у вашего `ingress-nginx-controller` (Service типа LoadBalancer) появляется `EXTERNAL-IP`, вы можете создать ресурс ingress. В следующем примере предполагается, что вы настроили DNS-запись для `www.developer.io`:

```
kubectl create ingress demo --class=nginx \
  --rule="www.developer.io/*=demo:80"
```

Вы можете получить доступ к `http://www.developer.io` и увидеть тот же вывод.

Предварительные требования

- В текущем namespace должен быть доступен **Service**.
- Убедитесь у администратора, что для проекта, связанного с текущим namespace, выделено используемое доменное имя.

- Для доступа к домену через HTTPS необходимо предварительно сохранить HTTPS-сертификат в виде TLS-секрета.

Пример Ingress:

```yaml
# nginx-ingress.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx-ingress
  namespace: k-1
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: / (1)
spec:
  ingressClassName: nginx (2)
  rules:
    - host: demo.local (3)
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: nginx-service
                port:
                  number: 80
```

(1) Для получения дополнительных настроек смотрите nginx-configuration↗.

(2) Используется контроллер `ingress-nginx`.

(3) Если вы хотите запускать ingress только локально, предварительно настройте `hosts`.

Создание Ingress через веб-консоль

1. Зайдите в **Container Platform**.

2. В левой навигационной панели выберите **Network** > **Ingress**.

3. Нажмите **Create Ingress**.

4. Используйте инструкции ниже для настройки параметров.

Параметр	Описание
Ingress Class	Ingress может реализовываться разными контроллерами с разными именами `IngressClass`. Если на платформе доступно несколько ingress контроллеров, пользователь может выбрать нужный с помощью этого параметра.
Domain Name	Хосты могут быть точными совпадениями (например, `foo.bar.com`) или шаблонами с подстановочными знаками (например, `*.foo.com`). Доступные доменные имена выделяются администратором платформы.
Certificates	TLS-секрет или сертификаты, выделенные администратором платформы.
Match Type и **Path**	• **Prefix**: Совпадение по префиксу пути, например, `/abcd` совпадает с `/abcd/efg` или `/abcde`. • **Exact**: Совпадение по точному пути, например, `/abcd`. • **Implementation specific**: Если вы используете кастомный Ingress контроллер для управления правилами Ingress, можно позволить контроллеру решать самостоятельно.
Service	Внешний трафик будет перенаправлен на этот Service.
Service Port	Укажите порт Service, на который будет перенаправлен трафик.

5. Нажмите **Create**.

Создание Ingress через CLI

```
kubectl apply -f nginx-ingress.yaml
```

NOTE

Если у ingress отсутствует Ingress Class, все ALB-инстансы, выделенные этому проекту, будут обрабатывать этот ingress.

Настройка Gateway

Входящий шлюз (Gateway) — это экземпляр, развернутый из Gateway Class. Он создает слушатели для перехвата внешнего трафика на указанных доменных именах и портах. Вместе с правилами маршрутизации он может направлять указанный внешний трафик на соответствующие backend-экземпляры.

Создайте входящий шлюз для более точного распределения сетевых ресурсов.

Содержание

Терминология

Название ресурса	Обзор	Инструкция по использованию
Gateway Class	В стандартной документации Gateway API Gateway Class определяется как шаблон для создания шлюзов. Разные шаблоны могут создавать входящие шлюзы для различных бизнес-сценариев, что облегчает быстрое управление трафиком.	Платформа включает выделенные Gateway Classes.
Inbound Gateway	Входящий шлюз соответствует конкретным экземплярам ресурсов, и пользователь может эксклюзивно использовать все слушающие и вычислительные ресурсы этого входящего шлюза. Это конфигурация правил маршрутизации, действующая для слушателя. При обнаружении внешнего трафика шлюз распределяет его на backend-экземпляры согласно правилам маршрутизации.	Его можно рассматривать как экземпляр балансировщика нагрузки.
Route Rule	Правила маршрутизации определяют набор правил для распределения трафика от шлюза к сервисам. В настоящее время стандартно поддерживаемые типы правил маршрутизации в Gateway API включают HTTPRoute, TCPRoute, UDPRoute и др.	Платформа поддерживает прослушивание протоколов HTTP, HTTPS, TCP и UDP.

Предварительные требования

Администратор платформы должен убедиться, что кластер поддерживает внутреннюю маршрутизацию типа LoadBalancer. Для публичных облачных кластеров должен быть

установлен LoadBalancer Service Controller. В непубличных облачных кластерах платформа предоставляет функцию пула внешних адресов, которая позволяет внутренней маршрутизации типа LoadBalancer автоматически получать IP из пула внешних адресов для внешнего доступа после завершения настройки.

Пример Gateway и пользовательского ресурса Alb2 (CR)

```yaml
# demo-gateway.yaml
apiVersion: gateway.networking.k8s.io/v1beta1
kind: Gateway
metadata:
  namespace: k-1
  name: test
  annotations:
    cpaas.io/display-name: ces
    listeners.cpaas.io/creationTimestamp: '["2025-05-26T02:05:56.135Z"]'
    listeners.cpaas.io/display-name: '[""]'
  labels:
    alb.cpaas.io/alb-ref: test-o93q7
spec:
  gatewayClassName: exclusive-gateway (1)
  listeners:
    - allowedRoutes:
        namespaces:
          from: All
      name: gateway-metric
      protocol: TCP
      port: 11782
---
apiVersion: crd.alauda.io/v2beta1
kind: ALB2
metadata:
  namespace: k-1
  name: test-o93q7 (2)
spec:
  type: nginx
  config:
    enableAlb: false
    networkMode: container
    resources:
      limits:
        cpu: 200m
        memory: 256Mi
      requests:
        cpu: 200m
        memory: 256Mi
    vip:
      enableLbSvc: true
      lbSvcAnnotations: {}
    gateway:
```

```
    mode: standalone
    name: test # # [!code callout] 3
```

1. См. описание Gateway Class ниже.
2. Имя `alb2` формируется как `{gatewayName}-{random}`.
3. Имя `gateway`.

Создание Gateway через веб-консоль

1. Перейдите в **Container Platform**.
2. В левой навигационной панели выберите **Network** > **Inbound Gateway**.
3. Нажмите **Create Inbound Gateway**.
4. Следуйте инструкциям для настройки конкретных параметров.

Параметр	Описание
Name	Имя входящего шлюза.
Gateway Class	Gateway Class определяет поведение шлюза, аналогично концепции классов хранения (StorageClasses); это ресурс кластера. **Dedicated**: входящий шлюз будет соответствовать конкретному экземпляру ресурса, и пользователь сможет использовать все слушатели и вычислительные ресурсы этого шлюза.
Specification	Можно выбрать рекомендованный сценарий использования в зависимости от потребностей или настроить лимиты ресурсов самостоятельно.
Access Address	Адрес входящего шлюза, который по умолчанию получается автоматически.
Internal Routing Annotation	Используется для объявления конфигурации или возможностей внутренней маршрутизации типа

Параметр	Описание
	LoadBalancer. Для подробной информации об аннотациях смотрите LoadBalancer type internal routing annotation instructions.

5. Нажмите **Create**.

Создание Gateway через CLI

```
kubectl apply -f demo-gateway.yaml
```

Просмотр ресурсов, созданных платформой

После создания входящего шлюза платформа автоматически создает множество ресурсов. Не удаляйте перечисленные ниже ресурсы.

Ресурсы, создаваемые по умолчанию	Имя
Ресурс типа ALB2	*name-lb-random*
Deployment	*name-lb-random*
Внутренняя маршрутизация	• *name-lb-random* • *name-lb-random-lb-random*
Конфигурационный словарь	• *name-lb-random-port-info* • *name-lb-random*
Service Account	*name-lb-random-serviceaccount*

Обновление Gateway

NOTE

Обновление входящего шлюза приведет к прерыванию сервиса на 3-5 минут. Пожалуйста, выбирайте подходящее время для выполнения этой операции.

Обновление Gateway через веб-консоль

1. Зайдите в **Container Platform**.
2. В левой навигационной панели выберите **Network** > **Inbound Gateway**.
3. Нажмите ⋮ > **Update**.
4. Обновите конфигурацию входящего шлюза по необходимости.

   Примечание: Пожалуйста, разумно задавайте спецификации в соответствии с бизнес-требованиями.
5. Нажмите **Update**.

Добавление слушателя

Отслеживает трафик по указанным доменным именам и перенаправляет его на backend-экземпляры согласно привязанным правилам маршрутизации.

Предварительные требования

- Если необходимо отслеживать протокол HTTP, заранее свяжитесь с администратором для подготовки **доменного имени**.

- Если необходимо отслеживать протокол HTTPS, заранее свяжитесь с администратором для подготовки **доменного имени** и **сертификата**.

Добавление слушателя через веб-консоль

1. В левой навигационной панели выберите **Network** > **Inbound Gateway**.
2. Нажмите на ***Имя входящего шлюза***.
3. Нажмите **Add Listener**.
4. Следуйте инструкциям для настройки конкретных параметров.

Параметр	Описание
Протокол и порт слушателя	В настоящее время поддерживается мониторинг протоколов HTTP, HTTPS, TCP и UDP, можно ввести порт для мониторинга, например: `80`. **Примечание**: • При совпадении портов типы слушателей HTTP, HTTPS и TCP не могут сосуществовать; можно выбрать только один из протоколов. • При использовании HTTP или HTTPS и совпадении портов доменные имена должны быть разными.
Доменное имя	Выберите доступное доменное имя в текущем namespace для мониторинга сетевого трафика, обращающегося к этому домену. **Подсказка**: протоколы TCP и UDP не поддерживают выбор доменных имен.

5. Нажмите **Create**.

Добавление слушателя через CLI

```
kubectl patch gateway test \
  -n k-1 \
  --type=merge \
  -p '{
    "metadata": {
      "annotations": {
        "listeners.cpaas.io/creationTimestamp": "[\"2025-05-26T02:05:56.135Z\",\"2025-05-
26T03:33:52.431Z\"]",
        "listeners.cpaas.io/display-name": "[\"\",\"\" ]"
      }
    },
    "spec": {
      "listeners": [
        {
          "allowedRoutes": {
            "namespaces": {
              "from": "All"
            }
          },
          "name": "gateway-metric",
          "protocol": "TCP",
          "port": 11782
        },
        {
          "allowedRoutes": {
            "namespaces": {
              "from": "All"
            }
          },
          "name": "demo-listener",
          "protocol": "HTTP",
          "port": 8088,
          "hostname": "developer.test.cn"
        }
      ]
    }
  }'
```

Создание правил маршрутизации

Правила маршрутизации задают политики маршрутизации для входящего трафика, аналогично входящим правилам (Kubernetes Ingress). Они обеспечивают доступ к сетевому трафику, отслеживаемому шлюзом, для внутренней маршрутизации кластера (Kubernetes Service), облегчая стратегию маршрутизации и переадресации. Главное отличие в том, что они нацелены на разные объекты сервиса: входящие правила обслуживают Ingress Controller, а правила маршрутизации — Ingress Gateway.

После настройки прослушивания в ingress gateway шлюз будет в реальном времени отслеживать трафик с указанных доменов и портов. Правила маршрутизации могут направлять входящий трафик на backend-экземпляры по желанию.

Пример пользовательского ресурса HTTPRoute (CR)

```yaml
# example-httproute.yaml
apiVersion: gateway.networking.k8s.io/v1beta1
kind: HTTPRoute (1)
metadata:
  namespace: k-1
  name: example-http-route
  annotations:
    cpaas.io/display-name: ""
spec:
  hostnames:
    - developer.test.cn
  parentRefs:
    - kind: Gateway
      namespace: k-1
      name: test
      sectionName: demo-listener (2)
  rules:
    - matches:
        - path:
            type: Exact
            value: "/demo"
      filters: []
      backendRefs:
        - kind: Service
          name: test-service
          namespace: k-1
          port: 80
          weight: 100
```

1. Доступные типы: `HTTPRoute`, `TCPRoute`, `UDPRoute`.
2. Имя слушателя `Gateway`.

NOTE

Если для объекта **Path** в правиле маршрутизации типа HTTPRoute нет совпадающего правила, автоматически добавляется правило с режимом PathPrefix и значением /.

Создание маршрута через веб-консоль

1. Зайдите в **Container Platform**.

2. В левой навигационной панели выберите **Network** > **Route Rules**.

3. Нажмите **Create Route Rule**.

4. Следуйте инструкциям для настройки параметров.

Параметр	Описание
Тип маршрута	В настоящее время поддерживаются типы маршрутов: HTTPRoute, TCPRoute, UDPRoute. **Подсказка:** HTTPRoute поддерживает публикацию на слушатели протоколов HTTP и HTTPS.
Публикация на слушатель	В левой панели выберите созданный **Ingress Gateway**, в правой — созданный **Listener**. Платформа опубликует созданные правила маршрутизации на выбранный слушатель, позволяя шлюзу перенаправлять перехваченный трафик на указанные backend-экземпляры. **Примечание:** Нельзя публиковать правила маршрутизации на слушатель, который работает на порту **11782** или уже содержит TCP или UDP маршруты.
Совпадение (Match)	Можно добавить одно или несколько правил совпадения для перехвата трафика, соответствующего требованиям. Например, **перехват трафика с указанным Path**, **перехват трафика с указанным методом** и т.д. **Примечание:** • Нажмите **Add**; при добавлении нескольких правил маршрутизации связь между ними — 'AND', все правила должны совпадать для активации.

Параметр	Описание
	• Нажмите **Add Match**; при добавлении нескольких групп правил связь между группами — 'OR', достаточно совпадения любой группы для активации. • TCPRoute и UDPRoute не поддерживают настройку правил совпадения. • Если объект совпадения — **path**, а метод совпадения — **Exact** или **PathPrefix**, значение **value** должно начинаться с "/" и не должно содержать символы типа "//", "/./", "/../", "%2f", "%2F", "#", "/..", "/." и т.п.
Действие (Action)	Можно добавить одно или несколько действий для обработки перехваченного трафика. • Header: заголовок HTTP-сообщения содержит много метаданных, предоставляющих дополнительную информацию о запросе или ответе. Изменяя поля заголовка, сервер может влиять на обработку запроса и ответа. • Redirect: совпадающий URL будет обработан указанным образом, затем запрос будет инициирован заново. • Rewrite: совпадающий URL будет обработан указанным образом, затем запрос будет перенаправлен на другой путь ресурса или имя файла. **Примечание:** • Нажмите **Add**; при добавлении нескольких правил действий платформа выполнит все действия последовательно в порядке отображения. • TCPRoute и UDPRoute не поддерживают настройку правил действий. • В одном правиле маршрутизации не может быть несколько действий типа **Header** с одинаковым значением **value**.

Параметр	Описание
	• В одном правиле маршрутизации может быть только одно действие типа **Redirect** или **Rewrite**, и только один режим из **FullPath** или **PrefixPath**. • Если хотите использовать операцию **PrefixPath**, сначала добавьте правило совпадения с режимом **PathPrefix**.
Backend Instance	После активации правила трафик будет перенаправлен на backend-экземпляры согласно выбранным внутренним маршрутам и портам в текущем namespace. Можно также настроить веса, при этом экземпляры с большим весом будут выбираться с большей вероятностью. **Подсказка:** Процент рядом с весом показывает вероятность перенаправления на этот экземпляр, рассчитываемую как отношение текущего значения веса к сумме всех весов.

5. Нажмите **Create**.

Создание маршрута через CLI

```
kubectl apply -f example-httproute.yaml
```

Создание доменного имени

Добавьте ресурсы доменных имен на платформу и выделите домены для использования всеми проектами в кластере или ресурсами в конкретном проекте. При создании доменного имени поддерживается привязка сертификата.

NOTE

Созданные на платформе доменные имена должны быть разрешены на адрес балансировщика нагрузки кластера, прежде чем к ним можно будет получить доступ по доменному имени. Поэтому необходимо убедиться, что добавленные на платформу доменные имена успешно зарегистрированы и что доменные имена разрешаются на адрес балансировщика нагрузки кластера.

Успешно созданные и выделенные доменные имена на платформе могут использоваться в следующих функциях **Container Platform**:

- **Создание входящих правил**: **Network Management** > **Inbound Rules** > **Create Inbound Rule**
- **Создание нативных приложений**: **Application Management** > **Native Applications** > **Create Native Application** > **Add Inbound Rule**
- **Добавление прослушиваемых портов** для балансировки нагрузки: **Network Management** > **Load Balancer Details** > **Add Listening Port**

После привязки доменного имени к сертификату разработчики приложений могут просто выбрать доменное имя при настройке балансировщика нагрузки и входящих правил, что позволит использовать сертификат, связанный с доменным именем, для поддержки https.

Содержание

Пример ресурса Domain custom resource (CR)

```yaml
# test-domain.yaml
apiVersion: crd.alauda.io/v2
kind: Domain
metadata:
  name: "000000000030755752601296 86e67ed4-917a-454a-8553-d55fc4030f81"
  annotations:
    cpaas.io/secret-ref: developer.test.cn-xfd8x (1)
  labels:
    cluster.cpaas.io/name: global
    project.cpaas.io/name: cong
spec:
  name: developer.test.cn
  kind: full
```

(1) Если включены сертификаты, необходимо заранее создать Secret типа LTS. `secret-ref` — это имя секрета.

Создание домена через веб-консоль

1. Перейдите в **Platform Management**.
2. В левой навигационной панели выберите **Network Management** > **Domain Names**.
3. Нажмите **Create Domain Name**.
4. Настройте соответствующие параметры согласно следующим инструкциям.

Параметр	Описание
Type	• Domain: Полное доменное имя, например, `developer.test.cn`. • Wildcard Domain: Подстановочный домен с символом подстановки (*), например, `*.test.cn`, который включает все поддомены домена `test.cn`.
Domain	Введите полное доменное имя или суффикс домена в зависимости от выбранного типа доменного имени.
Allocate Cluster	Если выделяется кластер, необходимо также выбрать проект, связанный с выделенным кластером, например, все проекты, связанные с кластером.
Certificate	Включает публичный ключ (tls.crt) и приватный ключ (tls.key) для создания сертификата, привязанного к доменному имени. Проект, которому выделен сертификат, совпадает с проектом привязанного доменного имени. **Примечания**: • Импорт бинарных файлов не поддерживается. • Привязанный сертификат должен соответствовать условиям корректного формата, быть в пределах срока действия и подписан для доменного имени и т.д. • После создания привязанного сертификата формат имени сертификата: доменное имя - случайные символы. • После создания привязанного сертификата он отображается в списке сертификатов, но обновление и удаление привязанного сертификата поддерживается только на странице деталей домена. • После создания привязанного сертификата поддерживается обновление содержимого сертификата, но замена на другой сертификат не поддерживается.

5. Нажмите **Create**.

Создание домена с помощью CLI

```
kubectl apply -f test-domain.yaml
```

Последующие действия

- **Регистрация домена**: Зарегистрируйте домен, если созданный домен еще не зарегистрирован.
- **Разрешение домена**: Выполните разрешение домена, если домен не указывает на адрес балансировщика нагрузки кластера платформы.

Дополнительные ресурсы

- Configure Certificate

Создание сертификатов

После того как администратор платформы импортирует TLS-сертификат и назначит его определённому проекту, разработчики с соответствующими правами на проект смогут использовать сертификат, импортированный и назначенный администратором платформы, при работе с входящими правилами и функционалом балансировки нагрузки. В дальнейшем, например, при истечении срока действия сертификата, администратор платформы сможет централизованно обновить сертификат.

NOTE

Функциональность сертификатов в настоящее время не поддерживается для использования в кластерах публичного облака. При необходимости вы можете создавать секреты типа TLS в указанном namespace.

Содержание

Создание сертификата через веб-консоль

1. Перейдите в **Platform Management**.
2. В левой навигационной панели выберите **Network Management** > **Certificates**.
3. Нажмите **Create Certificate**.
4. Следуйте инструкциям ниже для настройки соответствующих параметров.

Параметр	Описание
Assign Project	• All Projects: Назначить сертификат для использования во всех проектах, связанных с текущим кластером. • Specified Project: Назначить сертификат для использования в указанном проекте. • No Assignment: Пока не назначать проект. После создания сертификата вы сможете обновить проекты, которые могут использовать сертификат, через операцию **Update Project**.
Public Key	Это tls.crt. При импорте публичного ключа бинарные файлы не поддерживаются.
Private Key	Это tls.key. При импорте приватного ключа бинарные файлы не поддерживаются.

5. Нажмите **Create**.

Создание пула внешних IP-адресов

Пул внешних IP-адресов — это набор IP-адресов, который MetalLB использует для получения внешних IP-адресов доступа для внутренних маршрутов типа LoadBalancer.

Содержание

Предварительные требования

Если необходимо использовать пул внешних IP-адресов типа BGP, обратитесь к администратору для включения соответствующих функций.

Ограничения и условия

IP-ресурсы для внешнего адреса должны соответствовать следующим условиям:

- Пул внешних адресов должен быть связан на уровне канального уровня (L2) с доступными узлами.
- IP-адреса должны быть пригодны для использования платформой и не могут включать IP-адреса, уже используемые физической сетью, например, IP-адреса шлюзов.
- Не должно быть пересечений с сетями, используемыми кластером, включая Cluster CIDR, Service CIDR, подсети и т.д.
- В среде с двойным стеком необходимо, чтобы в одном пуле внешних адресов одновременно присутствовали как IPv4, так и IPv6 адреса, и их количество было больше 0. В противном случае внутренние маршруты типа LoadBalancer с двойным стеком не смогут получить внешние адреса доступа.
- В среде IPv6 DNS узлов должен поддерживать IPv6, иначе плагин MetalLB не сможет быть успешно развернут.

Развертывание плагина MetalLB

Использование пула внешних адресов зависит от плагина MetalLB.

1. Перейдите в **Platform Management**.
2. В левой навигационной панели выберите **Marketplace** > **Cluster Plugin**.
3. Найдите MetalLB, нажмите на **MetalLB** справа от ⋮ > **Deploy**.
4. Дождитесь, пока статус развертывания не изменится на **Deployment Successful**, чтобы завершить процесс.

Пример ресурса custom resource (CR) IPAddressPool

```yaml
# ippool-with-L2advertisement.yaml
kind: IPAddressPool
apiVersion: metallb.io/v1beta1
metadata:
  name: test-ippool
  namespace: metallb-system
spec:
  addresses:
    - 13.1.1.1/24
  avoidBuggyIPs: true
---
kind: L2Advertisement
apiVersion: metallb.io/v1beta1
metadata:
  name: test-ippool
  namespace: metallb-system
spec:
  ipAddressPools:
    - test-ippool (1)
  nodeSelectors:
    - matchLabels: {}
      matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
            - 192.168.66.210
```

Режим BGP:

```yaml
# ippool-with-bgpadvertisement.yaml
kind: IPAddressPool
apiVersion: metallb.io/v1beta1
metadata:
  name: test-pool-bgp
  namespace: metallb-system
spec:
  addresses:
    - 4.4.4.3/23
  avoidBuggyIPs: true
---
kind: BGPAdvertisement
apiVersion: metallb.io/v1beta1
metadata:
  name: test-pool-bgp
  namespace: metallb-system
spec:
  ipAddressPools:
    - test-pool-bgp
  nodeSelectors:
    - matchLabels:
        alertmanager: "true"
  peers:
    - test-bgp-example
```

1 Ссылка на пул IP-адресов.

INFO

Q: Что такое `L2Advertisement`?

A:

1. `L2Advertisement` — это Custom Resource (CRD), предоставляемый MetalLB для управления тем, какие IP-адреса из пула IPAddressPool должны транслироваться через ARP (IPv4) или NDP (IPv6) в режиме канального уровня (Layer 2).

Q: Какова цель `L2Advertisement`?

A:

1. Указать, какие IP-адреса из IPAddressPool транслировать на уровне L2 (ARP/NDP объявления);
2. Контролировать поведение трансляции, чтобы предотвратить конфликты IP или трансляцию между сегментами;
3. Ограничивать область трансляции в средах с несколькими сетевыми интерфейсами и сетями.

Проще говоря, это говорит MetalLB: какие IP-адреса можно транслировать и кому (например, каким узлам).

Без определения `L2Advertisement` в режиме Layer2 MetalLB не будет транслировать никакие адреса.

Q: Что такое `BGPAdvertisement` в MetalLB?

A:

`BGPAdvertisement` — это Custom Resource Definition (CRD) Kubernetes, используемый в MetalLB↗, реализации балансировщика нагрузки для bare-metal Kubernetes кластеров. Он управляет тем, как диапазоны IP-адресов (определённые в `IPAddressPool`) объявляются во внешние сети через BGP (Border Gateway Protocol).

Q: Почему `BGPAdvertisement` важен?

A:

В режиме BGP MetalLB контроллер устанавливает пиринг с внешними маршрутизаторами через BGP и объявляет IP-адреса, назначенные объектам Kubernetes `Service`. Ресурс `BGPAdvertisement` позволяет:

- Управлять тем, какие пулы адресов объявляются
- Настраивать параметры объявления маршрутов, такие как:
  - Агрегация маршрутов
  - BGP сообщества
  - Локальные предпочтения (приоритет BGP)

Без определения `BGPAdvertisement` MetalLB не будет объявлять никакие адреса, даже если настроены BGP пиры.

Создание пула внешних IP-адресов через веб-консоль

1. Перейдите в **Platform Management**.
2. В левой навигационной панели выберите **Network Management** > **External IP Address Pool**.
3. Нажмите **Create External IP Address Pool**.
4. Следуйте инструкциям для настройки параметров.

Параметр	Описание
Type	• L2: Коммуникация и пересылка на основе MAC-адресов, подходит для небольших или локальных сетей, требующих простой и быстрой коммутации на уровне 2, с преимуществами в простоте настройки и низкой задержке. • BGP (Alpha): Маршрутизация и пересылка на основе IP-адресов, использующая протокол BGP для обмена маршрутной информацией, подходит для крупных сетей с необходимостью сложной маршрутизации между несколькими автономными системами, с преимуществами в высокой масштабируемости и надежности.
IP Resources	Поддерживается ввод в форматах CIDR и диапазона IP. Нажмите **Add** для добавления нескольких записей, примеры: **CIDR**: `192.168.1.1/24`. **Диапазон IP**: `192.168.2.1` ~ `192.168.2.255`.
Available Nodes	В режиме L2 доступные узлы — это те, которые используются для передачи всего трафика VIP; в режиме BGP доступные узлы — это те, которые несут VIP, устанавливают BGP-соединения с пирами и объявляют маршруты во внешнюю сеть. • **Имя узла**: выбор доступных узлов по имени.

Параметр	Описание
	• **Выбор по меткам**: выбор доступных узлов по меткам. • **Показать детали узлов**: просмотр итогового списка доступных узлов. **Примечание**: • При использовании типа BGP доступные узлы — это узлы следующего перехода; убедитесь, что выбранные доступные узлы являются подмножеством BGP Connection Nodes. • Можно настроить либо выбор по меткам, либо по имени узла; если заданы оба варианта, итоговые доступные узлы — пересечение обоих.
BGP Peers	Выберите BGP пиры; подробности настройки смотрите в разделе BGP Peers.

5. Нажмите **Create**.

Создание пула внешних IP-адресов через CLI

```
kubectl apply -f ippool-with-L2advertisement.yaml -f ippool-with-bgpadvertisement.yaml
```

Просмотр политики оповещений

1. Перейдите в **Platform Management**.
2. В левой навигационной панели выберите **Network Management** > **External IP Address Pool**.
3. Нажмите **View Alarm Policy** в правом верхнем углу страницы, чтобы просмотреть общую политику оповещений для MetalLB.

Создание BGP-пиров

Узлы, которые устанавливают соединения для обмена маршрутной информацией либо между разными AS, либо внутри одного AS, и которые обмениваются данными по протоколу BGP.

Содержание

Терминология

Термин	Объяснение
AS Number	AS — это совокупность маршрутизаторов, управляемых одной технической административной организацией и использующих единую политику маршрутизации. Каждому AS в сети BGP присваивается уникальный номер AS для отличия от других AS. Номера AS делятся на 2-байтовые и 4-байтовые. • Диапазон 2-байтовых номеров AS — 1~65535, где 1~64511 — зарегистрированные публичные номера AS в Интернете, аналогичные публичным IP-адресам; 64512~65535 — приватные номера AS, аналогичные приватным IP-адресам.

Термин	Объяснение
	• Диапазон 4-байтовых номеров AS — 1~4294967295. Устройства, поддерживающие 4-байтовые номера AS, совместимы с устройствами, поддерживающими 2-байтовые номера AS.

Предварительные требования

Пожалуйста, свяжитесь с администратором для включения соответствующих функций.

Пример пользовательского ресурса BGPPeer (CR)

```yaml
# test-bgb-example.yaml
apiVersion: metallb.io/v1beta2
kind: BGPPeer
metadata:
  name: example
  namespace: metallb-system
spec:
  myASN: 64512
  peerASN: 64512
  peerAddress: 172.30.0.3
  peerPort: 180
  nodeSelectors:
    - matchLabels:
        alertmanager: "true"
```

Создание BGPPeer через веб-консоль

1. Перейдите в **Platform Management**.

2. В левой навигационной панели выберите **Network Management** > **BGP Peers**.

3. Нажмите **Create BGP Peer**.

4. Следуйте инструкциям ниже для настройки параметров.

Параметр	Описание
Local AS Number	Номер AS, к которому принадлежит узел, устанавливающий BGP-соединение. **Примечание**: Если нет особых требований, рекомендуется использовать конфигурацию IBGP, то есть локальный номер AS должен совпадать с номером AS пира.
Peer AS Number	Номер AS, к которому принадлежит BGP-пир.
Peer IP	IP-адрес BGP-пира, который должен быть действительным IP-адресом, способным установить BGP-соединение.
Local IP	IP-адрес узла, устанавливающего BGP-соединение. Если у узла несколько IP-адресов, выберите конкретный локальный IP для установления BGP-соединения с пиром.
Peer Port	Номер порта BGP-пира.
BGP Connected Node	Узел, который устанавливает BGP-соединение. Если этот параметр не задан, BGP-соединения будут устанавливаться со всеми узлами.
eBGP Multi-Hop	Позволяет устанавливать BGP-сессии между BGP-маршрутизаторами, которые не связаны напрямую. При включении этой функции значение TTL по умолчанию для BGP-пакетов равно 5, что позволяет устанавливать пиринговые отношения BGP через несколько промежуточных сетевых устройств, делая дизайн сети более гибким.
RouterID	32-битное числовое значение (обычно представлено в формате с точками, аналогично формату IPv4-адреса), используемое для уникальной идентификации BGP-маршрутизатора в сети BGP.

Параметр	Описание
	Обычно применяется для установления соседских отношений BGP, обнаружения петель маршрутизации, выбора оптимальных путей и устранения неполадок в сети.

5. Нажмите **Create**.

Создание BGPPeer через CLI

```
kubectl apply -f test-bgb-example.yaml
```

Настройка подсетей

Содержание

Правила выделения IP

NOTE

Если проекту или пространству имён назначено несколько подсетей, IP-адрес будет случайным образом выбран из одной из подсетей.

- Выделение для проекта:
  - Если проект не привязан к подсети, Pods во всех пространствах имён этого проекта могут использовать IP-адреса только из подсети по умолчанию. Если в подсети по умолчанию недостаточно IP-адресов, Pods не смогут запуститься.
  - Если проект привязан к подсети, Pods во всех пространствах имён этого проекта могут использовать IP-адреса только из этой конкретной подсети.
- Выделение для пространства имён:
  - Если пространство имён не привязано к подсети, Pods в этом пространстве имён могут использовать IP-адреса только из подсети по умолчанию. Если в подсети по умолчанию недостаточно IP-адресов, Pods не смогут запуститься.

- Если пространство имён привязано к подсети, Pods в этом пространстве имён могут использовать IP-адреса только из этой конкретной подсети.

Сеть Calico

Создание подсетей в сети Calico для достижения более тонкой изоляции ресурсов внутри кластера.

Ограничения и особенности

В среде кластера с IPv6 подсети, создаваемые в сети Calico, по умолчанию используют инкапсуляцию VXLAN. Порты, необходимые для инкапсуляции VXLAN, отличаются от портов для инкапсуляции IPIP. Необходимо убедиться, что открыт UDP порт 4789.

Пример custom resource (CR) подсети с сетью Calico

```yaml
# test-calico-subnet.yaml
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: test-calico
spec:
  cidrBlock: 10.1.1.1/24
  default: false (1)
  ipipMode: Always (2)
  natOutgoing: true (3)
  private: false
  protocol: Dual
  v4blockSize: 30
```

(1) При `default` true используется инкапсуляция VXLAN.

(2) См. параметры Encapsulation Mode и Encapsulation Protocol.

(3) См. параметры Outbound Traffic NAT.

Создание подсети в сети Calico через веб-консоль

1. Перейдите в **Platform Management**.
2. В левой навигационной панели выберите **Network Management** > **Subnets**.
3. Нажмите **Create Subnet**.
4. Настройте параметры согласно следующим инструкциям.

Параметр	Описание
CIDR	После назначения подсети проекту или пространству имён, контейнерные группы в этом пространстве будут случайным образом использовать IP-адреса из этого CIDR для связи. **Примечание**: Соответствие между CIDR и BlockSize смотрите в Reference Content.
Encapsulation Protocol	Выберите протокол инкапсуляции. **IPIP** не поддерживается в режиме dual-stack. • **IPIP**: реализует межсегментную связь с помощью протокола IPIP. • **VXLAN (Alpha)**: реализует межсегментную связь с помощью протокола VXLAN. • **No Encapsulation**: прямое соединение через маршрутизацию.
Encapsulation Mode	При протоколе инкапсуляции **IPIP** или **VXLAN** необходимо указать режим инкапсуляции, по умолчанию **Always**. • **Always**: всегда включать туннели IPIP / VXLAN. • **Cross Subnet**: включать туннели IPIP / VXLAN только если хосты находятся в разных подсетях; при нахождении в одной подсети — прямое соединение через маршрутизацию.
Outbound Traffic NAT	Выберите, включать ли NAT исходящего трафика (Network Address Translation), по умолчанию включено.

Параметр	Описание
	Используется для установки адреса доступа, видимого из внешней сети, когда контейнерные группы подсети выходят в интернет. При включенном NAT исходящего трафика в качестве адреса доступа используется IP хоста; при отключённом — IP контейнерных групп подсети напрямую видны во внешней сети.

5. Нажмите **Confirm**.
6. На странице с деталями подсети выберите **Actions** > **Allocate Project** / **Allocate Namespace**.
7. Завершите настройку и нажмите **Allocate**.

Создание подсети в сети Calico через CLI

```
kubectl apply -f test-calico-subnet.yaml
```

Reference Content

Динамическое соответствие между CIDR и blockSize показано в таблице ниже.

CIDR	blockSize Size	Количество хостов	Размер одного пула IP
`prefix<=16`	26	1024+	64
`16<prefix<=19`	27	256~1024	32
`prefix=20`	28	256	16
`prefix=21`	29	256	8
`prefix=22`	30	256	4
`prefix=23`	30	128	4

CIDR	blockSize Size	Количество хостов	Размер одного пула IP
`prefix=24`	30	64	4
`prefix=25`	30	32	4
`prefix=26`	31	32	2
`prefix=27`	31	16	2
`prefix=28`	31	8	2
`prefix=29`	31	4	2
`prefix=30`	31	2	2
`prefix=31`	31	1	2

NOTE

Подсети с префиксом больше 31 не поддерживаются.

Сеть Kube-OVN

Создание подсети в Overlay-сети Kube-OVN для более тонкой изоляции ресурсов в кластере.

NOTE

Платформа имеет встроенную подсеть **join** для связи между нодами и Pods; избегайте конфликтов сетевых сегментов между **join** и вновь создаваемыми подсетями.

Пример custom resource (CR) подсети с Overlay-сетью Kube-OVN

```yaml
# test-overlay-subnet.yaml
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: test-overlay-subnet
spec:
  default: false
  protocol: Dual
  cidrBlock: 10.1.0.0/23
  natOutgoing: true (1)
  excludeIps: (2)
    - 10.1.1.2
  gatewayType: distributed (3)
  gatewayNode: "" (4)
  private: false
  enableEcmp: false (5)
```

(1) См. параметры Outbound Traffic NAT.

(2) См. параметры Reserved IP.

(3) См. параметры Gateway Type. Доступные значения: `distributed` или `centralized`.

(4) См. параметры Gateway Nodes.

(5) См. параметры ECMP. Требуется предварительно связаться с администратором для включения feature gate.

Создание подсети в Overlay-сети Kube-OVN через веб-консоль

1. Перейдите в **Platform Management**.

2. В левой навигационной панели выберите **Network Management** > **Subnet**.

3. Нажмите **Create Subnet**.

4. Настройте параметры согласно следующим инструкциям.

Параметр	Описание
Network Segment	После назначения подсети проекту или пространству имён IP из этого сегмента будут случайным образом выделяться для использования Pods.
Reserved IP	Указанные зарезервированные IP не будут автоматически выделяться. Например, могут использоваться как фиксированные IP для вычислительных компонентов.
Gateway Type	Выберите тип шлюза для подсети для управления исходящим трафиком. - **Distributed**: каждый хост в кластере может выступать в роли исходящего узла для Pods на текущем хосте, обеспечивая распределённый выход в сеть. - **Centralized**: все Pods кластера используют один или несколько конкретных хостов в качестве исходящих узлов, что облегчает внешний аудит и контроль межсетевого экрана. Настройка нескольких централизованных **gateway nodes** обеспечивает высокую доступность.
ECMP (Alpha)	При выборе **Centralized** gateway можно использовать функцию ECMP. По умолчанию шлюз работает в режиме мастер-слейв, только мастер-шлюз обрабатывает трафик. При включении ECMP (Equal-Cost Multipath Routing) исходящий трафик маршрутизируется по нескольким равнозначным путям ко всем доступным узлам шлюза, увеличивая суммарную пропускную способность шлюза. **Примечание**: необходимо предварительно включить соответствующие функции.
Gateway Nodes	При использовании **Centralized** gateway выберите один или несколько конкретных хостов в качестве узлов шлюза.
Outbound Traffic NAT	Выберите, включать ли NAT исходящего трафика (Network Address Translation). По умолчанию включено. Используется для установки адреса доступа, видимого из внешней сети, когда Pods подсети выходят в интернет.

Параметр	Описание
	При включенном NAT исходящего трафика в качестве адреса доступа используется IP хоста; при отключённом — IP Pods подсети напрямую видны во внешней сети. В этом случае рекомендуется использовать централизованный шлюз.

5. Нажмите **Confirm**.
6. На странице с деталями подсети выберите **Actions** > **Allocate Project** / **Namespace**.
7. Завершите настройку и нажмите **Allocate**.

Создание подсети в Overlay-сети Kube-OVN через CLI

```
kubectl apply -f test-overlay-subnet.yaml
```

Underlay-сеть

Создание подсетей в Underlay-сети Kube-OVN не только обеспечивает более тонкую изоляцию ресурсов, но и улучшает производительность.

INFO

Сетевая инфраструктура контейнеров в Kube-OVN Underlay требует поддержки физической сети. Пожалуйста, ознакомьтесь с лучшими практиками Preparing the Kube-OVN Underlay Physical Network для обеспечения сетевой связности.

Инструкция по использованию

Общий процесс создания подсетей в Underlay-сети Kube-OVN: Добавить мостовую сеть > Добавить VLAN > Создать подсеть.

1. Имя сетевой карты по умолчанию.
2. Настройка сетевой карты по узлам.

Добавление мостовой сети через веб-консоль (опционально)

```yaml
# test-provider-network.yaml
kind: ProviderNetwork
apiVersion: kubeovn.io/v1
metadata:
  name: test-provider-network
spec:
  defaultInterface: eth1 (1)
  customInterfaces: (2)
    - interface: eth2
      nodes:
        - node1
  excludeNodes:
    - node2
```

(1) Имя сетевой карты по умолчанию.

(2) Настройка сетевой карты по узлам.

Мостовая сеть — это мост, после привязки сетевой карты к мосту она может передавать трафик контейнерной сети, обеспечивая связь с физической сетью.

Процедура:

1. Перейдите в **Platform Management**.
2. В левой навигационной панели выберите **Network Management** > **Bridge Network**.
3. Нажмите **Add Bridge Network**.
4. Настройте параметры согласно следующим инструкциям.

   Примечание:

   - *Target Pod* — все Pods, запланированные на текущем узле, или Pods в пространствах имён, привязанных к определённым подсетям, запланированным на текущем узле. Это зависит от области действия подсети под мостовой сетью.
   - Узлы в Underlay подсети должны иметь несколько сетевых карт, и сетевая карта, используемая мостовой сетью, должна быть выделена исключительно для

Underlay и не должна нести другой трафик, например SSH. Например, если в мостовой сети три узла, планирующие eth0, eth0, eth1 для эксклюзивного использования Underlay, то сетевая карта по умолчанию может быть eth0, а для третьего узла — eth1.

Параметр	Описание
Default Network Card Name	По умолчанию целевой Pod будет использовать эту сетевую карту моста для связи с физической сетью.
Configure Network Card by Node	Целевые Pods на указанных узлах будут подключаться к указанной сетевой карте вместо сетевой карты по умолчанию.
Exclude Nodes	Исключённые узлы не будут иметь мостового подключения для Pods. **Примечание**: Pods на исключённых узлах не смогут взаимодействовать с физической сетью или контейнерными сетями других узлов, поэтому следует избегать планирования соответствующих Pods на эти узлы.

5. Нажмите **Add**.

Добавление мостовой сети через CLI

```
kubectl apply -f test-provider-network.yaml
```

Добавление VLAN через веб-консоль (опционально)

```
# test-vlan.yaml
kind: Vlan
apiVersion: kubeovn.io/v1
metadata:
  name: test-vlan
spec:
  id: 0 (1)
  provider: test-provider-network (2)
```

(1) VLAN ID.

(2) Ссылка на мостовую сеть.

Платформа имеет преднастроенную виртуальную LAN **ovn-vlan**, которая подключается к мостовой сети **provider**. Вы также можете настроить новую VLAN для подключения к другим мостовым сетям, обеспечивая изоляцию между VLAN.

Процедура:

1. Перейдите в **Platform Management**.
2. В левой навигационной панели выберите **Network Management** > **VLAN**.
3. Нажмите **Add VLAN**.
4. Настройте параметры согласно следующим инструкциям.

Параметр	Описание
VLAN ID	Уникальный идентификатор VLAN, используемый для различения виртуальных LAN.
Bridge Network	VLAN будет подключена к этой мостовой сети для связи с физической сетью.

5. Нажмите **Add**.

Добавление VLAN через CLI

```
kubectl apply -f test-vlan.yaml
```

Пример custom resource (CR) подсети с Underlay-сетью Kube-OVN

```yaml
# test-underlay-network.yaml
apiVersion: kubeovn.io/v1
kind: Subnet
metadata:
  name: test-underlay-network
spec:
  default: false
  protocol: Dual
  cidrBlock: 11.1.0.0/23
  gateway: 11.1.0.1
  excludeIps:
    - 11.1.0.3
  private: false
  allowSubnets: []
  vlan: test-vlan (1)
  enableEcmp: false
```

(1) Ссылка на VLAN.

Создание подсети в Underlay-сети Kube-OVN через веб-консоль

> **NOTE**
>
> Платформа также преднастроила подсеть **join** для связи между нодами и Pods в режиме Overlay. Эта подсеть не используется в режиме Underlay, поэтому важно избегать конфликтов IP-сегментов между **join** и другими подсетями.

Процедура:

1. Перейдите в **Platform Management**.
2. В левой навигационной панели выберите **Network Management** > **Subnet**.
3. Нажмите **Create Subnet**.

4. Настройте параметры согласно следующим инструкциям.

Параметр	Описание
VLAN	VLAN, к которой принадлежит подсеть.
Subnet	После назначения подсети проекту или пространству имён IP из физической подсети будут случайным образом выделяться для использования Pods.
Gateway	Физический шлюз в указанной подсети.
Reserved IP	Указанные зарезервированные IP не будут автоматически назначаться. Например, могут использоваться как фиксированные IP вычислительных компонентов.

5. Нажмите **Confirm**.

6. На странице с деталями подсети выберите **Action** > **Assign Project** / **Namespace**.

7. Завершите настройку и нажмите **Assign**.

Создание подсети в Underlay-сети Kube-OVN через CLI

```
kubectl apply -f test-underlay-network.yaml
```

Связанные операции

Если в кластере одновременно существуют подсети Underlay и Overlay, при необходимости можно настроить Автоматическую взаимосвязь между подсетями Underlay и Overlay.

Управление подсетями

Обновление шлюза через веб-консоль

Включает изменение способа исходящего трафика, узлов шлюза и конфигурации NAT.

1. Перейдите в **Platform Management**.
2. В левой панели выберите **Network Management** > **Subnets**.
3. Нажмите на имя подсети.
4. Выберите **Action** > **Update Gateway**.
5. Обновите параметры; подробности смотрите в Описание параметров.
6. Нажмите **OK**.

Обновление шлюза через CLI

```
kubectl patch subnet test-overlay-subnet --type=json -p='[
  {"op": "replace", "path": "/spec/gatewayType", "value": "centralized"},
  {"op": "replace", "path": "/spec/gatewayNode", "value": "192.168.66.210"},
  {"op": "replace", "path": "/spec/natOutgoing", "value": true},
  {"op": "replace", "path": "/spec/enableEcmp", "value": true}
]'
```

Обновление зарезервированных IP через веб-консоль

IP шлюза нельзя удалить из зарезервированных IP, остальные зарезервированные IP можно редактировать, удалять или добавлять.

1. Перейдите в **Platform Management**.
2. В левой панели выберите **Network Management** > **Subnets**.
3. Нажмите на имя подсети.
4. Выберите **Action** > **Update Reserved IP**.
5. После внесения изменений нажмите **Update**.

Обновление зарезервированных IP через CLI

```
kubectl patch subnet test-overlay-subnet --type=json -p='[
  {
    "op": "replace",
    "path": "/spec/excludeIps",
    "value": ["10.1.0.1", "10.1.1.2", "10.1.1.4"]
  }
]'
```

Назначение проектов через веб-консоль

Назначение подсетей конкретным проектам помогает командам лучше управлять и изолировать сетевой трафик для разных проектов, обеспечивая достаточные сетевые ресурсы для каждого проекта.

1. Перейдите в **Platform Management**.
2. В левой панели выберите **Network Management** > **Subnets**.
3. Нажмите на имя подсети.
4. Выберите **Action** > **Assign Project**.
5. После добавления или удаления проектов нажмите **Assign**.

Назначение проектов через CLI

```
kubectl patch subnet test-overlay-subnet --type=json -p='[
  {
    "op": "replace",
    "path": "/spec/namespaceSelectors",
    "value": [
      {
        "matchLabels": {
          "cpaas.io/project": "cong"
        }
      }
    ]
  }
]'
```

Назначение пространств имён через веб-консоль

Назначение подсетей конкретным пространствам имён позволяет добиться более тонкой сетевой изоляции.

Примечание: Процесс назначения приведёт к перестройке шлюза, и исходящие пакеты будут отброшены! Пожалуйста, убедитесь, что в данный момент нет бизнес-приложений, обращающихся к внешним кластерам.

1. Перейдите в **Platform Management**.
2. В левой панели выберите **Network Management** > **Subnets**.
3. Нажмите на имя подсети.
4. Выберите **Action** > **Assign Namespace**.
5. После добавления или удаления пространств имён нажмите **Assign**.

Назначение пространств имён через CLI

```
kubectl patch subnet test-overlay-subnet --type=json -p='[
  {
    "op": "replace",
    "path": "/spec/namespaces",
    "value": ["cert-manager"]
  }
]'
```

Расширение подсетей через веб-консоль

Когда диапазон зарезервированных IP подсети достигает предела использования или близок к исчерпанию, его можно расширить на основе исходного диапазона подсети без влияния на нормальную работу существующих сервисов.

1. Перейдите в **Platform Management**.
2. В левой панели выберите **Network Management** > **Subnets**.
3. Нажмите на имя подсети.

4. Выберите **Action** > **Expand Subnet**.

5. Завершите настройку и нажмите **Update**.

Расширение подсетей через CLI

```
kubectl patch subnet test-overlay-subnet --type=json -p='[
  {
    "op": "replace",
    "path": "/spec/cidrBlock",
    "value": "10.1.0.0/22"
  }
]'
```

Управление сетями Calico

Поддерживается назначение проектов и пространств имён; подробности смотрите в разделах назначение проектов и назначение пространств имён.

Удаление подсети через веб-консоль

NOTE

- При удалении подсети, если есть контейнерные группы, использующие IP из этой подсети, они смогут продолжать работу с сохранением IP, но не смогут обмениваться сетевым трафиком. Контейнерные группы можно пересоздать для использования IP из подсети по умолчанию или назначить новое подсеть для пространства имён, в котором они находятся.
- Подсеть по умолчанию удалить нельзя.

1. Перейдите в **Platform Management**.

2. В левой навигационной панели выберите **Network Management** > **Subnets**.

3. Нажмите ⋮ > **Delete** и подтвердите удаление.

Удаление подсети через CLI

```
kubectl delete subnet test-overlay-subnet
```

Создание сетевых политик

INFO

Платформа теперь предоставляет два разных интерфейса для работы с сетевыми политиками. Старый интерфейс поддерживается для совместимости, а новый более гибкий и включает встроенный YAML-редактор. Рекомендуется использовать новую версию.

Пожалуйста, свяжитесь с администратором платформы, чтобы включить feature gate `network-policy-next` для доступа к новому интерфейсу.

NetworkPolicy — это ресурс Kubernetes с областью видимости в пределах namespace, реализуемый плагинами CNI.
С помощью сетевых политик можно контролировать сетевой трафик Pod’ов, обеспечивая сетевую изоляцию и снижая риск атак.

По умолчанию все Pod’ы могут свободно обмениваться трафиком, разрешая входящий и исходящий трафик из любых источников.
При применении NetworkPolicy целевые Pod’ы будут принимать только трафик, соответствующий спецификации.

WARNING

Сетевые политики применяются только к трафику контейнеров. Они не влияют на Pod’ы, работающие в режиме **hostNetwork**.

Пример NetworkPolicy:

```yaml
# example-network-policy.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: example
  namespace: demo-1
  annotations:
    cpaas.io/display-name: test
spec:
  podSelector:
    matchLabels:
      pod-template-hash: 55c84b59bb
  ingress:
    - ports:
        - protocol: TCP
          port: 8989
      from: (1)
        - podSelector:
            matchLabels:
              kubevirt.io/vm: test
  egress:
    - ports:
        - protocol: TCP
          port: 80
      to:
        - ipBlock:
            cidr: 192.168.66.221/23
            except: []
  policyTypes:
    - Ingress
    - Egress
```

(1) `from` и `to` поддерживают `namespaceSelector`, `podSelector`, `ipBlock`

Содержание

Справка

Создание NetworkPolicy через веб-консоль

1. Войдите в **Container Platform**.
2. В левой навигационной панели выберите **Network** > **Network Policies**.
3. Нажмите **Create Network Policy**.
4. Следуйте инструкциям ниже для заполнения соответствующих параметров.

Область	Параметр	Описание
Целевой Pod	Выбор Pod	Введите метки целевых Pod в формате ключ-значение; если не указано, политика применяется ко всем Pod в текущем namespace.
	Предварительный просмотр целевых Pod, затронутых текущей политикой	Нажмите **Preview**, чтобы увидеть целевые Pod, на которые влияет данная сетевая политика.
Ingress	Блокировать весь входящий трафик	Блокирует весь входящий трафик к целевому Pod. **Примечание**: • Если в поле `spec.policyTypes` в YAML добавлен Ingress без настройки конкретных правил, при возврате к форме опция **Блокировать весь входящий трафик** будет автоматически установлена.

Область	Параметр		Описание
			• Если одновременно удалить поля `spec.ingress`, `spec.egress` и `spec.policyTypes` в YAML, при возврате к форме опция **Блокировать весь входящий трафик** также будет автоматически установлена.
	Правила **Описание**: Если в правилах добавлено несколько источников, между ними действует логическое **ИЛИ**.	Pod’ы в текущем namespace	Соответствуют Pod’ам с указанными метками в текущем namespace; только такие Pod могут обращаться к целевому Pod. Можно нажать **Preview**, чтобы увидеть Pod’ы, затронутые текущим правилом. Если этот параметр не настроен, по умолчанию все Pod’ы в текущем namespace имеют доступ к целевому Pod.
		Pod’ы в текущем кластере	Соответствуют namespace или Pod’ам с указанными метками в кластере; только такие Pod могут обращаться к целевому Pod. Можно нажать **Preview**, чтобы увидеть Pod’ы, затронутые текущим правилом. • Если настроены и селекторы namespace, и Pod, будет взято пересечение — то есть выбраны Pod с указанными метками из указанного namespace. • Если этот параметр не настроен, по умолчанию все

Область	Параметр		Описание
			Pod’ы из всех namespace в кластере имеют доступ к целевому Pod.
		IP-диапазон	Введите CIDR, который может обращаться к целевому Pod, и можно исключить CIDR-диапазоны, которым доступ запрещён. Если этот параметр не настроен, любой трафик может обращаться к целевому Pod. **Описание**: Можно добавить исключения в формате *example_ip*/32 для исключения отдельного IP-адреса.
	Порт		Соответствует трафику на указанных протоколах и портах; можно добавить числовые порты или имена портов на Pod’ах. Если этот параметр не настроен, будут соответствовать все порты.

Область	Параметр	Описание
Egress	Блокировать весь исходящий трафик	Блокирует весь исходящий трафик от целевого Pod. **Примечание**: • Если в поле `spec.policyTypes` в YAML добавлен Egress без настройки конкретных правил, при возврате к форме опция **Блокировать весь исходящий трафик** будет автоматически установлена.
	Другие параметры	Аналогично параметрам **Ingress**, здесь не приводится подробное описание.

1. Нажмите **Create**.

Создание NetworkPolicy через CLI

```
kubectl apply -f example-network-policy.yaml
```

Справка

Для получения дополнительной информации ознакомьтесь с официальной документацией по Network Policies↗.

Создание Admin Network Policies

INFO

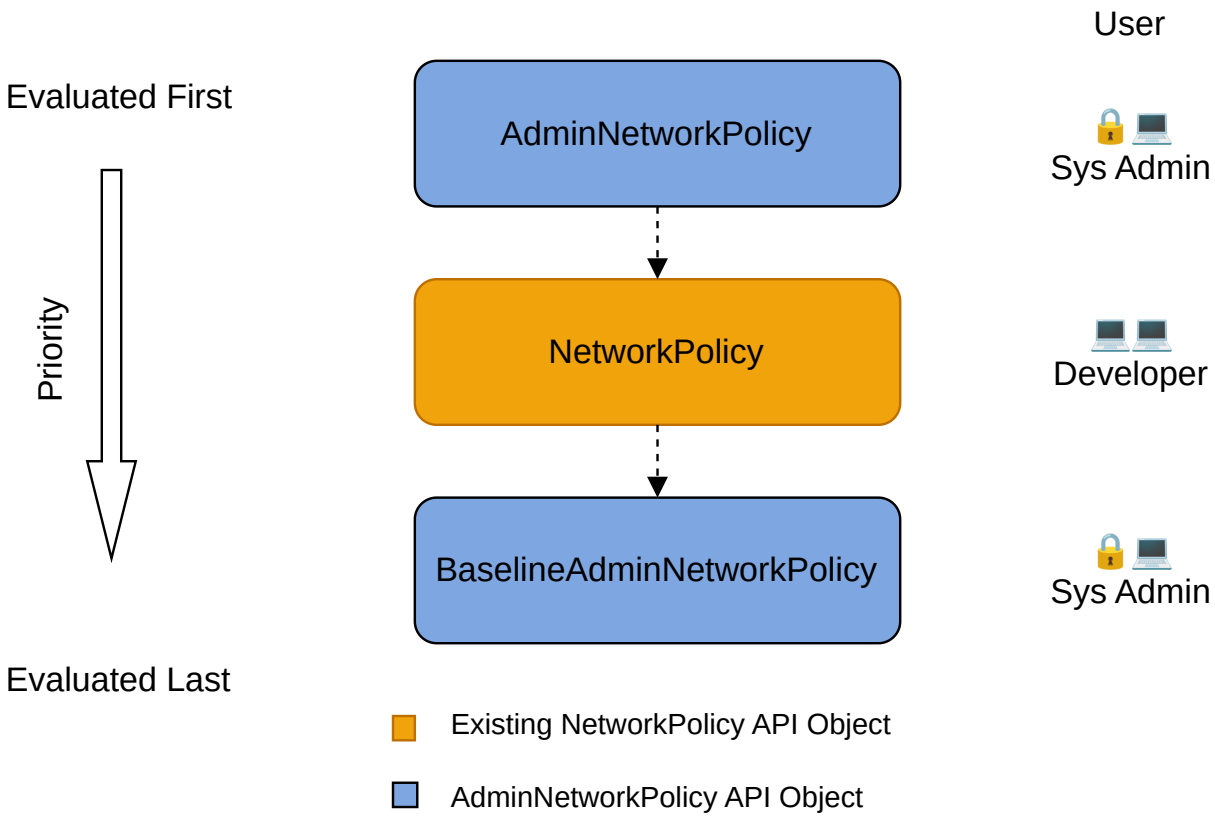
Платформа теперь предоставляет два разных интерфейса для Cluster Network Policies. Старый интерфейс поддерживается для совместимости, а новый более гибкий и предоставляет нативный YAML-редактор. Рекомендуется использовать новую версию.

Пожалуйста, свяжитесь с администратором платформы для включения feature-gate `cluster-network-policy` и `cluster-network-policy-next` для доступа к новому интерфейсу.

Новая cluster network policy использует стандартный дизайн Kubernetes сообщества Admin Network Policy↗, предоставляя более гибкие методы настройки и богатые опции конфигурации.

При применении нескольких сетевых политик соблюдается строгий порядок приоритетов: Admin Network Policy имеет приоритет над Network Policy, которая в свою очередь имеет приоритет над Baseline Admin Network Policy.

Процедура следующая:



Содержание

Примечания

- Административные сетевые политики поддерживаются только в Kube-OVN CNI.
- В сетевом режиме Kube-OVN эта функция находится на уровне Alpha.
- В кластере может существовать только одна Baseline Admin Network Policy.

AdminNetworkPolicy

```yaml
# example-anp.yaml
apiVersion: policy.networking.k8s.io/v1alpha1
kind: AdminNetworkPolicy
metadata:
  name: example-anp
spec:
  priority: 3 (1)
  subject: (2)
    pods:
      namespaceSelector:
        matchLabels: {}
      podSelector:
        matchLabels:
          pod-template-hash: 55f66dd67d
  ingress:
    - name: ingress1
      action: Allow (3)
      ports:
        - portNumber:
            protocol: TCP
            port: 8090
      from: (4)
        - pods:
            namespaceSelector:
              matchLabels: {}
            podSelector:
              matchLabels:
                pod-template-hash: 55c84b59bb
  egress:
    - name: egress1
      action: Allow
      ports:
        - portNumber:
            protocol: TCP
            port: 8080
      to: (5)
        - networks:
            - 10.1.1.1/23
```

(1) Чем меньше число, тем выше приоритет.

2 `subject`: Можно указать не более одного селектора — либо namespaceSelector, либо podSelector.

3 `action`: Доступные значения — Allow, Deny и Pass. Allow разрешает доступ трафика, Deny запрещает доступ, Pass разрешает трафик и пропускает последующие политики с более низким приоритетом, позволяя обработку трафика другими политиками (NetworkPolicy и BaselineAdminNetworkPolicy).

4 Доступные значения — Namespace Selector, Pod Selector.

5 Доступные значения — Namespace Selector, Pod Selector, Node Selector, IP Block.

BaselineAdminNetworkpolicy:

```yaml
# default.yaml
apiVersion: policy.networking.k8s.io/v1alpha1
kind: BaselineAdminNetworkPolicy
metadata:
  name: default (1)
spec:
  subject:
    pods:
      namespaceSelector:
        matchLabels: {}
      podSelector:
        matchLabels:
          pod-template-hash: 55c84b59bb
  ingress:
    - name: ingress1
      action: Allow
      ports:
        - portNumber:
            protocol: TCP
            port: 8888
      from:
        - pods:
            namespaceSelector:
              matchLabels: {}
            podSelector:
              matchLabels:
                pod-template-hash: 55f66dd67d
  egress:
    - name: egress1
      action: Allow (2)
      ports:
        - portNumber:
            protocol: TCP
            port: 8080
      to:
        - networks:
            - 3.3.3.3/23
```

(1) В кластере может быть создана только одна baseline admin network policy с metadata.name= `default`.

(2) Доступные значения — Allow, Deny.

Создание AdminNetworkPolicy или BaselineAdminNetworkPolicy через веб-консоль

1. Перейдите в **Platform Management**.
2. В левой навигационной панели выберите **Network** > **Cluster Network Policies**.
3. Нажмите **Create Admin Network Policies** или **Configure the Baseline Admin Network Policy**.
4. Следуйте инструкциям ниже для завершения соответствующей настройки.

Область	Параметр	Описание
Основная информация	Имя	Имя Admin Network Policy или Baseline Admin Network Policy.
	Приоритет	Определяет порядок оценки и применения политик. Чем меньше числовое значение, тем выше приоритет. **Примечание:** baseline admin network policy не имеет приоритета.
Целевой Pod	Namespace Selector	Введите метки целевых Namespace в формате ключ-значение. Если не указано, политика применяется ко всем Namespace в текущем кластере. При указании политика применяется только к Pod в namespace, соответствующих этим селекторам.

Область	Параметр	Описание
	Предварительный просмотр целевых Pod, затронутых текущей политикой	Нажмите **Preview**, чтобы увидеть целевые Pod, на которые влияет эта сетевая политика.
	Pod Selector	Введите метки целевых Pod в формате ключ-значение. Если не указано, политика применяется ко всем Pod в текущем namespace.
	Предварительный просмотр целевых Pod, затронутых текущей политикой	Нажмите **Preview**, чтобы увидеть целевые Pod, на которые влияет эта сетевая политика.
Ingress	Действие с трафиком	Определяет, как обрабатывать входящий трафик к целевым Pod. Имеет три режима: **Allow** (разрешить трафик), **Deny** (запретить трафик) и **Pass** (пропустить все админские сетевые политики с более низким приоритетом, позволяя обработку трафика Network Policy или, если Network Policy отсутствует, Baseline Admin Network Policy). **Примечание:** baseline admin network policy не поддерживает действие **Pass**.

Область	Параметр		Описание
	Правило **Описание**: Если в правиле добавлено несколько источников, между ними действует логическая операция **ИЛИ**.	Pod Selector	Соответствует namespace или Pod с указанными метками в кластере; доступ к целевому Pod разрешён только для соответствующих Pod. Можно нажать **Preview**, чтобы увидеть Pod, затронутые текущим правилом. • Если настроены и namespace, и Pod селекторы, берётся их пересечение, то есть выбираются Pod с указанными метками из указанных namespace. • Если этот параметр не настроен, по умолчанию все Pod во всех namespace кластера имеют доступ к целевому Pod.
		Namespace Selector	Соответствует Pod с указанными метками в текущем namespace; доступ к целевому Pod разрешён только для соответствующих Pod. Можно нажать **Preview**, чтобы увидеть Pod, затронутые текущим

Область	Параметр		Описание
			правилом. Если этот параметр не настроен, по умолчанию все Pod в текущем namespace имеют доступ к целевому Pod.
	Порты		Соответствует трафику на указанных протоколах и портах; можно добавить числовые порты или имена портов на Pod. Если параметр не настроен, соответствуют все порты.
Egress	Правило **Описание**: Если в правиле добавлено несколько источников, между ними действует логическая операция **ИЛИ**.	Node Selector	Определяет, к каким IP-адресам узлов целевые Pod имеют доступ. Можно выбрать узлы по меткам для контроля доступа Pod к IP-адресам узлов.
		Диапазон IP	Укажите CIDR-диапазоны, к которым целевые Pod разрешено подключаться. Если параметр не настроен, по умолчанию целевые Pod могут подключаться к любым IP.
		Другие параметры	Аналогично параметрам Ingress, с теми же опциями настройки и поведением.

Создание AdminNetworkPolicy или BaselineAdminNetworkPolicy через CLI

```
kubectl apply -f example-anp.yaml -f default.yaml
```

Дополнительные ресурсы

- Configure Cluster Network Policies

Настройка сетевых политик кластера

Сетевые политики кластера отвечают за управление правилами контроля доступа **на уровне проектов**. При включении этой функции разные проекты по умолчанию изолируются друг от друга, и вычислительные компоненты в разных проектах не могут взаимодействовать по сети. Связь может быть организована путем добавления правил **доступа одного проекта** или **доступа по IP-сегменту**.

После настройки сетевые политики кластера будут синхронизированы с пространствами имён в кластере и могут быть просмотрены в модуле функции **Network Policies** платформы контейнеров.

Содержание

Примечания

- Эффективность сетевых политик кластера зависит от того, поддерживает ли используемый в кластере сетевой плагин сетевые политики.
  - Kube-OVN и Calico поддерживают сетевые политики.
  - Flannel не поддерживает сетевые политики.
  - При доступе к кластеру или использовании кастомного сетевого плагина можно обратиться к соответствующей документации для подтверждения поддержки.
- Функциональность находится на уровне Alpha в режиме сети Kube-OVN.

Процедура

1. Перейдите в **Platform Management**.
2. В левой навигационной панели выберите **Network Management** > **Cluster Network Policies**.
3. Нажмите **Configure Now**.
4. Следуйте инструкциям ниже для завершения соответствующей настройки.

Параметр настройки	**Описание**
Полная изоляция между проектами	Включение или отключение переключателя полной изоляции между проектами, который включён по умолчанию и может быть выключен нажатием. При включении достигается сетевая изоляция между всеми проектами текущего кластера, и другим ресурсам запрещён доступ к любому проекту внутри кластера (например, внешним IP, балансировщикам нагрузки). Это не влияет на доступ проектов к ресурсам вне кластера.
Доступ одного проекта	Параметр действует только при включённом переключателе **Полная изоляция между проектами**. Настройте **исходный проект** и **целевой проект** для однонаправленного доступа. Нажмите **Add** для добавления записи конфигурации, поддерживается несколько записей. В выпадающем списке **исходный проект** выберите проект, который будет иметь доступ к целевому проекту, или выберите все проекты; в списке **целевой проект** выберите проект, к которому будет осуществлён доступ.
Доступ по IP-сегменту	Параметр действует только при включённом переключателе **Полная изоляция между проектами**. Настройте конкретный **IP/сегмент** и **целевой проект** для однонаправленного доступа. Нажмите **Add** для добавления записи конфигурации,

Параметр настройки	Описание
	поддерживается несколько записей. В поле ввода **исходный IP-сегмент** введите IP или CIDR-сегмент, который будет иметь доступ к целевому проекту; в списке **целевой проект** выберите проект, к которому будет осуществлён доступ.

5. Нажмите **Configure**.

Menu

Как сделать

Развертывание высокодоступного VIP для ALB

Метод 1: Использование внутренней маршрутизации типа LoadBalancer для предоставления VIP

Метод 2: Использование внешнего устройства балансировщика нагрузки для предоставления VIP

Soft Data Center LB Solution (Alpha)

Предварительные требования

Процедура

Проверка

Подготовка физической сети Kube-OVN Underlay

Инструкции по использованию

Объяснение терминологии

Требования к среде

Пример конфигурации

Автоматическое взаимное подключение подсетей Underlay и Overlay

Процедура

Использование OAuth Proxy с ALB

Overview

Procedure

Result

Создание GatewayAPI Gateway

Развертывание MetalLB

Установка Pod Security Policies в режим Privileged

Настройка балансировщика нагрузки

Как правильно выделять ресурсы CPU и памяти

Проброс IPv6-трафика на IPv4-адреса внутри кластера

Метод настройки

Проверка результата

Calico Network поддерживает шифрование WireGuard

Статус установки

Терминология

Примечания

Требования

Процедура

Проверка результата

Kube-OVN Overlay Network поддерживает шифрование IPsec

Терминология

Примечания

Предварительные требования

Процедура

ALB Monitoring

Terminology

Procedure

Monitoring Metrics

Развертывание высокодоступного VIP для ALB

Для обеспечения высокой доступности **Load Balancer** требуется VIP. Существует два способа получения VIP.

Содержание

Метод 1: Использование внутренней маршрутизации типа LoadBalancer для предоставления VIP

При создании балансировщика нагрузки включается опция **внутренней маршрутизации**, и система автоматически создает внутреннюю маршрутизацию типа LoadBalancer для предоставления VIP балансировщику нагрузки. Перед использованием убедитесь, что текущий кластер поддерживает внутреннюю маршрутизацию типа LoadBalancer. Вы можете использовать встроенную в платформу реализацию внутренней маршрутизации LoadBalancer, для конкретной настройки смотрите External Address Pool; если опция **внутренней маршрутизации** отключена, необходимо настроить адрес доступа для балансировщика нагрузки.

Метод 2: Использование внешнего устройства балансировщика нагрузки для предоставления VIP

- Пожалуйста, согласуйте с сетевым инженером IP-адрес (публичный IP, приватный IP, VIP) или доменное имя сервиса балансировщика нагрузки перед развертыванием. Если вы хотите использовать доменное имя в качестве адреса для внешнего трафика к балансировщику нагрузки, необходимо заранее оформить доменное имя и настроить его разрешение. Рекомендуется использовать коммерческое устройство балансировщика нагрузки для предоставления VIP, в противном случае можно использовать Pure Software Data Center LB Solution (Alpha)
- В зависимости от бизнес-сценария внешний балансировщик нагрузки должен быть настроен на проверку состояния (health check) всех используемых портов, чтобы минимизировать время простоя при обновлении ALB. Конфигурация проверки состояния следующая:

Параметры проверки состояния	**Описание**
Port	• Для глобальных кластеров укажите: 11782. • Для бизнес-кластеров укажите: 1936.
Protocol	Тип протокола проверки состояния, рекомендуется использовать TCP.
Response Timeout	Время ожидания получения ответа проверки состояния, рекомендуется настроить на 2 секунды.
Check Interval	Интервал между проверками состояния, рекомендуется настроить на 5 секунд.
Unhealthy Threshold	Количество последовательных неудач, после которых состояние backend-сервера считается нерабочим, рекомендуется настроить на 3 раза.

Soft Data Center LB Solution (Alpha)

Разверните чисто программный балансировщик нагрузки (LB) для дата-центра, создав высокодоступный балансировщик нагрузки вне кластера, обеспечивающий балансировку нагрузки для нескольких ALB и стабильную работу бизнес-приложений. Поддерживается настройка только IPv4, только IPv6 или двойного стека IPv4 и IPv6.

Содержание

Предварительные требования

1. Подготовьте два или более хост-узлов в качестве LB. Рекомендуется установить на узлы LB операционную систему Ubuntu 22.04 для сокращения времени перенаправления трафика LB на аномальные backend-узлы.
2. Предварительно установите следующее программное обеспечение на все хост-узлы внешнего LB (в этом разделе приведён пример с двумя внешними LB-хостами):
   - `ipvsadm`
   - `Docker (20.10.7)`
3. Убедитесь, что служба Docker настроена на автоматический запуск при загрузке на каждом хосте с помощью команды: `sudo systemctl enable docker.service`.
4. Убедитесь, что часы каждого хост-узла синхронизированы.

5. Подготовьте образ Keepalived, используемый для запуска сервиса внешнего LB; платформа уже содержит этот образ. Адрес образа имеет следующий формат: `<image repository address>/tkestack/keepalived:<version suffix>`. Суффикс версии может незначительно отличаться в разных версиях. Получить адрес репозитория образа и суффикс версии можно следующим образом. В этом документе в качестве примера используется `build-harbor.alauda.cn/tkestack/keepalived:v3.16.0-beta.3.g598ce923`.

   - В глобальном кластере выполните `kubectl get prdb base -o json | jq .spec.registry.address` для получения параметра **image repository address**.
   - В каталоге, куда распакован установочный пакет, выполните `cat ./installer/res/artifacts.json |grep keepalived -C 2|grep tag|awk '{print $2}'|awk -F '"' '{print $2}'` для получения **version suffix**.

Процедура

Примечание: следующие операции необходимо выполнить один раз на каждом внешнем LB-хосте, при этом `hostname` хостов не должен дублироваться.

1. Добавьте следующую конфигурацию в файл `/etc/modules-load.d/alive.kmod.conf`.

```
ip_vs
ip_vs_rr
ip_vs_wrr
ip_vs_sh
nf_conntrack_ipv4
nf_conntrack
ip6t_MASQUERADE
nf_nat_masquerade_ipv6
ip6table_nat
nf_conntrack_ipv6
nf_defrag_ipv6
nf_nat_ipv6
ip6_tables
```

2. Добавьте следующую конфигурацию в файл `/etc/sysctl.d/alive.sysctl.conf`.

```
net.ipv4.ip_forward = 1
net.ipv4.conf.all.arp_accept = 1
net.ipv4.vs.conntrack = 1
net.ipv4.vs.conn_reuse_mode = 0
net.ipv4.vs.expire_nodest_conn = 1
net.ipv4.vs.expire_quiescent_template = 1
net.ipv6.conf.all.forwarding=1
```

3. Перезагрузите систему командой `reboot`.

4. Создайте папку для конфигурационного файла Keepalived.

```
mkdir -p /etc/keepalived
mkdir -p /etc/keepalived/kubecfg
```

5. Отредактируйте конфигурационные параметры согласно комментариям в следующем файле и сохраните его в папке `/etc/keepalived/` под именем `alive.yaml`.

```
instances:
  - vip: # Можно настроить несколько VIP
      vip: 192.168.128.118 # VIP должны быть уникальными
      id: 20 # ID каждого VIP должен быть уникальным, опционально
      interface: "eth0"
      check_interval: 1 # опционально, по умолчанию 1: интервал выполнения
скрипта проверки
      check_timeout: 3  # опционально, по умолчанию 3: таймаут скрипта проверки
      name: "vip-1" # Идентификатор экземпляра, может содержать только
буквенно-цифровые символы и дефисы, не может начинаться с дефиса
      peer: [ "192.168.128.116", "192.168.128.75" ] # IP узлов Keepalived,
фактически сгенерированный keepalived.conf удалит все IP интерфейса
https://github.com/osixia/docker-keepalived/issues/33
      kube_lock:
        kubecfgs: # Список kube-config, которые kube-lock будет последовательно
использовать для выборов лидера в Keepalived
          - "/live/cfg/kubecfg/kubecfg01.conf"
          - "/live/cfg/kubecfg/kubecfg02.conf"
          - "/live/cfg/kubecfg/kubecfg03.conf"
    ipvs: # Конфигурация опции IPVS
      ips: [ "192.168.143.192", "192.168.138.100","192.168.129.100" ] # IPVS backend,
замените IP узлов k8s master на IP узлов ALB
      ports: # Настройка логики проверки здоровья для каждого порта VIP
        - port: 80 # Порт виртуального сервера должен совпадать с портом
реального сервера
          virtual_server_config: |
            delay_loop 10  # Интервал выполнения проверки здоровья реального
сервера
            lb_algo rr
            lb_kind NAT
            protocol TCP
          raw_check: |
            TCP_CHECK {
                connect_timeout 10
                connect_port 1936
            }
  - vip:
      vip: 2004::192:168:128:118
      id: 102
      interface: "eth0"
      peer: [ "2004::192:168:128:75","2004::192:168:128:116" ]
      kube_lock:
        kubecfgs: # Список kube-config, которые kube-lock будет последовательно
```

```
использовать для выборов лидера в Keepalived
          - "/live/cfg/kubecfg/kubecfg01.conf"
          - "/live/cfg/kubecfg/kubecfg02.conf"
          - "/live/cfg/kubecfg/kubecfg03.conf"
    ipvs:
      ips: [ "2004::192:168:143:192","2004::192:168:138:100","2004::192:168:129:100" ]
      ports:
        - port: 80
          virtual_server_config: |
            delay_loop 10
            lb_algo rr
            lb_kind NAT
            protocol TCP
          raw_check: |
            TCP_CHECK {
                connect_timeout 1
                connect_port 1936
            }
```

6. Выполните следующую команду в бизнес-кластере для проверки срока действия сертификата в конфигурационном файле, чтобы убедиться, что сертификат ещё действителен. После истечения срока действия сертификата функциональность LB станет недоступной, потребуется обратиться к администратору платформы для обновления сертификата.

```
openssl x509 -in <(cat /etc/kubernetes/admin.conf | grep client-certificate-data | awk
'{print $NF}' | base64 -d ) -noout -dates
```

7. Скопируйте файл `/etc/kubernetes/admin.conf` с трёх Master-узлов Kubernetes кластера в папку `/etc/keepalived/kubecfg` на внешних LB-узлах, присвоив им имена с индексом, например, `kubecfg01.conf`, и измените в этих трёх файлах адреса узлов `apiserver` на реальные адреса узлов Kubernetes кластера.

   Примечание: после обновления сертификата платформы этот шаг необходимо повторить, перезаписав исходные файлы.

8. Проверьте валидность сертификатов.

   1. Скопируйте `/usr/bin/kubectl` с Master-узла бизнес-кластера на LB-узел.

   2. Выполните `chmod +x /usr/bin/kubectl` для предоставления прав на выполнение.

3. Выполните следующие команды для подтверждения валидности сертификатов.

```
kubectl --kubeconfig=/etc/keepalived/kubecfg/kubecfg01.conf get node
kubectl --kubeconfig=/etc/keepalived/kubecfg/kubecfg02.conf get node
kubectl --kubeconfig=/etc/keepalived/kubecfg/kubecfg03.conf get node
```

Если возвращаются следующие результаты, сертификат действителен.

```
kubectl --kubeconfig=/etc/keepalived/kubecfg/kubecfg01.conf get node
## Output
NAME              STATUS   ROLES                  AGE     VERSION
192.168.129.100   Ready    <none>                 7d22h   v1.25.6
192.168.134.167   Ready    control-plane,master   7d22h   v1.25.6
192.168.138.100   Ready    <none>                 7d22h   v1.25.6
192.168.143.116   Ready    control-plane,master   7d22h   v1.25.6
192.168.143.192   Ready    <none>                 7d22h   v1.25.6
192.168.143.79    Ready    control-plane,master   7d22h   v1.25.6

kubectl --kubeconfig=/etc/keepalived/kubecfg/kubecfg02.conf get node
## Output
NAME              STATUS   ROLES                  AGE     VERSION
192.168.129.100   Ready    <none>                 7d22h   v1.25.6
192.168.134.167   Ready    control-plane,master   7d22h   v1.25.6
192.168.138.100   Ready    <none>                 7d22h   v1.25.6
192.168.143.116   Ready    control-plane,master   7d22h   v1.25.6
192.168.143.192   Ready    <none>                 7d22h   v1.25.6
192.168.143.79    Ready    control-plane,master   7d22h   v1.25.6

kubectl --kubeconfig=/etc/keepalived/kubecfg/kubecfg03.conf get node
## Output
NAME              STATUS   ROLES                  AGE     VERSION
192.168.129.100   Ready    <none>                 7d22h   v1.25.6
192.168.134.167   Ready    control-plane,master   7d22h   v1.25.6
192.168.138.100   Ready    <none>                 7d22h   v1.25.6
192.168.143.116   Ready    control-plane,master   7d22h   v1.25.6
192.168.143.192   Ready    <none>                 7d22h   v1.25.6
192.168.143.79    Ready    control-plane,master   7d22h   v1.25.6
```

9. Загрузите образ Keepalived на внешний LB-узел и запустите Keepalived с помощью Docker.

```
docker run -dt --restart=always --privileged --network=host -v
/etc/keepalived:/live/cfg build-harbor.alauda.cn/tkestack/keepalived:v3.16.0-
beta.3.g598ce923
```

10. Выполните на узле, с которого осуществляется доступ к `keepalived`, команду: `sysctl -w net.ipv4.conf.all.arp_accept=1`.

Проверка

1. Выполните команду `ipvsadm -ln` для просмотра правил IPVS, вы увидите правила IPv4 и IPv6, применимые к ALB бизнес-кластера.

```
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight        ActiveConn InActConn
TCP  192.168.128.118:80 rr
  -> 192.168.129.100:80           Masq    1      0          0
  -> 192.168.138.100:80           Masq    1      0          0
  -> 192.168.143.192:80           Masq    1      0          0
TCP  [2004::192:168:128:118]:80 rr
  -> [2004::192:168:129:100]:80   Masq    1      0          0
  -> [2004::192:168:138:100]:80   Masq    1      0          0
  -> [2004::192:168:143:192]:80   Masq    1      0          0
```

2. Выключите LB-узел, на котором находится VIP, и проверьте, успешно ли VIP для IPv4 и IPv6 мигрирует на другой узел, обычно это происходит в течение 20 секунд.

3. Используйте команду `curl` на узле, не являющемся LB, чтобы проверить нормальность связи с VIP.

```
curl 192.168.128.118

<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and working.
Further configuration is required.</p>

<p>For online documentation and support please refer to <a
href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at <a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

```
curl -6 [2004::192:168:128:118]:80 -g

<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and working.
Further configuration is required.</p>

<p>For online documentation and support please refer to <a
href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Подготовка физической сети Kube-OVN Underlay

Сетевая инфраструктура контейнеров в режиме передачи Kube-OVN Underlay зависит от поддержки физической сети. Перед развертыванием сети Kube-OVN Underlay необходимо совместно с сетевым администратором спланировать и выполнить соответствующие настройки физической сети заранее, чтобы обеспечить сетевую связность.

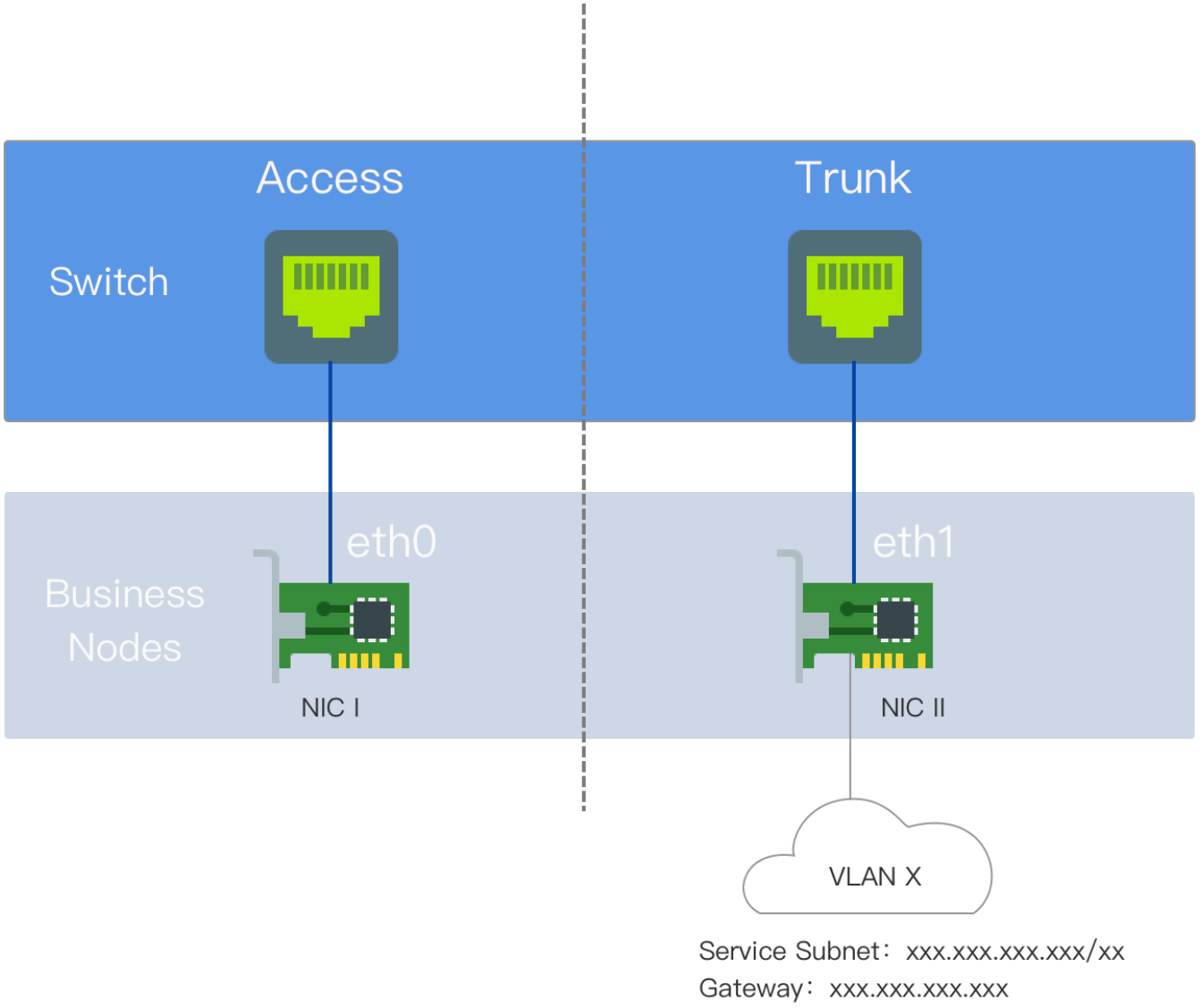
Содержание

Инструкции по использованию

Kube-OVN Underlay требует развертывания с несколькими сетевыми интерфейсными картами (NIC), при этом подсеть Underlay должна использовать исключительно один NIC. Другие типы трафика, например SSH, не должны использовать этот NIC и должны работать через другие NIC.

Перед использованием убедитесь, что на сервере узла имеется как минимум **двухсетевой NIC**, рекомендуется, чтобы скорость NIC была **не менее 10 Гбит/с или выше** (например, 10 Гбит/с, 25 Гбит/с, 40 Гбит/с).

- NIC Один: NIC с маршрутом по умолчанию, настроенный с IP-адресом, соединённый с внешним интерфейсом коммутатора, который настроен в режиме Access.
- NIC Два: NIC без маршрута по умолчанию и без настроенного IP-адреса, соединённый с внешним интерфейсом коммутатора, который настроен в режиме Trunk. Подсеть Underlay использует исключительно NIC Два.



Объяснение терминологии

VLAN (Virtual Local Area Network) — это технология, которая логически разделяет локальную сеть на несколько сегментов (или меньших LAN) для облегчения обмена

данными между виртуальными рабочими группами.

Появление технологии VLAN позволяет администраторам логически сегментировать различных пользователей в одной физической локальной сети на отдельные домены широковещания в соответствии с реальными потребностями приложений. Каждый VLAN состоит из группы рабочих станций с похожими требованиями и обладает теми же свойствами, что и физически сформированная LAN. Поскольку VLAN делятся логически, а не физически, рабочие станции в одном VLAN не ограничены одной физической областью; они могут находиться в разных физических сегментах LAN.

Основные преимущества VLAN включают:

- Сегментация портов. Даже на одном коммутаторе порты в разных VLAN не могут взаимодействовать друг с другом. Физический коммутатор может функционировать как несколько логических коммутаторов. Это часто используется для контроля взаимного доступа между разными отделами и площадками в сети.
- Безопасность сети. Разные VLAN не могут напрямую обмениваться данными, что исключает небезопасность широковещательной информации. Широковещательный и одноадресный трафик внутри VLAN не будет передаваться в другие VLAN, что помогает контролировать трафик, снижать затраты на оборудование, упрощать управление сетью и повышать безопасность сети.
- Гибкое управление. При изменении сетевой принадлежности пользователя нет необходимости менять порты или кабели; достаточно изменить конфигурацию программно.

Требования к среде

В режиме Underlay Kube-OVN связывает физический NIC с OVS и отправляет пакеты напрямую во внешнюю сеть через этот физический NIC. Возможности L2/L3 пересылки зависят от базовых сетевых устройств. Соответствующие шлюз, VLAN и политики безопасности должны быть предварительно настроены на базовых сетевых устройствах.

- **Требования к сетевой конфигурации**

- Kube-OVN проверяет доступность шлюза с помощью протокола ICMP при запуске контейнеров; базовый шлюз должен отвечать на ICMP-запросы.
- Для трафика доступа к сервисам Pods сначала отправляют пакеты на шлюз, который должен иметь возможность пересылать пакеты обратно в локальную подсеть.
- Если на коммутаторе или мосту включена функция Hairpin, **Hairpin должен быть отключён**. При использовании среды виртуальных машин VMware необходимо установить **Net.ReversePathFwdCheckPromisc** на хосте VMware в значение **1**, тогда Hairpin отключать не нужно.
- NIC, используемый для мостирования, **не может** быть **Linux Bridge**.
- Режимы агрегации NIC поддерживают Mode 0 (balance-rr), Mode 1 (active-backup), Mode 4 (802.3ad), Mode 6 (balance-alb), рекомендуется использовать 0 или 1. Другие режимы агрегации не тестировались, используйте их с осторожностью.

- **Требования к конфигурации слоя IaaS (виртуализации)**
  - Для среды виртуальных машин OpenStack необходимо отключить **PortSecurity** для соответствующего сетевого порта.
  - Для сети vSwitch VMware параметры **MAC Address Changes**, **Forged Transmits** и **Promiscuous Mode Operation** должны быть установлены в **Accept**.
  - Для публичных облаков, таких как AWS, GCE и Alibaba Cloud, сети в режиме Underlay не поддерживаются из-за отсутствия возможности задания MAC-адресов пользователем.

Пример конфигурации

В этом примере узлы — физические машины с двумя NIC. NIC Один — это NIC с маршрутом по умолчанию; NIC Два — NIC без маршрута по умолчанию и без настроенного IP-адреса, используемый исключительно для подсети Underlay. NIC Два соединён с внешним коммутатором.

- На стороне коммутатора интерфейс, подключённый к NIC Два, должен быть настроен в режиме Trunk с разрешением соответствующих VLAN.
- Настройте адрес шлюза подсети кластера на соответствующем интерфейсе vlan-interface. При необходимости двойного стека можно одновременно настроить IPv6-адрес шлюза.
- Если шлюз находится за файрволом, необходимо разрешить доступ от узлов к сети cluster-cidr.
- Конфигурация NIC сервера не требуется.

Конфигурация коммутатора

Настройка VLAN интерфейса:

```
#
interface Vlan-interface74
  ip address 192.168.74.254 255.255.255.0   //IPv4 адрес шлюза
  ipv6 address 2074::192:168:74:254/64  //IPv6 адрес шлюза
#
```

Настройка интерфейса, подключённого к NIC Два:

```
#
interface Ten-GigabitEthernet1/0/19
  port link mode bridge
  port link-type trunk  // Настройка интерфейса в режим Trunk
  undo port trunk permit vlan 1
  port trunk permit vlan 74  // Разрешение прохождения соответствующего VLAN
#
```

Проверка сетевой связности

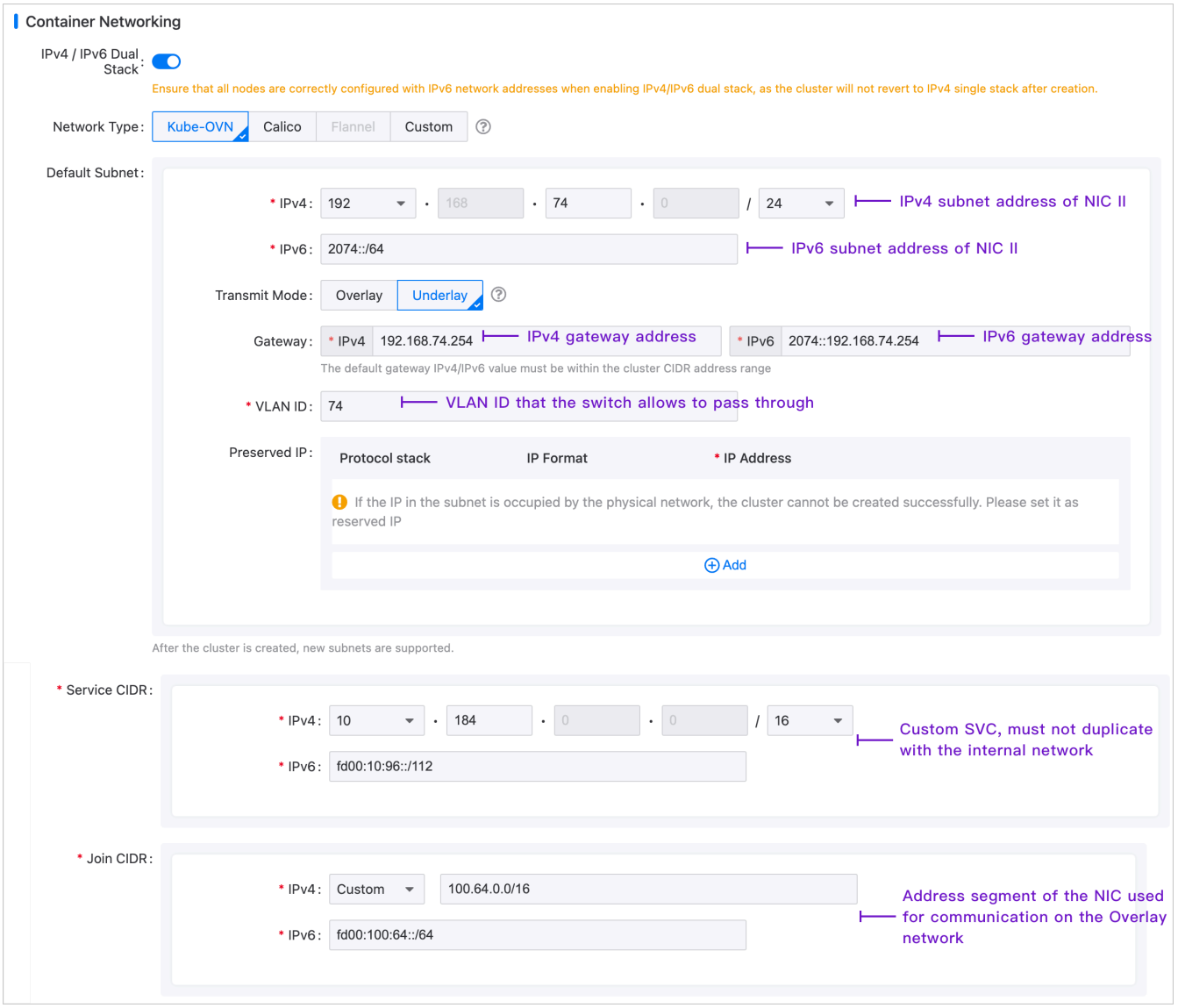
Проверьте, может ли NIC Два связаться с адресом шлюза:

```
ip link add ens224.74 link ens224 type vlan id 74  // Имя NIC – ens224, VLAN ID – 74
ip link set ens224.74 up
ip addr add 192.168.74.200/24 dev ens224.74  // Выберите тестовый адрес из подсети Underlay, здесь 192.168.74.200/24
ping 192.168.74.254  // Если пинг до шлюза успешен, значит физическая среда соответствует требованиям развертывания
ip addr del 192.168.74.200/24 dev ens224.74  // Удалите тестовый адрес после проверки
ip link del ens224.74  // Удалите подинтерфейс после проверки
```

Конфигурация платформы

В левой навигационной панели нажмите **Cluster Management > Cluster**, затем нажмите **Create Cluster**. Для подробной процедуры настройки обратитесь к документу Create Cluster, где показана конфигурация сетей контейнеров на изображении ниже.

Примечание: Подсеть Join не имеет практического значения в среде Underlay и служит в основном для последующего создания подсети Overlay, предоставляя диапазон IP-адресов, необходимый для связи между узлами и группами контейнеров.


Container Networking
IPv4 / IPv6 Dual Stack:
Ensure that all nodes are correctly configured with IPv6 network addresses when enabling IPv4/IPv6 dual stack, as the cluster will not revert to IPv4 single stack after creation.
Network Type: Kube-OVN Calico Flannel Custom
Default Subnet:
* IPv4: 192 . 168 . 74 . 0 / 24
IPv4 subnet address of NIC II
* IPv6: 2074::/64
IPv6 subnet address of NIC II
Transmit Mode: Overlay Underlay
Gateway: * IPv4 192.168.74.254
IPv4 gateway address
* IPv6 2074::192.168.74.254
IPv6 gateway address
The default gateway IPv4/IPv6 value must be within the cluster CIDR address range
* VLAN ID: 74
VLAN ID that the switch allows to pass through
Preserved IP:
Protocol stack
IP Format
* IP Address
If the IP in the subnet is occupied by the physical network, the cluster cannot be created successfully. Please set it as reserved IP
Add
After the cluster is created, new subnets are supported.
* Service CIDR:
* IPv4: 10 . 184 . 0 . 0 / 16
Custom SVC, must not duplicate with the internal network
* IPv6: fd00:10:96::/112
* Join CIDR:
* IPv4: Custom 100.64.0.0/16
Address segment of the NIC used for communication on the Overlay network
* IPv6: fd00:100:64::/64

Автоматическое взаимное подключение подсетей Underlay и Overlay

Если в кластере присутствуют подсети как Underlay, так и Overlay, по умолчанию Pods в подсети Overlay могут обращаться к IP-адресам Pods в подсети Underlay через шлюз с использованием NAT. Однако Pods в подсети Underlay для доступа к Pods в подсети Overlay должны настроить маршрутизацию на узлах.

Для автоматического взаимного подключения подсетей Underlay и Overlay можно вручную изменить YAML-файл подсети Underlay. После настройки Kube-OVN также будет использовать дополнительный IP Underlay для подключения подсети Underlay и логического маршрутизатора ovn-cluster, устанавливая соответствующие правила маршрутизации для обеспечения взаимосвязи.

Содержание

Процедура

1. Перейдите в **Platform Management**.
2. В левой навигационной панели выберите **Cluster Management** > **Resource Management**.
3. Введите **Subnet** для фильтрации объектов ресурсов.
4. Нажмите ⋮ > **Update** рядом с подсетью Underlay, которую необходимо изменить.
5. Измените YAML-файл, добавив поле `u2oInterconnection: true` в раздел `Spec`.

6. Нажмите **Update**.

Примечание: Существующие вычислительные компоненты в подсети Underlay необходимо пересоздать, чтобы изменения вступили в силу.

Использование OAuth Proxy с ALB

Содержание

Overview

В этом документе показано, как использовать OAuth Proxy с ALB для реализации внешней аутентификации.

Procedure

Выполните следующие шаги для использования данной функции:

1. Разверните kind

```
kind create cluster --name alb-auth --image=kindest/node:v1.28.0
kind get kubeconfig --name=alb-auth > ~/.kube/config
```

2. Разверните alb

```
helm repo add alb https://alauda.github.io/alb/;helm repo update;helm search repo|grep
alb
helm install alb-operator alb/alauda-alb2
alb_ip=$(docker inspect -f '{{range.NetworkSettings.Networks}}{{.IPAddress}}{{end}}'
alb-auth-control-plane)
echo $alb_ip
cat <<EOF | kubectl apply -f -
apiVersion: crd.alauda.io/v2
kind: ALB2
metadata:
    name: alb-auth
spec:
    address: "$alb_ip"
    type: "nginx"
    config:
        networkMode: host
        loadbalancerName: alb-demo
        projects:
        - ALL_ALL
        replicas: 1
EOF
```

3. Разверните тестовое приложение

- Создайте github oauth app ↗

> Обратите внимание, что `$GITHUB_CLIENT_ID` и `$GITHUB_CLIENT_SECRET` будут получены на этом шаге и должны быть установлены в переменные окружения

- Настройте DNS

> Здесь мы используем echo.com в качестве домена приложения, auth.alb.echo.com и alb.echo.com

- Разверните oauth-proxy

> oauth2-proxy должен иметь доступ к github, возможно потребуется установка переменной окружения HTTPS_PROXY

```
COOKIE_SECRET=$(python -c 'import os,base64;
print(base64.urlsafe_b64encode(os.urandom(32)).decode())')
OAUTH2_PROXY_IMAGE="quay.io/oauth2-proxy/oauth2-proxy:v7.7.1"
kind load docker-image $OAUTH2_PROXY_IMAGE --name alb-auth
cat <<EOF | kubectl apply -f -
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    k8s-app: oauth2-proxy
  name: oauth2-proxy
spec:
  replicas: 1
  selector:
    matchLabels:
      k8s-app: oauth2-proxy
  template:
    metadata:
      labels:
        k8s-app: oauth2-proxy
    spec:
      containers:
        - args:
            - --http-address=0.0.0.0:4180
            - --redirect-url=http://auth.alb.echo.com/oauth2/callback
            - --provider=github
            - --whitelist-domain=.alb.echo.com
            - --email-domain=*
            - --upstream=file:///dev/null
            - --cookie-domain=.alb.echo.com
            - --cookie-secure=false
            - --reverse-proxy=true
          env:
            - name: OAUTH2_PROXY_CLIENT_ID
              value: $GITHUB_CLIENT_ID
            - name: OAUTH2_PROXY_CLIENT_SECRET
              value: $GITHUB_CLIENT_SECRET
            - name: OAUTH2_PROXY_COOKIE_SECRET
              value: $COOKIE_SECRET
          image: $OAUTH2_PROXY_IMAGE
          imagePullPolicy: IfNotPresent
          name: oauth2-proxy
          ports:
```

```
            - containerPort: 4180
              name: http
              protocol: TCP
            - containerPort: 44180
              name: metrics
              protocol: TCP
---
apiVersion: v1
kind: Service
metadata:
  labels:
    k8s-app: oauth2-proxy
  name: oauth2-proxy
spec:
 ports:
 - appProtocol: http
   name: http
   port: 80
   protocol: TCP
   targetPort: http
 - appProtocol: http
   name: metrics
   port: 44180
   protocol: TCP
   targetPort: metrics
 selector:
   k8s-app: oauth2-proxy
EOF
```

4. Настройте ingress

   Мы настроим два ingress: auth.alb.echo.com и alb.echo.com

```
cat <<EOF | kubectl apply -f -
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/auth-url: "https://auth.alb.echo.com/oauth2/auth"
    nginx.ingress.kubernetes.io/auth-signin: "https://auth.alb.echo.com/oauth2/start?
rd=http://\$host\$request_uri"
  name: echo-resty
spec:
  ingressClassName: alb-auth
  rules:
    - host: alb.echo.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: echo-resty
                port:
                  number: 80
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: oauth2-proxy
spec:
  ingressClassName: alb-auth
  rules:
    - host: auth.alb.echo.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: oauth2-proxy
                port:
                  number: 80
EOF
```

Result

- После завершения операций будут развернуты alb, oauth-proxy и тестовое приложение.
- При обращении к alb.echo.com вы будете перенаправлены на страницу аутентификации github, после проверки вы увидите вывод приложения.

Создание GatewayAPI Gateway

GatewayAPI — это новый API для Kubernetes, который предоставляет более гибкий и расширяемый способ управления входящим трафиком. Он позволяет определять правила маршрутизации, политики трафика и другие настройки в более декларативной форме.
В этом документе представлен пошаговый гид по созданию GatewayAPI gateway в кластере Kubernetes платформы Alauda Container Platform.

Требования

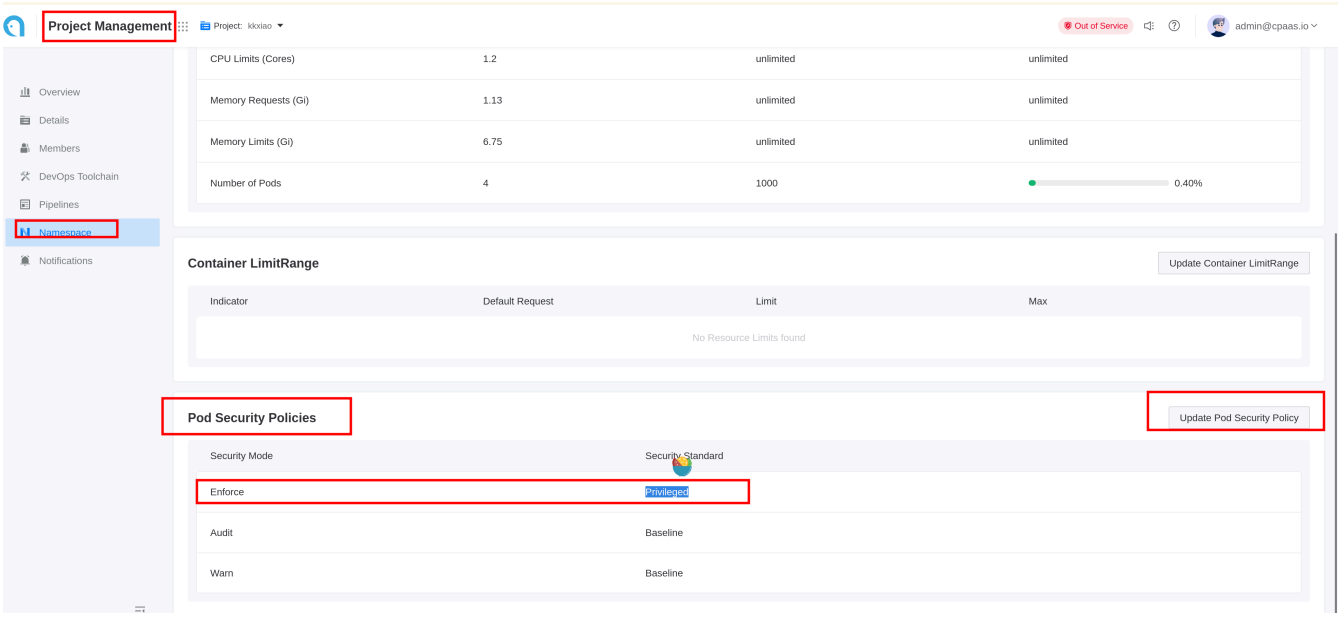
Содержание

Развертывание MetalLB

Для работы GatewayAPI gateway требуется MetalLB для выделения IP-адреса. Пожалуйста, ознакомьтесь с инструкцией Create MetalLB по развертыванию MetalLB.

Установка Pod Security Policies в режим Privileged

Если namespace, в котором вы хотите развернуть gateway, создан через UI, необходимо обновить его Pod Security Policy (PSP) до режима privileged.



Процедура

1. Перейдите в **Platform Management**.

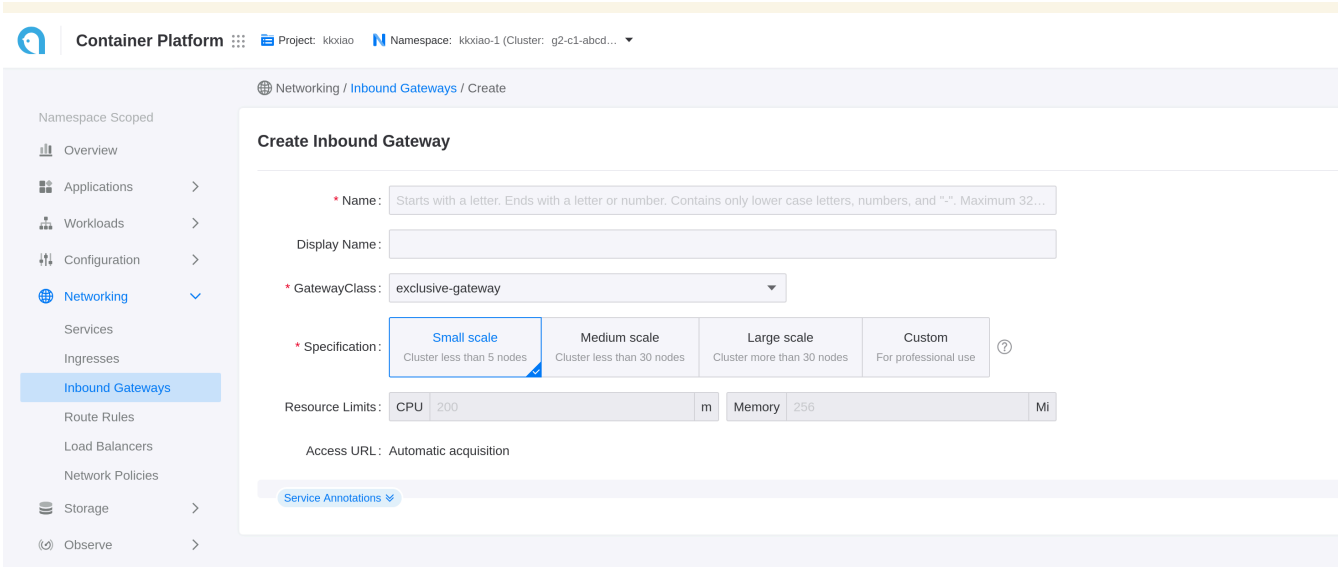
2. В левой боковой панели нажмите **Network Management** > **Inbound Gateways**.

3. Нажмите **Create Inbound Gateways**.

4. Следуйте инструкциям ниже для завершения настройки сети:

Параметр	Описание
Name	Имя gateway.
GatewayClass	Встроенный `exclusive-gateway` предоставляется платформой Alauda Container Platform и поддерживается ALB. Он создаст ALB в режиме container-network-mode для реализации спецификации GatewayAPI gateway.
Specification	Установите параметры в соответствии с потребностями вашего бизнеса. Также можно обратиться к руководству How to properly allocate CPU and memory resources для рекомендаций.

5. Нажмите **Create**. Процесс создания может занять некоторое время, пожалуйста, наберитесь терпения.


Container Platform
Project: kkxiao
Namespace: kkxiao-1 (Cluster: g2-c1-abcd...
Networking / Inbound Gateways / Create
Namespace Scoped
Overview
Applications
Workloads
Configuration
Networking
Services
Ingresses
Inbound Gateways
Route Rules
Load Balancers
Network Policies
Storage
Observe
Create Inbound Gateway
* Name:
Starts with a letter. Ends with a letter or number. Contains only lower case letters, numbers, and "-". Maximum 32...
Display Name:
* GatewayClass:
exclusive-gateway
* Specification:
Small scale
Cluster less than 5 nodes
Medium scale
Cluster less than 30 nodes
Large scale
Cluster more than 30 nodes
Custom
For professional use
Resource Limits:
CPU
200
m
Memory
256
Mi
Access URL:
Automatic acquisition
Service Annotations

Настройка балансировщика нагрузки

Балансировщик нагрузки — это сервис, который распределяет трафик по контейнерным инстансам. Используя функциональность балансировки нагрузки, он автоматически распределяет входящий трафик для вычислительных компонентов и перенаправляет его на контейнерные инстансы этих компонентов. Балансировка нагрузки может повысить отказоустойчивость вычислительных компонентов, масштабировать внешнюю сервисную способность этих компонентов и улучшить доступность приложений.

Администраторы платформы могут создавать одноточечные или высокодоступные балансировщики нагрузки для любого кластера на платформе, а также централизованно управлять и распределять ресурсы балансировщиков нагрузки. Например, балансировка нагрузки может быть назначена проектам, обеспечивая использование балансировки только пользователями с соответствующими правами в проекте.

Пожалуйста, обратитесь к таблице ниже для объяснения связанных понятий в этом разделе.

Параметр	Описание
Load Balancer	Программное или аппаратное устройство, которое распределяет сетевые запросы между доступными узлами в кластере. Используемый на платформе балансировщик нагрузки — это программный балансировщик уровня 7.
VIP	Виртуальный IP-адрес (Virtual IP Address) — IP-адрес, который не соответствует конкретному компьютеру или конкретной сетевой карте. При использовании балансировщика нагрузки типа высокодоступности адрес доступа должен быть VIP.

Содержание

Предварительные требования

Высокая доступность **Load Balancer** требует наличия VIP. Пожалуйста, обратитесь к Настройка VIP.

Пример ресурса ALB2 (CR)

```yaml
# test-alb.yaml
apiVersion: crd.alauda.io/v2beta1
kind: ALB2
metadata:
  name: alb-demo
  namespace: cpaas-system
  annotations:
    cpaas.io/display-name: ""
spec:
  address: 192.168.66.215
  config:
    vip: (1)
      enableLbSvc: false
      lbSvcAnnotations: {}
    networkMode: host (2)
    enablePortProject: false (3)
    nodeSelector:
      cpu-model.node.kubevirt.io/Nehalem: "true"
    projects: (4)
      - ALL_ALL
    replicas: 1
    resources: (5)
      limits:
        cpu: 200m
        memory: 256Mi
      requests:
        cpu: 200m
        memory: 256Mi
  type: nginx
```

(1) При `enableLbSvc` равном true будет создан внутренний сервис типа LoadBalancer для адреса доступа балансировщика нагрузки. `lbSvcAnnotations` — ссылка на конфигурацию аннотаций сервиса типа LoadBalancer.

(2) Ознакомьтесь с конфигурацией Network Mode ниже.

(3) Ознакомьтесь с методом распределения ресурсов ниже.

(4) Ознакомьтесь с назначенным проектом ниже.

(5) Ознакомьтесь с техническими характеристиками ниже.

Создание балансировщика нагрузки через веб-консоль

1. Перейдите в **Platform Management**.
2. В левой боковой панели нажмите **Network Management** > **Load Balancer**.
3. Нажмите **Create Load Balancer**.
4. Следуйте инструкциям ниже для завершения настройки сети.

Параметр	Описание
Network Mode	• **Host Network Mode**: На одном узле разрешается развертывать только один реплика балансировщика нагрузки, при этом несколько сервисов используют один ALB, что обеспечивает лучшую производительность сети. • **Container Network Mode**: На одном узле можно развернуть несколько реплик балансировщика нагрузки, чтобы удовлетворить требования отдельных ALB для каждого сервиса, с немного меньшей производительностью сети.
Service and Annotations (Alpha)	• **Service**: При включении создаётся внутренний сервис типа LoadBalancer для адреса доступа балансировщика нагрузки. Перед использованием убедитесь, что текущий кластер поддерживает сервис типа LoadBalancer. Можно использовать встроенный сервис типа LoadBalancer платформы; при отключении необходимо настроить External Address Pool для балансировщика нагрузки. • **Annotations**: Используются для объявления конфигурации или возможностей внутреннего маршрутизации типа LoadBalancer; подробности см. в Annotations for Internal LoadBalancer Type Routing.

Параметр	Описание
Access Address	Адрес доступа для балансировки нагрузки, то есть адрес сервиса инстанса балансировщика нагрузки. После успешного создания балансировщика к нему можно обращаться по этому адресу. • В режиме host network mode заполните согласно реальным условиям; это может быть доменное имя или IP-адрес (внутренний IP, внешний IP, VIP). • В режиме container network mode адрес будет получен автоматически.

5. Следуйте инструкциям ниже для завершения настройки ресурсов.

Параметр	Описание
Specification	Установите характеристики разумно в соответствии с бизнес-требованиями. Также можно обратиться к Как правильно распределять ресурсы CPU и памяти для справки.
Deployment Type	• **Single Point**: Контейнерная группа балансировщика нагрузки развёрнута на одном узле, что может привести к недоступности балансировщика при сбое машины. • **High Availability**: Несколько контейнерных групп балансировщика нагрузки развёрнуты на соответствующем количестве узлов, обычно 3. Это удовлетворяет потребности балансировки при больших объёмах бизнеса и обеспечивает возможности аварийного восстановления.
Replicas	Количество реплик — это количество контейнерных групп балансировщика нагрузки. **Совет**: Для обеспечения высокой доступности балансировщика рекомендуется количество реплик не менее 3.

Параметр	Описание
Node Labels	Фильтрация узлов по меткам для развертывания балансировщика нагрузки. **Совет**: • Рекомендуется, чтобы количество узлов, удовлетворяющих требованиям, было больше количества реплик балансировщика. • Метка с одинаковым ключом может выбрать только одну (если выбрано несколько, подходящих хостов не будет).
Resource Allocation Method	• **Instance**: Любой порт в диапазоне 1-65535, на котором может слушать инстанс балансировщика, может быть предоставлен для использования проектом. • **Port (Alpha)**: Можно выделять только порты в указанном диапазоне для использования проектом. Этот метод позволяет более тонко контролировать ресурсы при ограниченности портов.

Assigned Project	• При установке **Resource Allocation Method** в **Instance** балансировщик может быть выделен всем проектам, связанным с текущим кластером, или указанным проектам. Во всех выделенных проектах все Pod во всех пространствах имён могут получать запросы, распределяемые балансировщиком.
	  • **Все проекты**: Выделяет балансировщик для использования всеми проектами, связанными с текущим кластером.
	  • **Указанные проекты (Alpha)**: Нажмите выпадающий список под **Specified Projects** и отметьте чекбоксы слева от названий проектов для выбора одного или нескольких проектов, выделяя балансировщик для использования этими проектами.

Параметр	Описание
	Совет: Можно фильтровать проекты, вводя их имена в выпадающем списке. • **Без выделения (Alpha)**: Временно не выделяет ни один проект. После создания балансировщика можно использовать операцию **Update Project** для обновления параметров выделения проекта. • При установке **Resource Allocation Method** в **Port** этот пункт не нужно настраивать. Порты выделяются вручную после создания балансировщика.

6. Нажмите **Create**. Процесс создания займет некоторое время, пожалуйста, подождите.

Создание балансировщика нагрузки через CLI

```
kubectl apply -f test-alb.yaml -n cpaas-system
```

Обновление балансировщика нагрузки через веб-консоль

NOTE

Обновление балансировщика вызовет прерывание сервиса на 3–5 минут. Пожалуйста, выбирайте подходящее время для этой операции!

1. Перейдите в **Platform Management**.

2. В левой навигационной панели нажмите **Network Management** > **Load Balancer**.

3. Нажмите ⋮ > **Update**.

4. Обновите сетевые и ресурсные настройки по необходимости.

   - Устанавливайте характеристики разумно в соответствии с бизнес-требованиями. Также можно обратиться к Как правильно распределять ресурсы CPU и памяти для справки.

   - **Внутренняя маршрутизация** поддерживает обновление только из состояния **Disabled** в состояние **Enabled**.

5. Нажмите **Update**.

Удаление балансировщика нагрузки через веб-консоль

> **NOTE**
>
> После удаления балансировщика будут удалены связанные порты и правила, восстановить их будет невозможно.

1. Перейдите в **Platform Management**.

2. В левой навигационной панели нажмите **Network Management** > **Load Balancer**.

3. Нажмите ⋮ > **Delete** и подтвердите.

Удаление балансировщика нагрузки через CLI

```
kubectl delete alb2 test-alb -n cpaas-system
```

Настройка портов слушателя (Frontend)

Балансировщик нагрузки поддерживает приём клиентских соединений через порты слушателя и соответствующие протоколы, включая HTTPS, HTTP, gRPC, TCP и UDP.

Предварительные требования

Если необходимо добавить HTTPS-порт слушателя, следует также обратиться к администратору для назначения TLS-сертификата текущему проекту для шифрования.

Пример ресурса Frontend (CR)

```yaml
# alb-frontend-demo.yaml
apiVersion: crd.alauda.io/v1
kind: Frontend
metadata:
  labels:
    alb2.cpaas.io/name: alb-demo (1)
  name: alb-demo-00080 (2)
  namespace: cpaas-system
spec:
  backendProtocol: "http"
  certificate_name: "" (3)
  port: 80
  protocol: http (4)
  serviceGroup: (5)
    services:
      - name: hello-world
        namespace: default
        port: 80
        weight: 100 (6)
```

1. Обязательно, указывает ALB-инстанс, к которому принадлежит этот `Frontend`.
2. Формат `$alb_name-$port`.
3. Формат `$secret_ns/$secret_name`.
4. Протокол самого `Frontend`.
   - `http|https|grpc|grpcs` для прокси уровня 7.
   - `tcp|udp` для прокси уровня 4.
5. Для прокси уровня 4 `serviceGroup` обязателен. Для уровня 7 `serviceGroup` опционален. При поступлении запроса ALB сначала пытается сопоставить его с правилами, связанными с этим `Frontend`. Если запрос не соответствует ни одному правилу, ALB перенаправляет его в дефолтный `serviceGroup`, указанный в конфигурации `Frontend`.
6. Конфигурация `weight` применяется для алгоритмов планирования Round Robin и Weighted Round Robin.

> **NOTE**
>
> ALB слушает ingress и автоматически создаёт `Frontend` или Rule. Поле `source` определяется следующим образом:
>
> 1. `spec.source.type` в настоящее время поддерживает только `ingress`.
> 2. `spec.source.name` — имя ingress.
> 3. `spec.source.namespace` — пространство имён ingress.

Создание портов слушателя (Frontend) через веб-консоль

1. Перейдите в **Container Platform**.
2. В левой навигационной панели нажмите **Network** > **Load Balancing**.
3. Нажмите на имя балансировщика нагрузки для перехода на страницу деталей.

4. Нажмите **Add Listener Port**.

5. Ознакомьтесь с инструкциями ниже для настройки параметров.

Параметр	Описание
Protocol	Поддерживаемые протоколы: HTTPS, HTTP, gRPC, TCP и UDP. При выборе HTTPS необходимо добавить сертификат; для gRPC добавление сертификата опционально. **Примечание**: • При выборе протокола gRPC протокол бэкенда по умолчанию — gRPC, при этом не поддерживается сессия. • Если для gRPC задан сертификат, балансировщик снимет сертификат gRPC и перенаправит незашифрованный трафик gRPC на бэкенд. • При использовании кластера Google GKE балансировщик одного **типа контейнерной сети** не может одновременно иметь протоколы слушателя TCP и UDP.
Internal Routing Group	- При выборе алгоритма балансировки **Round Robin (RR)** трафик распределяется по портам внутренней маршрутизации в порядке группы маршрутов. - При выборе алгоритма **Weighted Round Robin (WRR)** внутренние маршруты с большим значением веса имеют большую вероятность выбора; трафик распределяется по портам внутренней маршрутизации согласно рассчитанной вероятности на основе настроенного **weight**. **Совет**: Вероятность рассчитывается как отношение текущего значения веса к сумме всех весов.
Session Persistence	Всегда перенаправляет определённые запросы на бэкенд-сервисы, соответствующие вышеуказанной внутренней группе маршрутов. Определённые запросы включают (выберите один):

Параметр	Описание
	• Хэш исходного адреса: все запросы с одного IP-адреса. **Примечание**: В публичных облаках исходный адрес часто меняется, что может привести к тому, что запросы от одного клиента будут иметь разные IP-адреса в разное время, и метод хэширования по адресу не даст ожидаемого результата. • Ключ cookie: запросы, содержащие указанный cookie. • Имя заголовка: запросы, содержащие указанный заголовок.
Backend Protocol	Протокол, используемый для передачи трафика на бэкенд-сервисы. Например, при передаче на Kubernetes или dex-сервисы бэкенда следует выбрать протокол HTTPS.

6. Нажмите **OK**.

Создание портов слушателя (Frontend) через CLI

```
kubectl apply -f alb-frontend-demo.yaml -n cpaas-system
```

Последующие действия

Для трафика с портов HTTP, gRPC и HTTPS, помимо дефолтной внутренней группы маршрутов, можно настроить более разнообразные правила сопоставления бэкенд-сервисов rules. Балансировщик сначала пытается сопоставить соответствующий бэкенд согласно установленным правилам; если совпадений нет, то направляет трафик на бэкенд из указанной внутренней группы маршрутов.

Связанные операции

Вы можете нажать значок ⋮ справа на странице списка или кнопку **Actions** в правом верхнем углу страницы деталей, чтобы при необходимости обновить дефолтный маршрут или удалить порт слушателя.

> **NOTE**
>
> Если метод распределения ресурсов балансировщика — **Port**, удалять связанные порты слушателя в представлении **Platform Management** могут только администраторы.

Настройка правил

Добавление правил переадресации для портов слушателя протоколов HTTPS, HTTP и gRPC. Балансировщик будет сопоставлять бэкенд-сервисы на основе этих правил.

> **NOTE**
>
> Правила переадресации нельзя добавлять для протоколов TCP и UDP.

Пример ресурса Rule (CR)

```yaml
# alb-rule-demo.yaml
apiVersion: crd.alauda.io/v1
kind: Rule
metadata:
  labels:
    alb2.cpaas.io/frontend: alb-demo-00080 (1)
    alb2.cpaas.io/name: alb-demo (2)
  name: alb-demo-00080-test
  namespace: cpaas-system
spec:
  backendProtocol: "" (3)
  certificate_name: "" (4)
  dslx:
    - type: METHOD
      values:
        - - EQ
          - POST
    - type: URL
      values:
        - - STARTS_WITH
          - /app-a
        - - STARTS_WITH
          - /app-b
    - type: PARAM
      key: group
      values:
        - - EQ
          - vip
    - type: HOST
      values:
        - - ENDS_WITH
          - .app.com
    - type: HEADER
      key: LOCATION
      values:
        - - IN
          - east-1
          - east-2
    - type: COOKIE
      key: uid
      values:
        - - EXIST
    - type: SRC_IP
```

```
      values:
        - - RANGE
          - "1.1.1.1"
          - "1.1.1.100"
  enableCORS: false
  priority: 4 5
  serviceGroup: 6
    services:
      - name: hello-world
        namespace: default
        port: 80
        weight: 100
```

1. Обязательно, указывает `Frontend`, к которому принадлежит это правило.
2. Обязательно, указывает ALB, к которому принадлежит это правило.
3. Аналогично `Frontend`.
4. Аналогично `Frontend`.
5. Чем меньше число, тем выше приоритет.
6. Аналогично `Frontend`.

dslx

dslx — это предметно-ориентированный язык, используемый для описания критериев сопоставления.

Например, следующее правило соответствует запросу, который удовлетворяет всем нижеперечисленным условиям:

- URL начинается с /app-a или /app-b
- метод POST
- параметр URL с ключом group равен vip
- хост соответствует *.app.com
- заголовок LOCATION равен east-1 или east-2
- присутствует cookie с именем uid
- исходные IP-адреса находятся в диапазоне 1.1.1.1-1.1.1.100

```
dslx:
  - type: METHOD
    values:
      - - EQ
        - POST
  - type: URL
    values:
      - - STARTS_WITH
        - /app-a
      - - STARTS_WITH
        - /app-b
  - type: PARAM
    key: group
    values:
      - - EQ
        - vip
  - type: HOST
    values:
      - - ENDS_WITH
        - .app.com
  - type: HEADER
    key: LOCATION
    values:
      - - IN
        - east-1
        - east-2
  - type: COOKIE
    key: uid
    values:
      - - EXIST
  - type: SRC_IP
    values:
      - - RANGE
        - "1.1.1.1"
        - "1.1.1.100"
```

Создание правила через веб-консоль

1. Перейдите в **Container Platform**.

2. В левой навигационной панели нажмите **Network** > **Load Balancing**.

3. Нажмите на имя балансировщика нагрузки.

4. Нажмите на имя порта слушателя.

5. Нажмите **Add Rule**.

6. Ознакомьтесь с описаниями ниже для настройки параметров.

Параметр	Описание
Internal Route Group	- При выборе алгоритма балансировки **Round Robin (RR)** трафик распределяется по портам внутренней маршрутизации в порядке группы маршрутов. - При выборе алгоритма **Weighted Round Robin (WRR)** внутренние маршруты с большим значением веса имеют большую вероятность выбора; трафик распределяется по портам внутренней маршрутизации согласно рассчитанной вероятности на основе настроенного **weight**. **Совет**: Вероятность рассчитывается как отношение текущего значения веса к сумме всех весов.
Rule	Критерии сопоставления балансировщика с бэкенд-сервисами, включая индикаторы правил и их значения. Отношение между разными индикаторами — «и». • Доменное имя: поддерживается добавление **wildcard-доменов** и **точных доменных имён**. При равенстве приоритетов для одного правила, если существуют и wildcard, и точные доменные правила, приоритет имеет точное доменное правило. • URL: **RegEx** — регулярные выражения URL, начинающиеся с `/`; **StartsWith** — префиксы URL, начинающиеся с `/`. • IP: **Equal** — конкретный IP-адрес; **Range** — диапазон IP-адресов. • Заголовок: кроме ключа заголовка, необходимо задать правило сопоставления. **Equal** — конкретное значение

Параметр	Описание
	заголовка; **Range** — диапазон значений заголовка; **RegEx** — регулярное выражение заголовка. • Cookie: кроме ключа cookie, необходимо задать правило сопоставления. **Equal** — конкретное значение cookie. • URL Param: в правилах сопоставления **Equal** — конкретный параметр URL; **Range** — диапазон параметров URL. • Service Name: имя сервиса, использующего протокол gRPC. При использовании gRPC можно настроить этот параметр, чтобы трафик перенаправлялся на соответствующий сервис по имени, например: `/helloworld.Greeter`.
Session Persistence	Всегда перенаправляет определённые запросы на бэкенд-сервисы, соответствующие внутренней группе маршрутов. Определённые запросы включают (выберите один): • Хэш исходного адреса: все запросы с одного IP-адреса. • Ключ cookie: запросы, содержащие указанный cookie. • Имя заголовка: запросы, содержащие указанный заголовок.
URL Rewrite	Перезаписывает обращаемый адрес на адрес бэкенд-сервиса платформы. Для работы требуется настройка правила URL с индикатором **StartsWith**, а адрес перезаписи (rewrite-target) должен начинаться с /. Например: при настройке домена *bar.example.com* и начального пути URL `/`, включении функции **URL Rewrite** и установке адреса перезаписи в */test* обращение к *bar.example.com* будет переписано в *bar.example.com/test*.
Backend Protocol	Протокол, используемый для передачи трафика на бэкенд-сервис. Например, при передаче на Kubernetes или dex-сервисы бэкенда следует выбрать протокол HTTPS.

Параметр	Описание
Redirection	Перенаправляет трафик на новый адрес, а не на бэкенд-сервисы внутренней группы маршрутов. Например: при обновлении страницы по исходному адресу, чтобы избежать ошибок 404 или 503, трафик можно перенаправить на новый адрес. • HTTP Status Code: код состояния, который браузер покажет пользователю перед перенаправлением. • Redirect Address: при вводе относительного адреса (например, */index.html*) трафик будет направлен на *адрес балансировщика/index.html*; при вводе абсолютного адреса (например, *https://www.example.com* ↗) трафик будет направлен на указанный адрес.
Rule Priority	Приоритет сопоставления правил: 10 уровней от 1 до 10, где 1 — самый высокий приоритет, по умолчанию 5. Если одновременно удовлетворяются несколько правил, выбирается правило с более высоким приоритетом; при равенстве приоритетов применяется правило по умолчанию.
Cross-Origin Resource Sharing (CORS)	CORS (Cross-origin resource sharing) — механизм, использующий дополнительные HTTP-заголовки, чтобы указать браузеру, что веб-приложение с одного источника (домена) может получить доступ к ресурсам с другого источника. При запросе ресурса с сервера другого домена, протокола или порта инициируется кросс-доменный HTTP-запрос.
Allowed Origins	Указывает разрешённые источники доступа. • *****: разрешает запросы с любого источника. • **Доменное имя**: разрешает запросы с указанного домена.
Allowed Headers	Указывает HTTP-заголовки, разрешённые в CORS, чтобы избежать лишних предварительных запросов и повысить эффективность. Примеры:

Параметр	Описание
	Примечание: Другие часто используемые или кастомные заголовки не перечислены; заполняйте по необходимости. • **Origin**: указывает источник запроса, то есть домен, отправляющий запрос. • **Authorization**: указывает данные авторизации, например Basic Authentication или Token. • **Content-Type**: указывает тип содержимого запроса/ответа, например application/json, application/x-www-form-urlencoded и др. • **Accept**: указывает типы содержимого, которые клиент может принимать, обычно для получения определённого типа ответа.

7. Нажмите **Add**.

Создание правила через CLI

```
kubectl apply -f alb-rule-demo.yaml -n cpaas-system
```

Логи и мониторинг

Сочетая визуализированные логи и данные мониторинга, можно быстро выявлять и устранять проблемы или сбои балансировщика нагрузки.

Просмотр логов

1. Перейдите в **Platform Management**.
2. В левой навигационной панели нажмите **Network Management** > **Load Balancer**.
3. Нажмите на ***Load Balancer Name***.
4. Во вкладке **Logs** просмотрите логи работы балансировщика с точки зрения контейнера.

Метрики мониторинга

> **NOTE**
>
> В кластере, где расположен балансировщик, должны быть развернуты сервисы мониторинга.

1. Перейдите в **Platform Management**.
2. В левой навигационной панели нажмите **Network Management** > **Load Balancer**.
3. Нажмите на ***Load Balancer Name***.
4. Во вкладке **Monitoring** просмотрите тренды метрик балансировщика с точки зрения узла.
   - **Usage Rate**: текущая загрузка CPU и памяти балансировщика на данном узле.
   - **Throughput**: общий входящий и исходящий трафик инстанса балансировщика.

Дополнительные ресурсы

- ALB Monitoring

Как правильно выделять ресурсы CPU и памяти

Для предложенных платформой спецификаций для **малых**, **средних**, **больших** и **пользовательских** производственных сред, а также методов выделения ресурсов для **инстансов** и **портов**, ниже приведены рекомендации для развертывания.

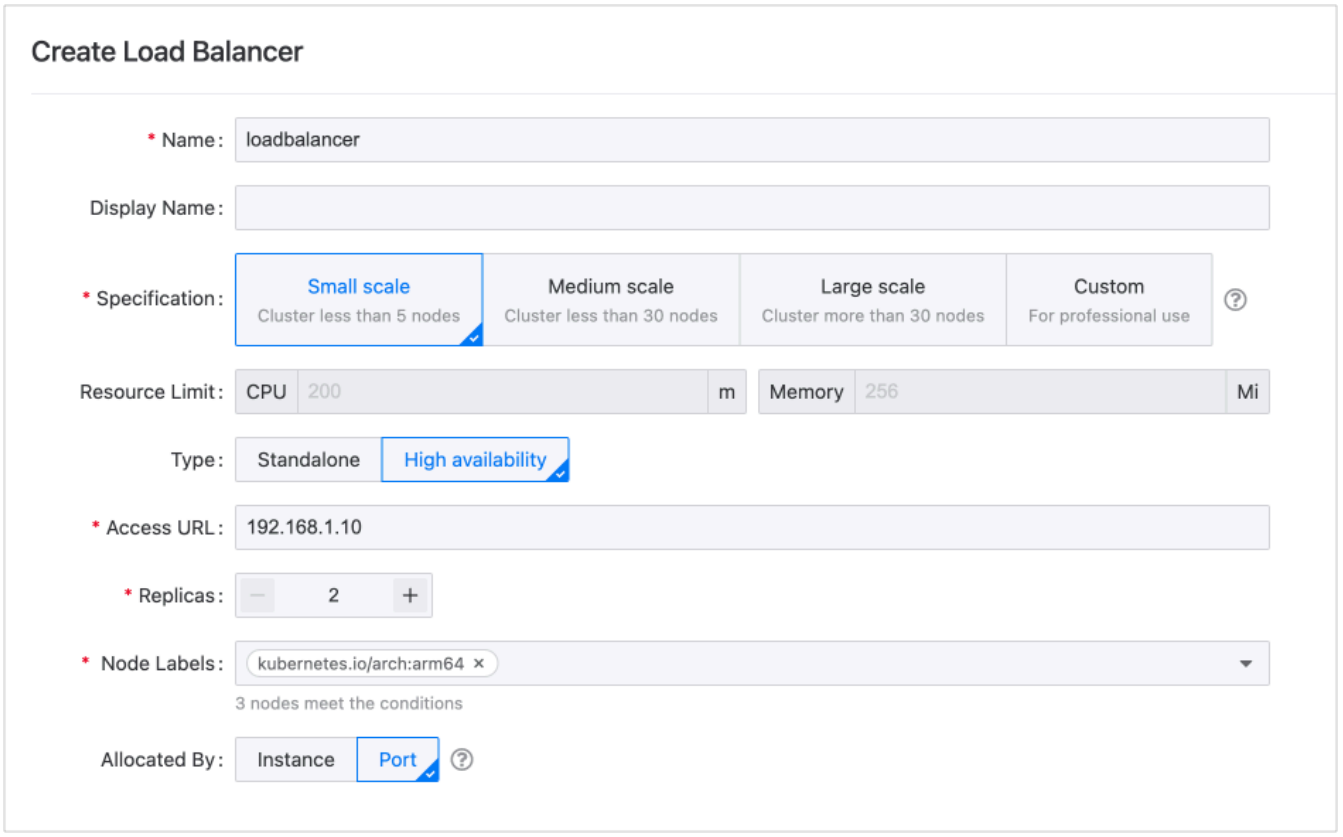
Содержание

Маленькая производственная среда

Для меньших масштабов бизнеса, например, при наличии не более 5 узлов в кластере и использовании только для запуска стандартных приложений, достаточно **одного** балансировщика нагрузки. Рекомендуется использовать его в режиме **высокой доступности** с минимум 2 репликами для обеспечения стабильности среды.

Вы можете изолировать балансировщик нагрузки с помощью **изоляции по портам**, что позволит нескольким проектам совместно его использовать.

Пиковый QPS, измеренный в лабораторных условиях для этой спецификации, составляет примерно 300 запросов в секунду.

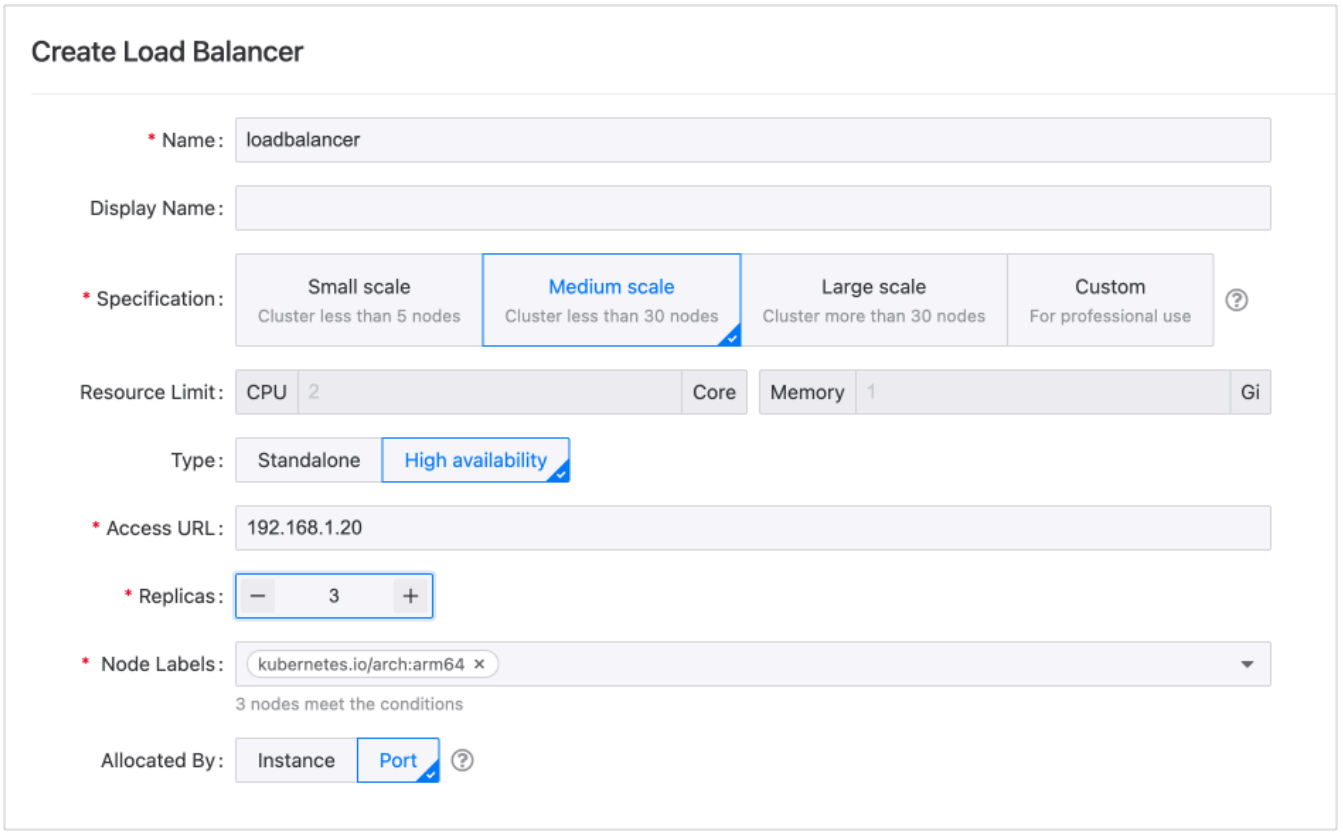


Средняя производственная среда

Когда объем бизнеса достигает определенного масштаба, например, при наличии не более 30 узлов в кластере и необходимости обработки высококонкурентного бизнеса наряду с запуском стандартных приложений, **одного** балансировщика нагрузки по-прежнему будет достаточно. Рекомендуется использовать режим **высокой доступности** с минимум 3 репликами для поддержания стабильности среды.

Вы можете использовать либо метод **изоляции по портам**, либо выделение по **инстансам** для совместного использования балансировщика нагрузки несколькими проектами. Конечно, вы также можете создавать новые балансировщики нагрузки для выделенного использования ключевыми проектами.

Пиковый QPS, измеренный в лабораторных условиях для этой спецификации, составляет около 10 000 запросов в секунду.

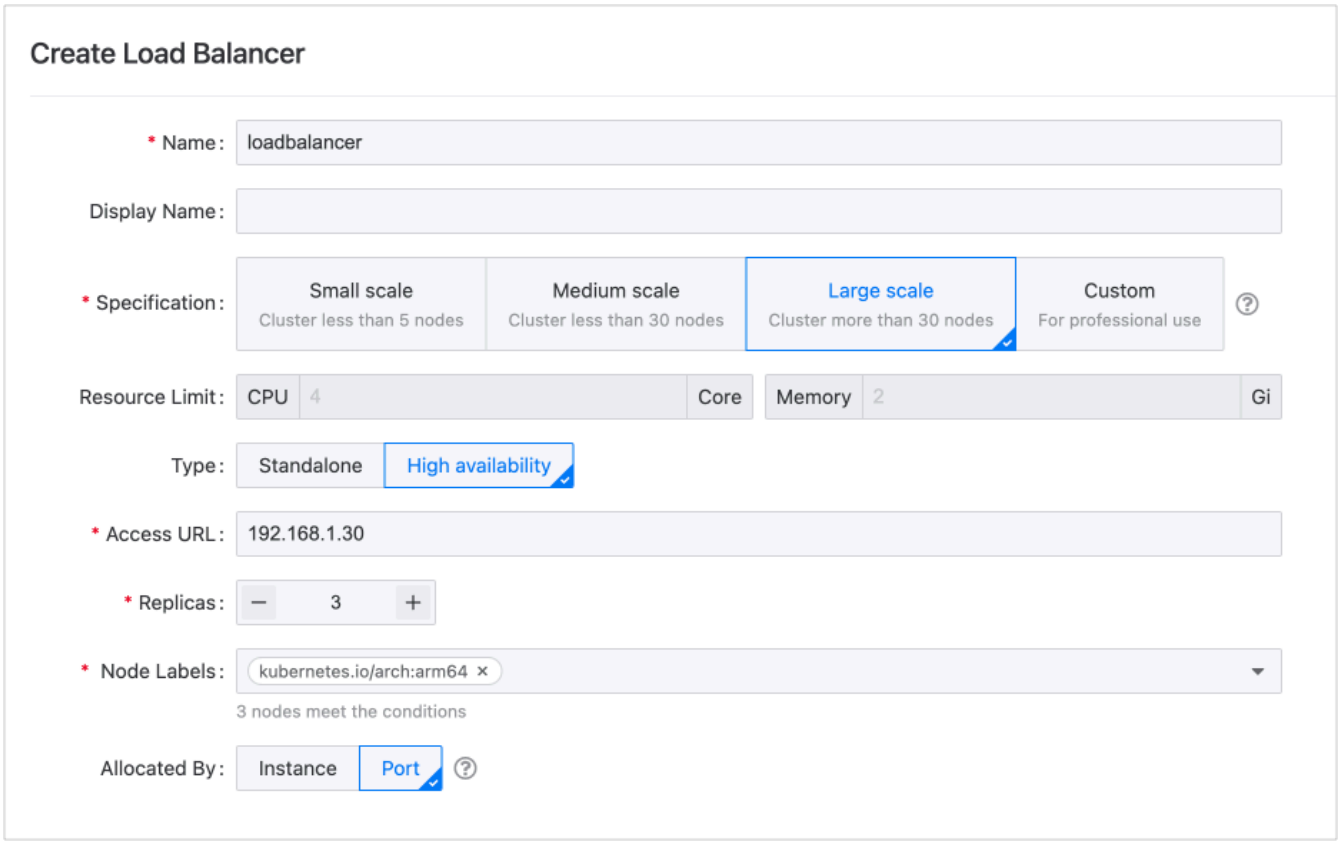


Большая производственная среда

Для больших объемов бизнеса, например, при наличии более 30 узлов в кластере и необходимости обработки высококонкурентного бизнеса, а также долгоживущих соединений с данными, рекомендуется использовать **несколько** балансировщиков нагрузки, каждый из которых работает в типе **высокой доступности** с минимум 3 репликами для обеспечения стабильности среды.

Вы можете изолировать балансировщик нагрузки с помощью либо метода **изоляции по портам**, либо выделения по **инстансам** для совместного использования несколькими проектами. Также можно создавать новые балансировщики нагрузки для эксклюзивного использования ключевыми проектами.

Пиковый QPS, измеренный в лабораторных условиях для этой спецификации, составляет примерно 20 000 запросов в секунду.



Рекомендации по развертыванию в особых сценариях

Сценарий	Рекомендации по развертыванию
Функциональное тестирование	Рекомендуется развернуть **один инстанс** балансировщика нагрузки.
Тестовая среда	Если тестовая среда соответствует определениям **малой** или **средней** среды, приведенным выше, достаточно использовать балансировщик нагрузки с **одной точкой**. Инстанс балансировщика нагрузки может быть разделен между **несколькими проектами**.
Ключевые приложения	Рекомендуется использовать отдельные балансировщики нагрузки исключительно для ключевых приложений.
Передача больших объемов данных	Из-за минимального потребления памяти самим балансировщиком нагрузки достаточно резервировать 2Gi

Сценарий	Рекомендации по развертыванию
	памяти даже для спецификации **большой**. Однако, если бизнес требует передачи больших объемов данных, что приведет к значительному потреблению памяти, выделение памяти для балансировщика нагрузки следует увеличить соответственно. Рекомендуется постепенно расширять память балансировщика нагрузки в сценариях со спецификацией **пользовательской**, внимательно отслеживая использование памяти, чтобы в итоге определить приемлемый размер памяти для разумных показателей использования.

Выбор режима использования балансировщика нагрузки

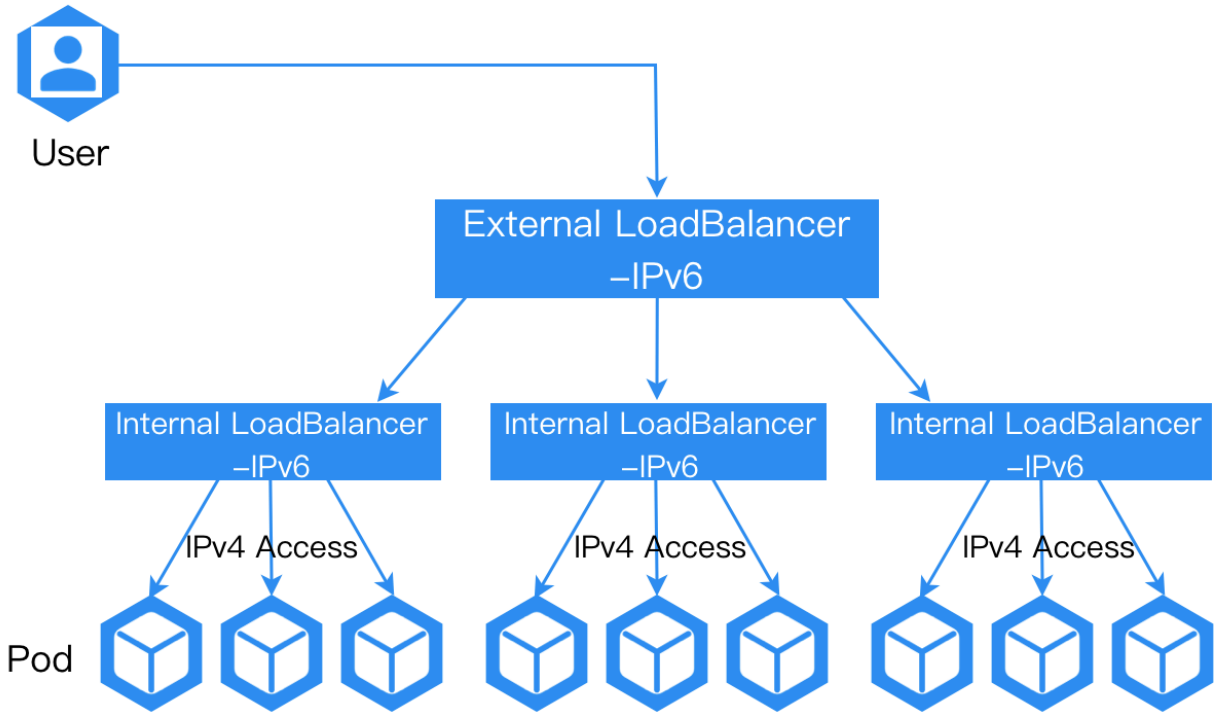
Режим использования	Преимущества	Недостатки
(Рекомендуется) Выделять балансировщик нагрузки как ресурс инстанса одному проекту	• Управление относительно простое. • Каждый проект имеет собственный балансировщик нагрузки, что обеспечивает изоляцию правил и разделение ресурсов без взаимных помех.	В режиме host network кластер должен иметь значительное количество узлов, доступных для балансировщика нагрузки, что приводит к высоким требованиям к потреблению ресурсов.

Режим использования	Преимущества	Недостатки
Выделять балансировщик нагрузки как ресурс инстанса нескольким проектам	Управление относительно простое.	Поскольку все назначенные проекты имеют полные права на инстанс балансировщика нагрузки, при конфигурировании портов и правил одним проектом могут возникнуть следующие ситуации: • Правила, настроенные этим проектом, могут повлиять на другие проекты. • Ошибочные действия при настройке балансировщика нагрузки могут изменить настройки других проектов. • Трафик от конкретного бизнеса может повлиять на общую доступность инстанса балансировщика нагрузки.
Динамическое выделение ресурсов балансировщика нагрузки по портам, с использованием разных портов разными проектами	Правила между проектами изолируют их, обеспечивая отсутствие взаимных помех.	• Управление усложняется. Администраторам платформы необходимо активно планировать и выделять порты для проектов и настраивать внешние сервисные отображения.

Режим использования	Преимущества	Недостатки
		• Зрелость выделения по портам ниже. В настоящее время используется меньшим числом клиентов и требует дальнейшего совершенствования функций. • Конфликты ресурсов. Все сервисы, использующие один балансировщик нагрузки, могут столкнуться с ситуациями, когда один сервис негативно влияет на весь балансировщик.

Проброс IPv6-трафика на IPv4-адреса внутри кластера

Настроив внешний балансировщик нагрузки для кластера, мы можем перенаправлять IPv6-трафик на внутренние IPv4-адреса внутри кластера. Это позволяет внедрить возможности IPv6 поверх существующей IPv4-сети, обеспечивая большую гибкость и масштабируемость архитектуры системы, а также лучшее удовлетворение разнообразных сетевых требований.



Содержание

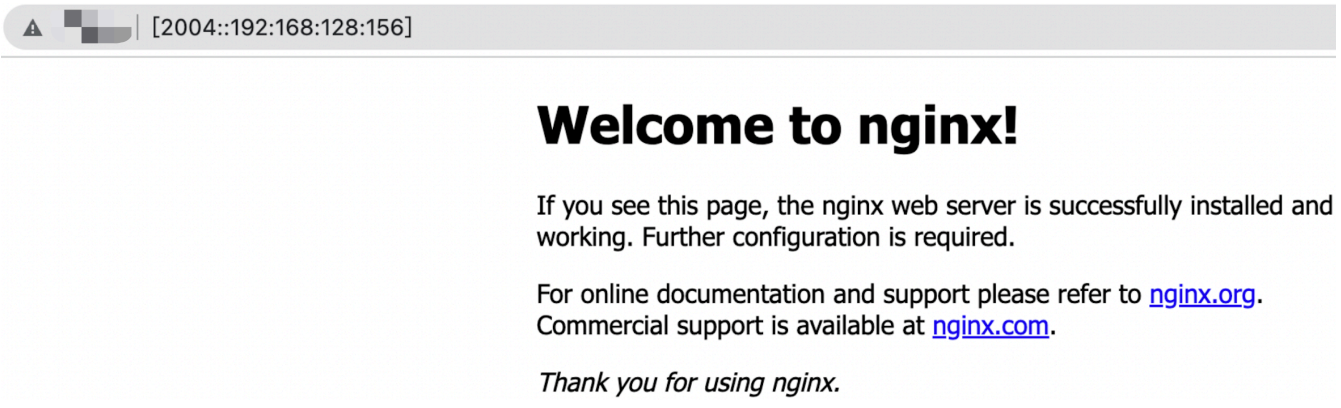
Метод настройки

1. Настройте IPv6-адрес для узла, на котором расположен балансировщик нагрузки.
2. Убедитесь, что у внешнего балансировщика нагрузки есть IPv6-адрес, и что трафик, обращающийся к IPv6-адресу балансировщика, может быть перенаправлен на IPv6-адрес узла, где находится балансировщик.

После выполнения вышеуказанной настройки IPv4-сервисы, размещённые на балансировщике нагрузки, смогут предоставлять внешние возможности доступа по IPv6 через балансировщик.

Проверка результата

После настройки доступ к IPv6-адресу внешнего балансировщика нагрузки должен обеспечивать нормальный доступ к приложению.



Calico Network поддерживает шифрование WireGuard

Calico поддерживает шифрование WireGuard как для IPv4, так и для IPv6 трафика, которое можно независимо включать с помощью параметров в ресурсе FelixConfiguration.

Содержание

Статус установки

Установка по умолчанию

Операционная система	Версия ядра
Linux	5.6 и выше установлены по умолчанию

Операционная система	Версия ядра
Ubuntu 20.04	5.4.0-135-generic
Kylin Linux Advanced Server V10 - SP3	4.19.90-52.22.v2207.ky10.x86_64

Не установлено по умолчанию

Операционная система	Версия ядра
openEuler	4.18.0-147.5.2.13.h996.eulerosv2r10.x86_64
CentOS 7	3.10.0-1160.el7.x86_64
Redhat 8.7	4.18.0-425.3.1.el8.x86_64
Kylin Linux Advanced Server V10 - SP2	4.19.90-24.4.v2101.ky10.x86_64
Kylin Linux Advanced Server V10 - SP1	4.19.90-23.8.v2101.ky10.x86_64
Kylin Linux Advanced Server V10	4.19.90-11.ky10.x86_64

Терминология

Термин	Объяснение
wireguardEnabled	Включить шифрование для IPv4 трафика поверх IPv4 Underlay сети.
wireguardEnabledV6	Включить шифрование для IPv6 трафика поверх IPv6 Underlay сети.

Примечания

1. При использовании сетевого плагина Calico убедитесь, что параметр `natOutgoing` установлен в `true` для поддержки шифрования WireGuard. По умолчанию этот параметр корректно настроен для подсети Calico при создании кластера и не требует дополнительной настройки.

2. WireGuard поддерживает шифрование как для IPv4, так и для IPv6 трафика; если необходимо шифровать оба типа трафика, настройка должна выполняться отдельно. Для подробной настройки параметров обратитесь к Felix Configuration Documentation ↗, настраивая параметры `wireguardEnabled` и `wireguardEnabledV6`.

3. Если WireGuard не установлен по умолчанию, обратитесь к WireGuard Installation Guide ↗ для ручной установки, хотя возможны случаи, когда ручная установка модуля WireGuard не удаётся.

4. Трафик между контейнерами на разных узлах будет зашифрован, включая сетевой трафик с одного хоста на другой; однако связь между Pod на одном узле и трафик между Pod и его хост-узлом не будет зашифрована.

Требования

- WireGuard должен быть установлен на всех узлах кластера заранее. Для подробностей обратитесь к WireGuard Installation Documentation ↗. Узлы без установленного WireGuard не поддерживают шифрование.

Процедура

1. Включите или отключите шифрование для IPv4 и IPv6.

   Примечание: Следующие команды необходимо выполнять в CLI-инструменте на Master-узле, где находится узел.

- Включить шифрование только для IPv4

```
kubectl patch felixconfiguration default --type='merge' -p '{"spec":
{"wireguardEnabled":true}}'
```

- Включить шифрование только для IPv6

```
kubectl patch felixconfiguration default --type='merge' -p '{"spec":
{"wireguardEnabledV6":true}}'
```

- Включить шифрование для IPv4 и IPv6

```
kubectl patch felixconfiguration default --type='merge' -p '{"spec":
{"wireguardEnabled":true,"wireguardEnabledV6":true}}'
```

- Отключить шифрование для IPv4 и IPv6
  - Способ 1: Выполнить команду в CLI для отключения шифрования.

```
kubectl patch felixconfiguration default --type='merge' -p '{"spec":
{"wireguardEnabled":false,"wireguardEnabledV6":false}}'
```

  - Способ 2: Изменить конфигурационный файл felixconfiguration для отключения шифрования.
    1. Выполните команду для открытия конфигурационного файла felixconfiguration.

```
kubectl get felixconfiguration -o yaml default
```

    2. Установите параметры `wireguardEnabled` и `wireguardEnabledV6` в false для отключения шифрования WireGuard.

```
apiVersion: crd.projectcalico.org/v1
kind: FelixConfiguration
metadata:
  annotations:
    projectcalico.org/metadata: '{"uid":"f5facabd-8304-46d6-81c1-
f1816235b487","creationTimestamp":"2024-08-06T03:46:51Z"}'
  generation: 2
  name: default
  resourceVersion: "890216"
spec:
  bpfLogLevel: ""
  floatingIPs: Disabled
  logSeverityScreen: Info
  reportingInterval: 0s
  wireguardEnabled: false # Измените на true для включения шифрования
IPv4
  wireguardEnabledV6: false # Измените на true для включения шифрования
IPv6
```

2. После завершения настройки шифрования Calico WireGuard выполните следующую команду для подтверждения статуса шифрования WireGuard. Если включено шифрование IPv4 и IPv6, наличие `wireguardPublicKey` или `wireguardPublicKeyV6` в поле `Status` указывает на успешное включение; если шифрование IPv4 и IPv6 отключено, эти поля не будут содержать `wireguardPublicKey` или `wireguardPublicKeyV6`, что означает успешное отключение.

   ```
   calicoctl get node <NODE-NAME> -o yaml # Замените <NODE-NAME> на имя узла.
   ```

   Вывод:

   ```
   Status:
       wireguardPublicKey: L/MUP9+Yxx/xxxxxxxxxxx/xxxxxxxxxx =
   ```

Проверка результата

В этом документе приведён пример проверки трафика IPv4; проверка трафика IPv6 аналогична IPv4 и не повторяется.

Проверка трафика IPv4

1. После настройки шифрования WireGuard проверьте информацию о маршрутизации, где трафик между узлами преимущественно использует интерфейс wireguard.cali для пересылки сообщений.

```
root@test:~# ip rule   # Просмотр текущих правил маршрутизации
     0: from all lookup local
     99:  not from all fwmark 0x100000/0x100000 lookup 1    # Для всех пакетов без метки 0x100000 используется таблица маршрутизации 1
     32766:  from all lookup main
     32767 :  from all lookup default

root@test:~# ip route show table 1    # Отобразить записи маршрутизации для таблицы 1.
    10.3.138.0 dev wireguard.cali scope link
    10.3.138.0/26 dev wireguard.cali scope link
    throw 10.3.231.192
    10.3.236.128 dev wireguard.cali scope link     # Трафик к IP 10.3.236.128 будет отправляться через интерфейс wireguard.cali
    10.3.236.128/26 dev wireguard.cali scope link
    throw 10.10.10.124/30
    10.10.10.200/30 dev wireguard.cali scope link
    throw 10.10.20.124/30
    10.10.20.200/30 dev wireguard.cali scope link
    throw
    10.13.138.0 dev wireguard.cali scope link
    10.13.138.0/26 dev wireguard.cali scope link
    throw 10.13.231.192/26
    10.13.236.128 dev wireguard.cali scope link
    10.13.236.128/26 dev wireguard.cali scope link

root@test:~# ip r get 10.10.10.202    # Маршрут от текущего узла до IP 10.10.10.202
    10.10.10.202 dev wireguard.cali table 1 src 10.10.10.127 uid 0  cache   # При доступе к IP 10.10.10.202 пакет будет отправлен через интерфейс wireguard.cali, используя таблицу маршрутизации 1, с исходным адресом 10.10.10.127

root@test:~# ip route    # Показать основную таблицу маршрутизации
    default via 192.168.128.1 dev eth0 proto static
    10.3.138.0/26 via 10.3.138.0 dev vxlan.
    blackhole 10.3.231.193
    10.3.231.194
    10.3.231.195
    10.3.231.196
    10.3.231.197
    3.231.192/26 proto 80
    dev cali8dcd31cIdOO scope link
    dev cali3012b5b29b scope link
    dev calibeefea2ff87 scope link
```

```
    dev cali2b27d5e4053 scope link
    dev cali1a35dbdd639 scope link
    calico on link
```

2. Захват пакетов на узле для наблюдения трафика между узлами.

```
root@test:~# ip a s wireguard.cali    # Просмотр подробной информации об интерфейсе wireguard.cali
    30: wireguard.cali: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1440 qdisc noqueue state UNKNOWN group default qlen 1000
    link/none
    inet 10.10.10.127/32 scope global wireguard.cali   # IP-адрес интерфейса wireguard.cali — 10.10.10.127
    valid_lft forever preferred_lft forever

root@test:~# tcpdump -i wireguard.cali -nnve icmp   # Захват и отображение ICMP пакетов через wireguard.cali
    tcpdump: listening on wireguard.cali, link-type RAW (Raw IP), capture size 262144 bytes
    08:58:36.987559 ip: (tos 0x0, ttl 63, id 29731, offset 0, flags [DF], proto ICMP (1), length 84)
    10.10.10.125 > 10.10.10.202: ICMP echo request, id 1110, seq 0, length 64
    08:58:36.988683 ip: (tos 0x0, ttl 63, id 1800, offset 0, flags [none], proto ICMP (1), length 84)
    10.10.10.202 > 10.10.10.125: ICMP echo reply, id 1110, seq 0, length 64
    2 packets captured
    2 packets received by filter
    0 packets dropped by kernel
```

3. Тестирование показывает, что трафик типа IPv4 пересылается через интерфейс wireguard.cali.

Kube-OVN Overlay Network поддерживает шифрование IPsec

В этом документе представлен подробный гид по включению и отключению шифрованного туннельного трафика IPsec в оверлейной сети Kube-OVN. Поскольку туннельный трафик OVN передаётся через физические маршрутизаторы и коммутаторы, которые могут находиться в ненадёжных публичных сетях или подвергаться атакам, включение шифрования IPsec эффективно предотвращает мониторинг и подмену данных трафика.

Содержание

Терминология

Термин	Объяснение
IPsec	Протокол и технология, используемые для защиты и проверки данных, передаваемых через интернет. Обеспечивает безопасную связь на уровне IP и в основном используется для создания виртуальных частных сетей (VPN) и защиты передачи IP-пакетов. IPsec

Термин	Объяснение
	обеспечивает безопасность данных преимущественно следующими способами: • **Шифрование данных**: с помощью технологий шифрования IPsec гарантирует, что данные не будут украдены или изменены во время передачи. Распространённые алгоритмы шифрования включают AES, 3DES и др. • **Проверка целостности данных**: IPsec использует хэш-функции (например, SHA-1, SHA-256) для проверки целостности данных, гарантируя, что данные не были изменены в процессе передачи. • **Аутентификация**: IPsec может проверять идентичность обеих сторон связи с помощью различных методов (например, предварительно разделённые ключи, цифровые сертификаты), чтобы предотвратить несанкционированный доступ. • **Управление ключами**: IPsec использует протокол Internet Key Exchange (IKE) для согласования и управления ключами шифрования.

Примечания

- Включение IPsec может вызвать кратковременное прерывание сети на несколько секунд.
- Если версия ядра — 3.10.0-1160.el7.x86_64, при включении функции IPsec в Kube-OVN могут возникнуть проблемы совместимости.

Предварительные требования

Выполните следующую команду, чтобы проверить, поддерживает ли текущее ядро операционной системы модули, связанные с IPsec. Если вывод показывает, что все

модули, связанные с XFRM, имеют значение `y` или `m`, это означает поддержку IPsec.

```
cat /boot/config-$(uname -r) | grep CONFIG_XFRM
```

Вывод:

```
CONFIG_XFRM_ALGO=y
CONFIG_XFRM_USER=y
CONFIG_XFRM_SUB_POLICY=y
CONFIG_XFRM_MIGRATE=y
CONFIG_XFRM_STATISTICS=y
CONFIG_XFRM_IPCOMP=m
```

Процедура

Примечание: если не указано иное, все команды необходимо выполнять в CLI-инструменте на Master-узле кластера.

Включение IPsec

1. Измените конфигурационный файл kube-ovn-controller.
    1. Выполните следующую команду для редактирования YAML-конфигурации kube-ovn-controller.

        ```
        kubectl edit deploy kube-ovn-controller -n kube-system
        ```

    2. Измените указанные поля согласно следующим инструкциям.

```
spec:
  template:
    spec:
      containers:
        - args:
            - --enable-ovn-ipsec=true # Добавьте это поле
          securityContext:
            runAsUser: 0 # Измените значение на 0
```

Объяснение полей:

- **spec.template.spec.containers[0].args**: Добавьте `- --enable-ovn-ipsec=true` под этим полем.
- **spec.template.spec.containers[0].securityContext.runAsUser**: Измените значение этого поля на 0.

3. Сохраните изменения.

2. Измените конфигурационный файл kube-ovn-cni.

1. Выполните следующую команду для редактирования YAML-конфигурации kube-ovn-cni.

```
kubectl edit ds kube-ovn-cni -n kube-system
```

2. Измените указанные поля согласно следующим инструкциям.

```
spec:
  template:
    spec:
      containers:
        - args:
            - --enable-ovn-ipsec=true # Добавьте это поле
          volumeMounts: # Добавьте путь монтирования, смонтируйте том с
именем ovs-ipsec-keys в контейнер
            - mountPath: /etc/ovs_ipsec_keys
              name: ovs-ipsec-keys
      volumes: # Добавьте том с именем ovs-ipsec-keys типа hostPath
        - name: ovs-ipsec-keys
          hostPath:
            path: /etc/origin/ovs_ipsec_keys
```

Объяснение полей:

- **spec.template.spec.containers[0].args**: Добавьте `- --enable-ovn-ipsec=true` под этим полем.
- **spec.template.spec.containers[0].volumeMounts**: Добавьте путь монтирования и смонтируйте том с именем ovs-ipsec-keys в контейнер.
- **spec.template.spec.volumes**: Добавьте том с именем ovs-ipsec-keys типа hostPath под этим полем.

3. Сохраните изменения.

3. Проверьте, успешно ли функция была включена.

1. Выполните следующую команду для входа в Pod kube-ovn-cni.

```
kubectl exec -it -n kube-system $(kubectl get pods -n kube-system -l app=kube-ovn-
cni -o=jsonpath='{.items[0].metadata.name}') -- /bin/bash
```

2. Выполните следующую команду для проверки количества соединений Security Associations. Если их (число узлов - 1) в статусе up, это означает успешное включение.

```
ipsec status | grep "Security"
```

Вывод:

```
Security Associations (2 up, 0 connecting):  # Поскольку в кластере 3 узла, видно, что количество соединений – 2 в статусе up
```

Отключение IPsec

1. Измените конфигурационный файл kube-ovn-controller.
    1. Выполните следующую команду для редактирования YAML-конфигурации kube-ovn-controller.

    ```
    kubectl edit deploy kube-ovn-controller -n kube-system
    ```

    2. Измените указанные поля согласно следующим инструкциям.

    ```
    spec:
      template:
        spec:
          containers:
            - args:
                - --enable-ovn-ipsec=false # Измените на false
              securityContext:
                runAsUser: 65534 # Измените значение на 65534
    ```

    Объяснение полей:

    - **spec.template.spec.containers[0].args**: Измените значение поля `enable-ovn-ipsec` на false.
    - **spec.template.spec.containers[0].securityContext.runAsUser**: Измените значение этого поля на 65534.

    3. Сохраните изменения.
2. Измените конфигурационный файл kube-ovn-cni.
    1. Выполните следующую команду для редактирования YAML-конфигурации kube-ovn-cni.

```
kubectl edit ds kube-ovn-cni -n kube-system
```

2. Измените указанные поля согласно следующим инструкциям.

- Конфигурация до изменения

```
spec:
  template:
    spec:
      containers:
        - args:
            - --enable-ovn-ipsec=true # Измените на false
          volumeMounts: # Удалите путь монтирования с именем ovs-ipsec-keys
            - mountPath: /etc/ovs_ipsec_keys
              name: ovs-ipsec-keys
      volumes: # Удалите том с именем ovs-ipsec-keys типа hostPath
        - name: ovs-ipsec-keys
          hostPath:
            path: /etc/origin/ovs_ipsec_keys
```

  Объяснение полей:

  - **spec.template.spec.containers[0].args**: Измените значение поля `enable-ovn-ipsec` на false.
  - **spec.template.spec.containers[0].volumeMounts**: Удалите путь монтирования с именем ovs-ipsec-keys.
  - **spec.template.spec.volumes**: Удалите том с именем ovs-ipsec-keys типа hostPath.

- Конфигурация после изменения

```
spec:
  template:
    spec:
      containers:
        - args:
            - --enable-ovn-ipsec=false
          volumeMounts:
      volumes:
```

3. Сохраните изменения.

3. Проверьте, успешно ли функция была отключена.

   1. Выполните следующую команду для входа в Pod kube-ovn-cni.

      ```
      kubectl exec -it -n kube-system $(kubectl get pods -n kube-system -l app=kube-ovn-cni -o=jsonpath='{.items[0].metadata.name}') -- /bin/bash
      ```

   2. Выполните следующую команду для проверки статуса соединения. Если вывода нет, это означает успешное отключение.

      ```
      ipsec status
      ```

ALB Monitoring

Содержание

Terminology

Term	Description
ALB	Самостоятельно разработанный платформой балансировщик нагрузки уровня 7.

Procedure

1. Перейдите в **Platform Management**.
2. В левой навигационной панели нажмите **Operation Center** > **Monitoring** > **Monitoring Dashboard**.

3. Нажмите на **Cluster** в верхней части страницы, чтобы переключиться на нужный кластер для мониторинга.

4. Нажмите **Switch** в правом верхнем углу страницы.

5. Вы можете войти в панель мониторинга **ALB Status** двумя способами:

   - Способ 1: Нажмите на карточку **container-platform**, чтобы развернуть каталог мониторинга, затем нажмите на название **ALB Status**, чтобы войти в панель мониторинга. При необходимости вы можете установить эту панель мониторинга в качестве основной.

   - Способ 2: Введите ключевое слово (например, alb) в строку поиска и выполните поиск, затем нажмите на название **ALB Status**, чтобы войти в панель мониторинга. При необходимости вы можете установить эту панель мониторинга в качестве основной.

6. Просматривайте различные метрики мониторинга через панель.

   - **Выбор namespace для мониторинга**: Нажмите на **namespace** в верхней части страницы, чтобы выбрать namespace для мониторинга, по умолчанию — все, то есть мониторинг всех namespace.

   - **Выбор ALB для мониторинга**: Нажмите на **name** в верхней части страницы, чтобы выбрать ALB для мониторинга, по умолчанию — все, то есть мониторинг всех ALB.

Monitoring Metrics

Отображает метрики мониторинга общего трафика, использования ресурсов, Ingress (входящие правила), HTTPRoute (правила маршрутизации типа HTTPRoute) и Rule (правила, которые не являются ни Ingress, ни HTTPRoute) для выбранного ALB за **последние 5 минут**.

Примечание: Все данные — это данные мониторинга, собранные за **последние 5 минут**.

ALB Traffic Monitoring

Monitoring Metric	Description
Active Connections	Количество активных соединений на выбранном ALB.
Requests Per Second	Общее количество запросов, получаемых в секунду на выбранном ALB.
Error Rate	Доля запросов с ошибками 4XX (например, 404) и 5XX, возникающих в секунду на выбранном ALB.
Latency	Средняя задержка запросов на выбранном ALB.

ALB Resource Usage

Monitoring Metric	Description
CPU Usage	Использование CPU выбранным ALB.
Memory Usage	Использование памяти выбранным ALB.
Network Receive/Transmit	Пропускная способность сети (входящий/исходящий трафик) выбранного ALB.
Disk Read/Write Rate	Пропускная способность диска (чтение/запись) выбранного ALB.

Ingress, HTTPRoute, Rule Traffic Monitoring

Monitoring Metric	Description
QPS (Queries Per Second)	Количество запросов в секунду, получаемых Ingress/HTTPRoute/Rule на выбранном ALB, единица измерения по умолчанию — req/s.

Monitoring Metric	Description
Request BPS (Bytes Per Second)	Общий размер запросов в байтах в секунду, получаемых Ingress/HTTPRoute/Rule на выбранном ALB.
Response BPS (Bytes Per Second)	Общий размер ответов в байтах в секунду, отправляемых Ingress/HTTPRoute/Rule на выбранном ALB.
Error Rate	Процент ошибок, возникших при обработке запросов Ingress/HTTPRoute/Rule на выбранном ALB.
P50, P90, P99	Время отклика запросов на выбранном ALB, а именно медианное время отклика. Показывает, что 50%, 90% и 99% запросов имеют время отклика меньше или равно этому значению. **Примечание**: Принцип P50, P90 и P99 заключается в сортировке собранных данных от меньшего к большему и взятии значений данных на позициях 50%, 90% и 99%; таким образом, 50%, 90% и 99% собранных данных находятся ниже этого значения. Процентили помогают анализировать распределение данных и выявлять различные крайние ситуации.
Upstream P50, Upstream P90, Upstream P99	Время отклика запросов к upstream-сервисам. Показывает, что 50%, 90% и 99% запросов, отправленных к upstream-сервисам, имеют время отклика меньше или равно этому значению.

Menu

Устранение неполадок

Как решить проблемы межузловой коммуникации в ARM-средах?

Определение причины ошибки

Как решить проблемы межузловой коммуникации в ARM-средах?

При использовании более низких версий ядра и некоторых отечественных сетевых карт может возникать проблема некорректного вычисления контрольных сумм сетевой картой после включения Checksum Offload. Это может привести к сбоям в коммуникации между узлами в Kube-OVN Overlay сети. Конкретные решения следующие:

- **Решение 1: Обновление версии ядра**. Рекомендуется обновить версию ядра до 4.19.90-25.16.v2101 или выше.
- **Решение 2: Отключение Checksum Offload**. Если невозможно сразу обновить версию ядра и возникают проблемы с межузловой коммуникацией, можно отключить Checksum Offload для физической сетевой карты с помощью следующей команды.

```
ethtool -K eth0 tx off
```

Menu

Определение причины ошибки

Поле `X-ALB-ERR-REASON` в заголовке ответа на ошибочный запрос указывает причину ошибки.

Причина ошибки может быть следующей:

```
InvalidBalancer : no balancer found for xx # это означает, что для сервиса не найден конечный пункт
BackendError : read xxx byte data from backend # это означает, что backend действительно вернул ответ, ошибка не вызвана alb.
InvalidUpstream : no rule match # это означает, что запрос не соответствует ни одному правилу, поэтому alb возвращает 404.
```