4 Лаба ABC

Показать Service discovery Traefik на примере docker-провайдера (Запуск контейнера с определенными labels должен автоматически привести к появлению нового endpoint в traefik и, как следствие, включение его в пул распределения запросов)

1. Напишем docker-compose файл:

```
version: "3.8"

services:
  traefik:
    image: traefik:v2.10
    container_name: traefik
    ports:
      - "80:80"
      - "8080:8080"
    command:
      - "--api.insecure=true"
      - "--providers.docker=true"
      - "--entrypoints.web.address=:80"
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock:ro"

  backend:
    image: traefik/whoami
    deploy:
      replicas: 3
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.backend.rule=Host(`localhost`)"
      -
"traefik.http.services.backend.loadbalancer.server.port=80"
```

2. Запустим его:

   docker-compose up -d

3. Пошлем запросы и проверим маршрутизацию

   curl -H "Host: localhost" http://localhost
   http://localhost:8080

Распределение запросов с помощью lvs/ipvs - для этого использовать три ВМ. У каждой свой ip адрес. Одна ВМ является фронтендом балансировщика, две другие - бэкэндами.

Установим утилиту с помощью команды:

bash
```
sudo apt-get install -y ipvsadm
```

Создание виртуального сервиса

Теперь мы можем использовать CLI для создания нового виртуального сервиса. Синтаксис команды:

bash
Копировать код
```
ipvsadm COMMAND [protocol] service-address [scheduling-method] [persistence options]
```

В документации **ipvsadm** указано, что:

- Флаг `-A` добавляет виртуальный сервис.
- Флаг `-s` задает метод распределения нагрузки. Например:
  - `rr` — Round Robin (циклический);
  - `wrr` — Weighted Round Robin (взвешенный циклический);
  - `lc` — Least-Connection (минимум подключений);
  - `lblc` — Locality-Based Least-Connection и другие.

Создадим виртуальный сервис для адреса `100.100.100.100:80` с методом распределения нагрузки Round Robin:

bash
Копировать код
```
sudo ipvsadm -A -t 100.100.100.100:80 -s rr
```

Создание двух Docker-контейнеров

Мы будем использовать образ **jwilder/whoami**, который возвращает ID контейнера.

Запустите два контейнера:

bash
Копировать код
```
docker run -d -p 8000:8000 --name first -t jwilder/whoami
docker run -d -p 8001:8000 --name second -t jwilder/whoami
```

Пример вывода:

- Для первого контейнера: `cd977829ae0c76236a1506c497d5ce1628f1f701f8ed074916b21fc286f3d0d1`
- Для второго контейнера: `5886b1ed7bd4095cb02b32d1642866095e6f4ce1750276bd9fc07e91e2fbc668`

Узнаем IP-адреса контейнеров:

bash
Копировать код
```
docker inspect -f '{{range
.NetworkSettings.Networks}}{{.IPAddress}}{{end}}' first
docker inspect -f '{{range
.NetworkSettings.Networks}}{{.IPAddress}}{{end}}' second
```

Пример вывода:

- Первый контейнер: `172.17.0.2`
- Второй контейнер: `172.17.0.3`

Проверим, что контейнеры работают:

bash
Копировать код
```
curl 172.17.0.2:8000
# Ответ: I'm cd977829ae0c
```

Добавление IP-адресов контейнеров в виртуальный сервис

Теперь добавим IP-адреса контейнеров в виртуальный сервис. Используем:

- Флаг `-a` для добавления сервера в виртуальный сервис;
- Флаг `-t` для указания виртуального сервиса;
- Флаг `-m` для NAT (masquerading).

Добавляем серверы:

bash
Копировать код

```
sudo ipvsadm -a -t 100.100.100.100:80 -r 172.17.0.2:8000 -m
sudo ipvsadm -a -t 100.100.100.100:80 -r 172.17.0.3:8000 -m
```