Биоинформатика

Фонд интеллект

Инфраструктура вычислений в биоинформатике

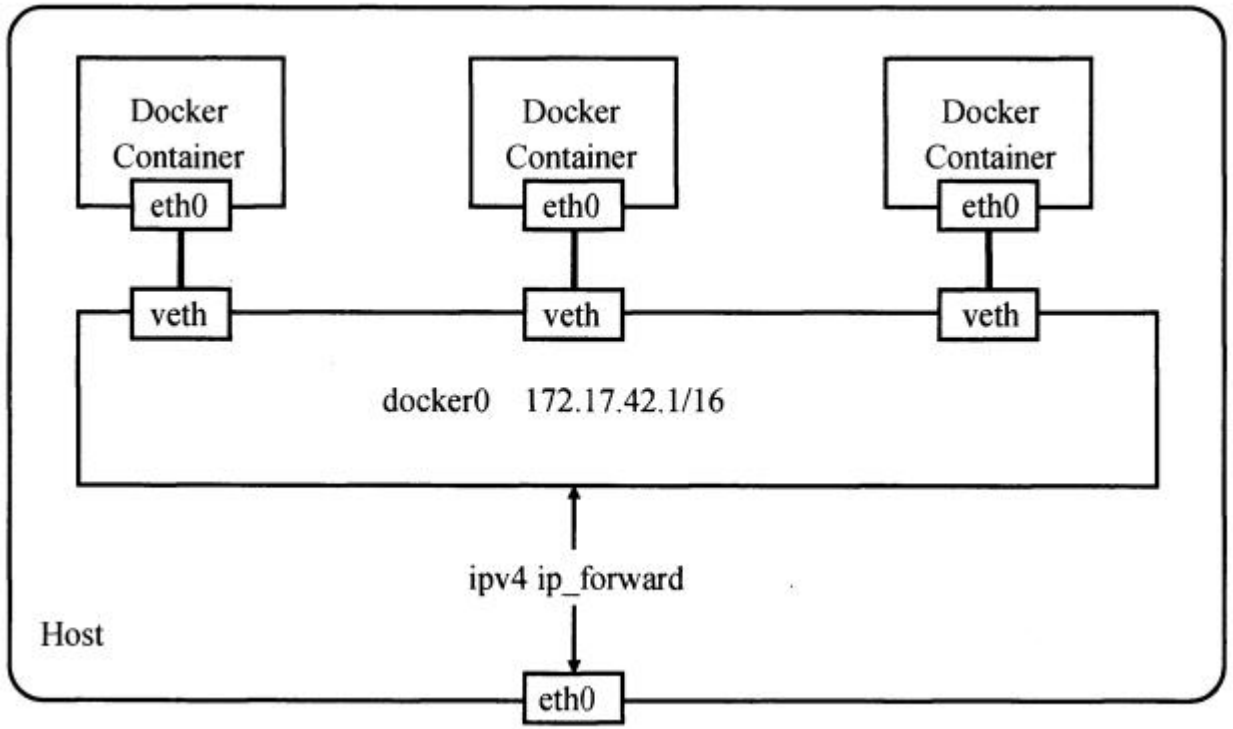
Лекция 6. Сеть Docker. Коммуникация контейнеров. Логирование. Docker Compose.

Типы сетей Docker

Имеется несколько драйверов по умолчанию, которые обеспечивают основной функционал по работе с сетью:

- ***none*:** отключение всех сетевых ресурсов.
- ***bridge*:** сетевой драйвер по умолчанию. По сути, это мост между контейнером и хостовой машиной. Мостовые сети обычно используются, когда приложения выполняются в автономных контейнерах, которые должны взаимодействовать друг с другом, например пример Nginx + MySQL
- ***host*:** для автономных контейнеров устраняется сетевая изолированность между контейнером и хостом Docker и напрямую используются сетевые ресурсы хоста.
- ***overlay*:** наложенные сети соединяют несколько демонов Docker.
- ***macvlan*:** сети Macvlan позволяют присваивать контейнеру MAC-адрес, благодаря чему он выглядит как физическое устройство в сети

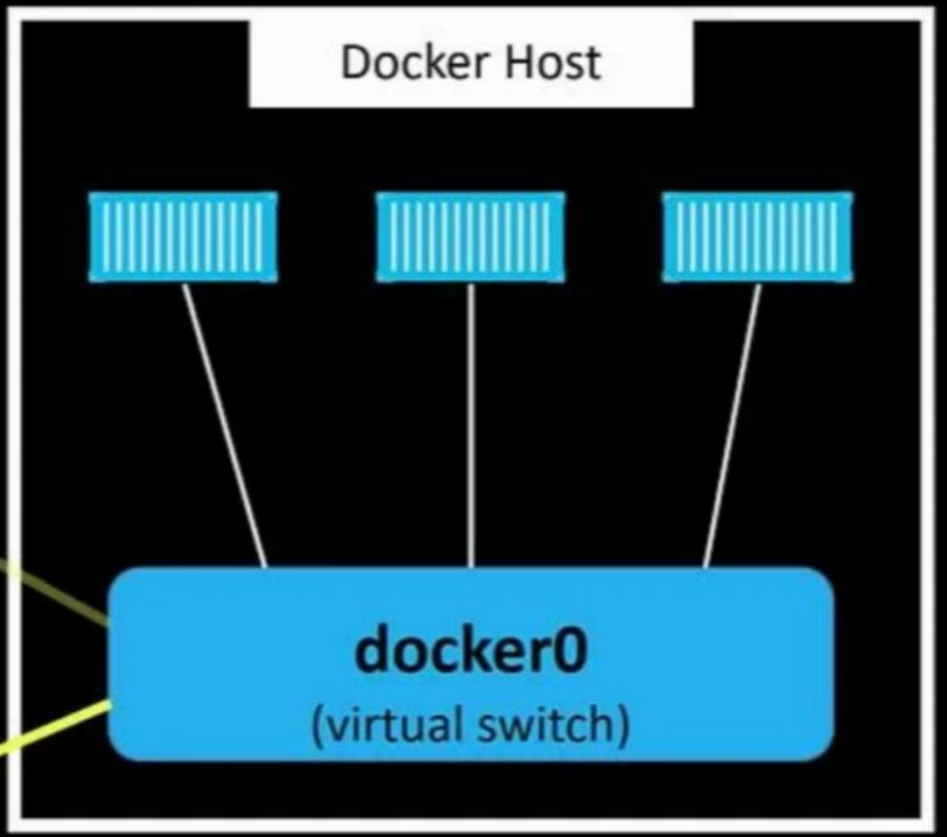
Стандартная bridge-сеть Docker



Контейнеры общаются друг с другом через мост (bridge) **`docker0`**, как через виртуальный свитч.

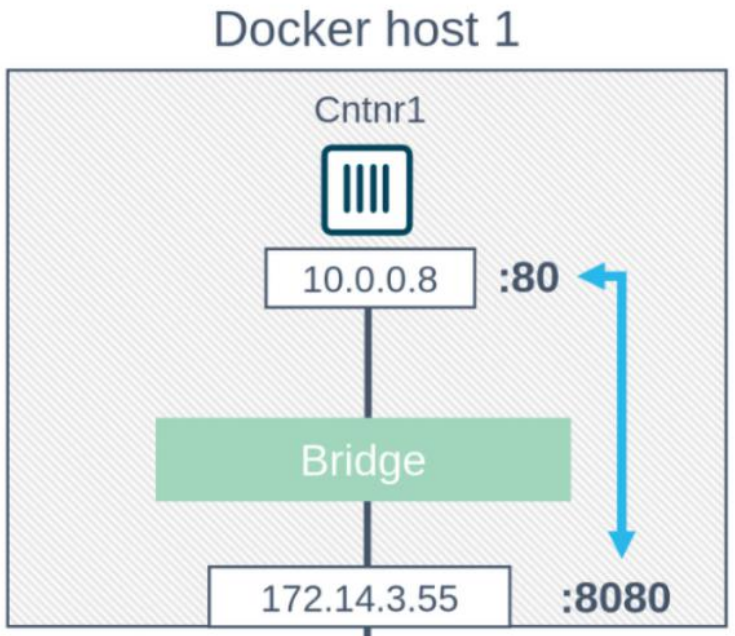
В каждом контейнере есть виртуальный Ethernet-интерфейс **`eth0`**. От **`eth0`** каждого контейнера вниз идёт линия к **`veth`** — это парный виртуальный интерфейс (как виртуальный кабель с двумя концами)

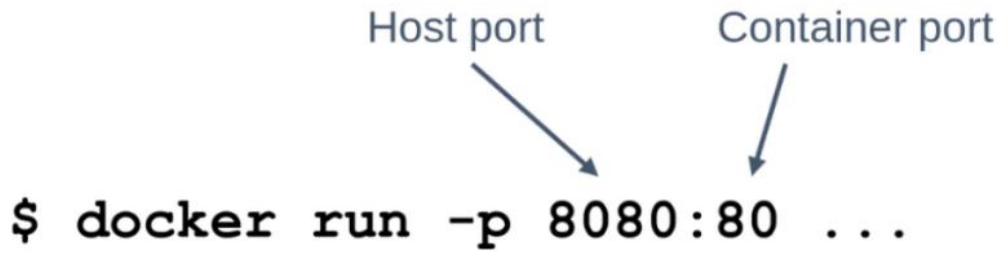
```
        "Id": "191853460aad5335fc29692d9cd8cc94357ca4c1cbd6f99a9fc50ed1029fc77a",
        "Created": "2016-12-20T21:47:45.342696205Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16",
                    "Gateway": "172.17.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Containers": {},
        "Options": {
            "com.docker.network.bridge.default_bridge": "true",
            "com.docker.network.bridge.enable_icc": "true",
            "com.docker.network.bridge.enable_ip_masquerade": "true",
            "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
            "com.docker.network.bridge.name": "docker0",
            "com.docker.network.driver.mtu": "1500"
        },
        "Labels": {}
    }
]
root@node1:/home/ubuntu#
```



Стандартная bridge-сеть Docker

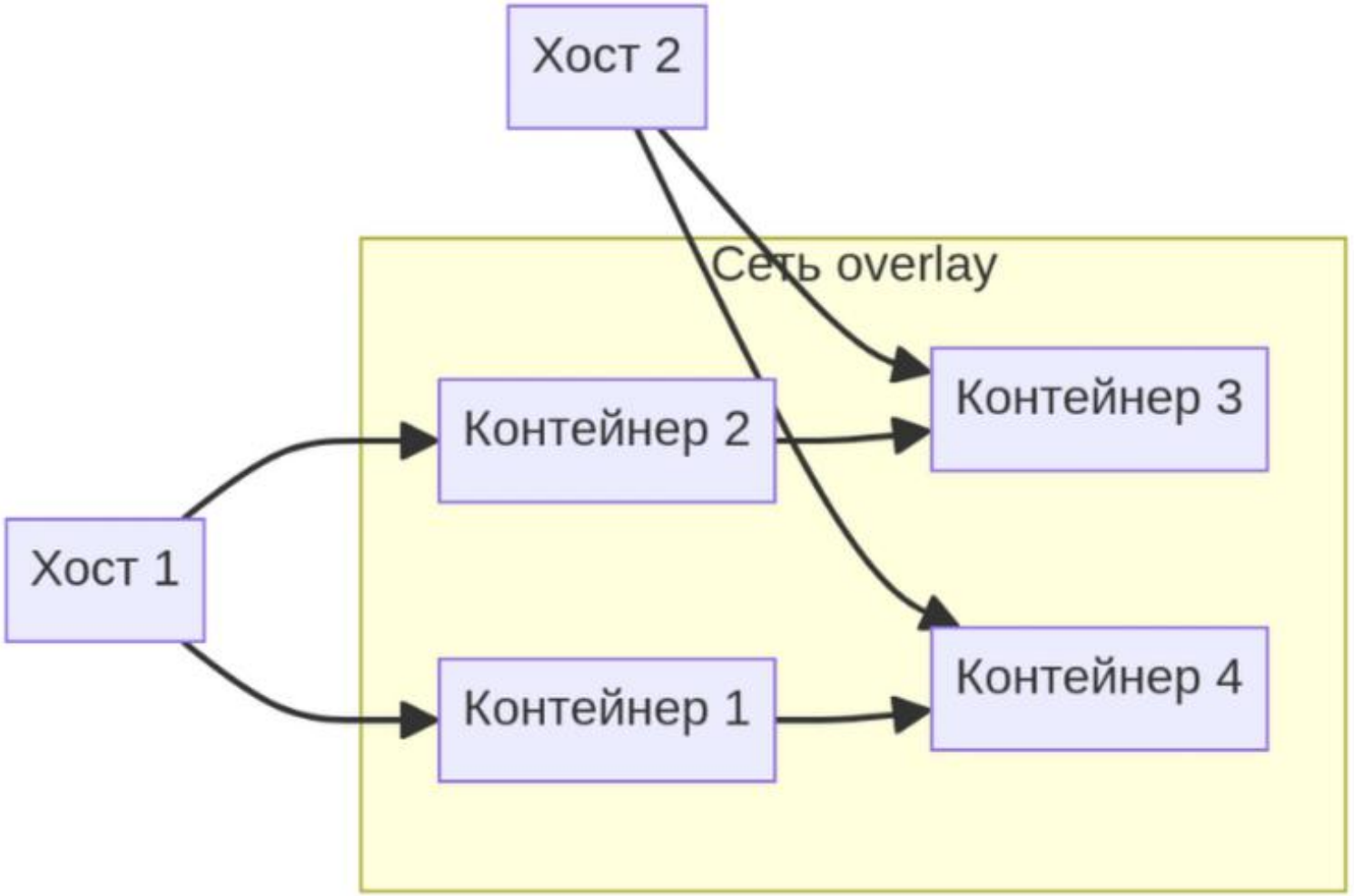
Для выхода в интернет используется NAT + IP forwarding (хост маскирует IP контейнеров под свой внешний IP).





Трансляция сетевых адресов (NAT)!!

overlay драйвер сети



Сеть контейнеров

Проверить сетевое подключение между контейнерами или между контейнером и хостом можно с помощью команд `ping` и `telnet` (ssh без криптографии)

Команда для подробностей сети Docker, включая подключенные контейнеры и их IP-адреса:

```
docker network inspect
```

Примеры

- Создание мостовой сети:
```
docker network create --driver bridge
my_bridge_network
```
- Запуск контейнеров в созданной сети:
```
docker run -d --name container1 --network
my_bridge_network nginx
docker run -d --name container2 --network
my_bridge_network nginx
```
- Контейнеры могут взаимодействовать друг с другом по имени (Docker предоставляет свой DNS):
```
docker exec container1 ping container2
```

Логирование

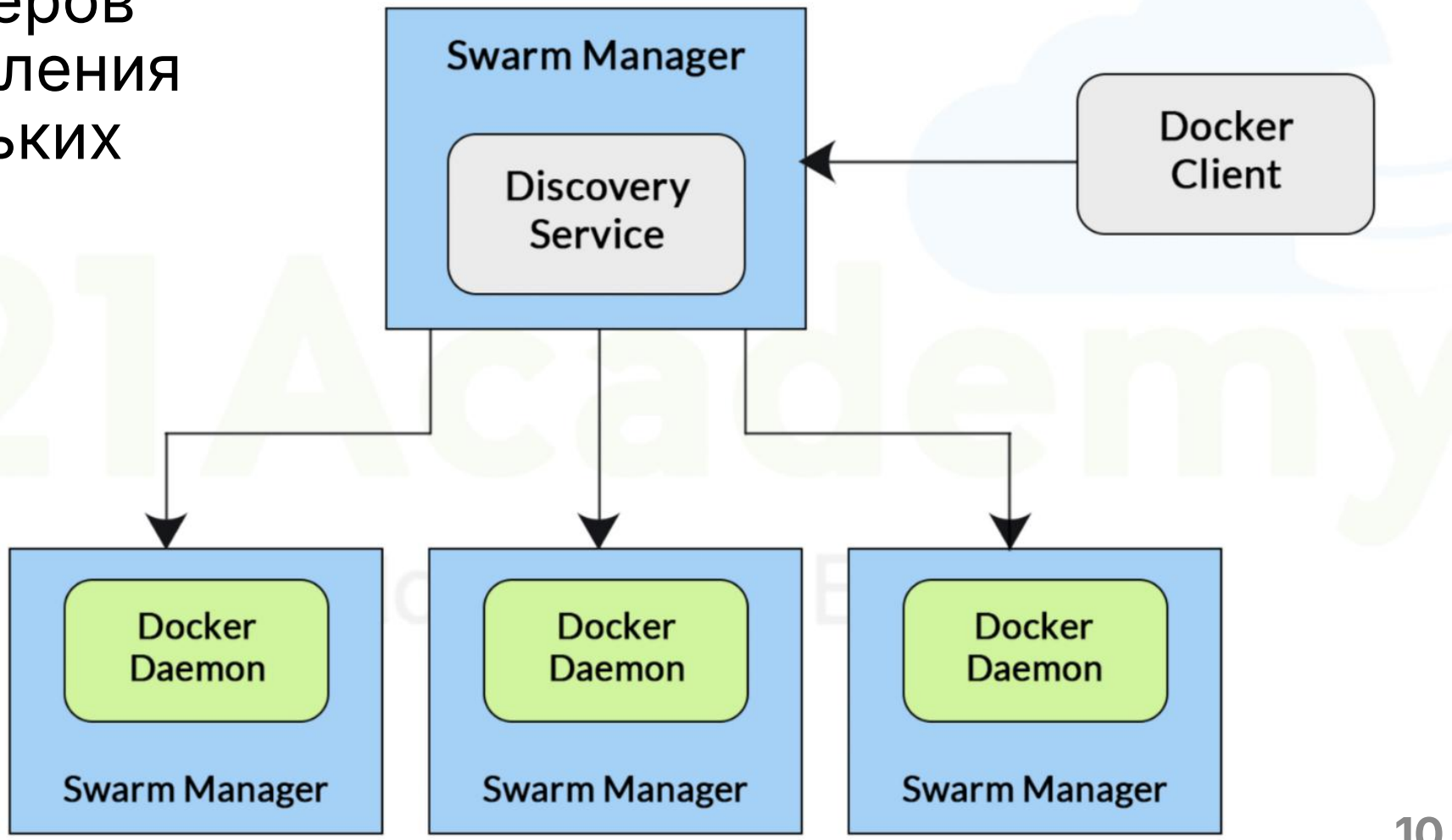
- Запуск:
  `docker run -d -p 5000:80 --name demo nginx`

- Проверка:
  `curl http://localhost:5000`

- Посмотреть логи в реальном времени:
  `docker logs -f demo`

Docker Swarm

– встроенный в Docker оркестратор контейнеров для создания и управления кластером из нескольких Docker-хостов.



Docker Compose

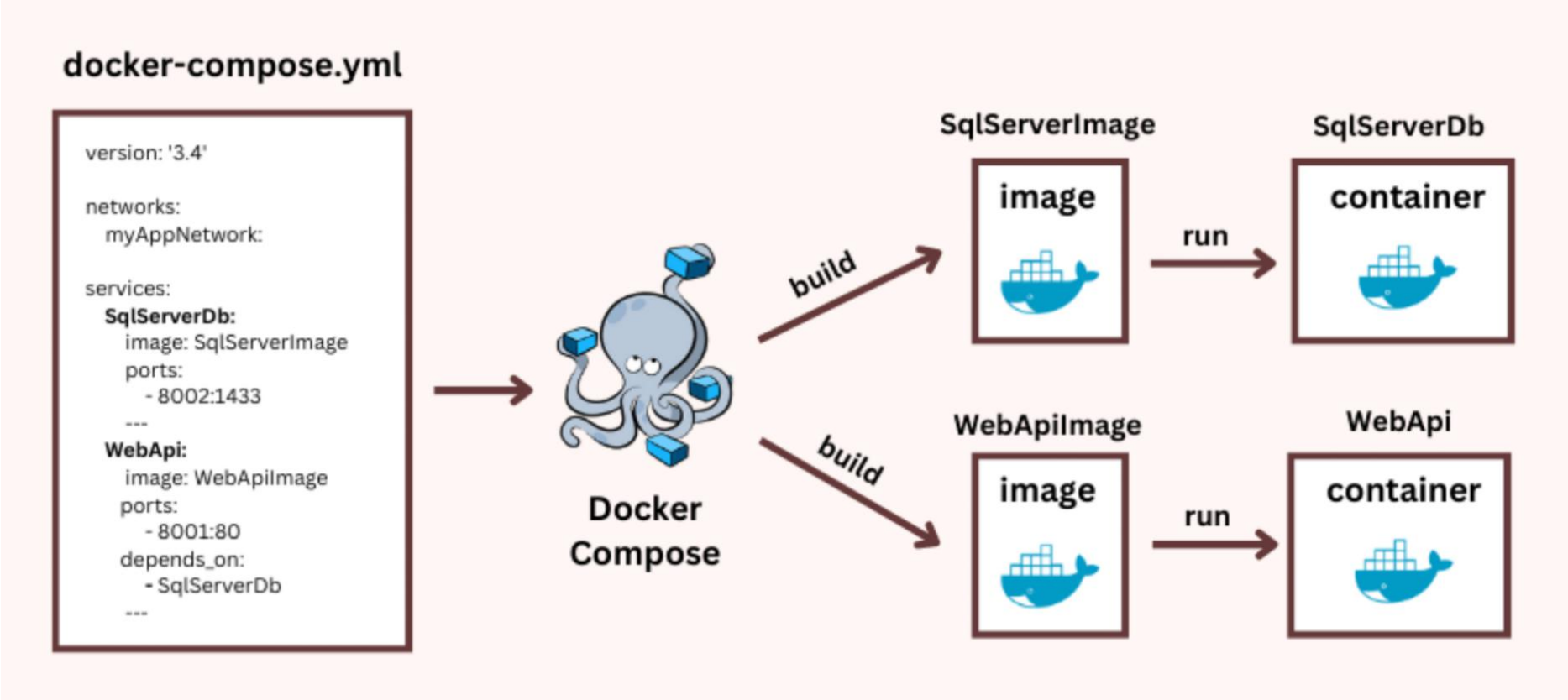
— инструмент для описания и запуска приложений, состоящих из одного или нескольких контейнеров через YAML-файл.

```yaml
services:
  recommendation-engine:
    image: ubuntu
    tty: true
    volumes:
    - DataVolume:/DataVolume
    labels:
      brownout.feature: "optional"
    deploy:
      replicas: 2
      restart_policy:
        condition: none
      placement:
        constraints: [node.role == worker]

  user-db:
    image: weaveworksdemos/user-db
    hostname: user-db
    deploy:
      placement:
        constraints: [node.role == manager]
```

Docker Compose

— инструмент для описания и запуска приложений, состоящих из одного или нескольких контейнеров через YAML-файл.



Docker Compose

Без Docker Compose запуск простого приложения выглядит так:

```
docker network create app-net
docker volume create db-data
docker run -d --name db --network app-net -v db-
data:/var/lib/mysql mysql
docker run -d --name api --network app-net api-image
docker run -d -p 80:80 --name web --network app-net nginx
```

Пример docker-compose.yml файла

```yaml
version: "3.9"                # Указывает версию формата Docker Compose

services:                     # Раздел со всеми сервисами (контейнерами)

  web:                        # Сервис 1 — фронтенд
    image: nginx:latest       # Используемый образ
    ports:                    # Пробрасываем порты на хост
      - "80:80"               # 80 порт хоста → 80 порт контейнера
    depends_on:               # Этот сервис стартует только после api
      - api

  api:                        # Сервис 2 — бэкенд
    image: my-api:latest      # Свой кастомный образ приложения
    expose:                   # Порт открыт только внутри docker-сети
      - "8080"                # API слушает внутри на 8080

  db:                         # Сервис 3 — база данных
    image: postgres:15        # Образ PostgreSQL
    environment:              # Переменные окружения для контейнера
      POSTGRES_PASSWORD: secret   # Пароль базы данных
    volumes:                  # Подключаем постоянное хранилище
      - db-data:/var/lib/postgresql/data  # Данные базы сохраняются вне контейнера

volumes:                      # Раздел объявления томов
  db-data:                    # Именованный том для сохранения данных PostgreSQL
```

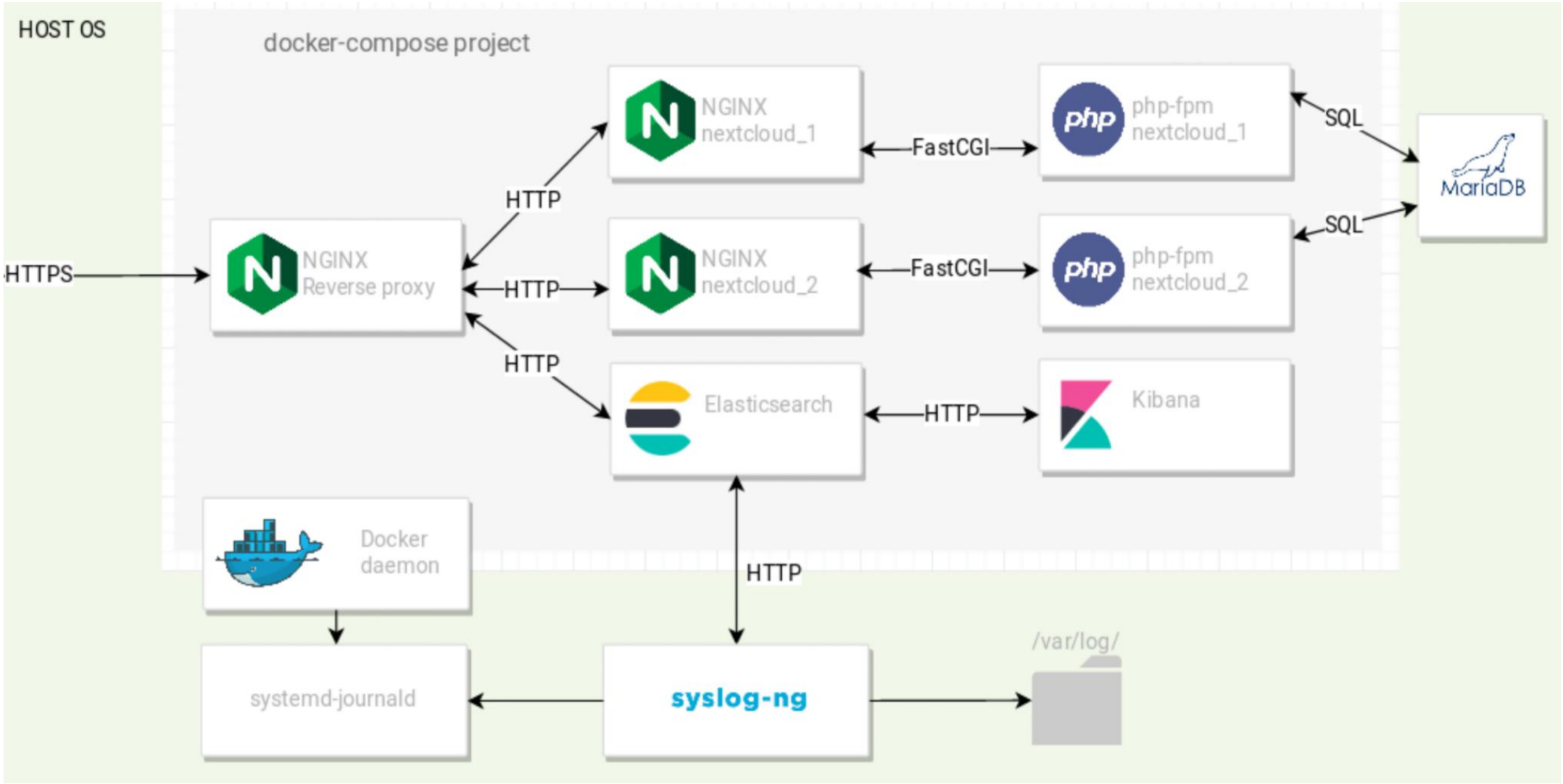
Docker Compose

Используйте **`.env`** для секретов (не коммитьте **`.env`**!).

Также можно в рантайм shell с помощью **`export`** поместить переменную.

Docker Compose

```
docker compose up -d
docker compose down -v
docker compose logs -f web
```



Стек, развёрнутый через один docker-compose.yml с Syslog-ng как коллектор логов и ELK Stack (централизованное логирование Elasticsearch + Kibana) для поиска, анализа и мониторинга логов.