

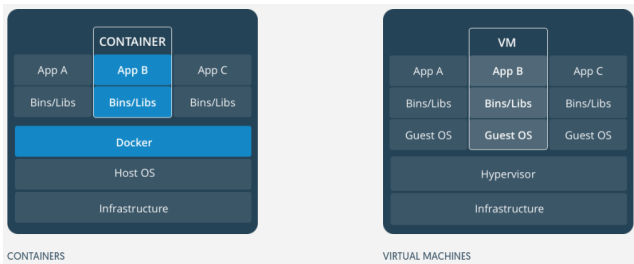
Развёртывание, Docker

Юрий Литвинов
y.litvinov@spbu.ru

28.11.2024

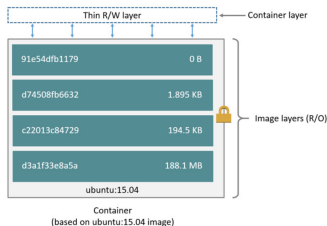
Docker

- ▶ Средство для “упаковки” приложений в изолированные контейнеры
- ▶ Что-то вроде легковесной виртуальной машины
- ▶ DSL для описания образов
- ▶ Публичный репозиторий
- ▶ Стандарт де-факто для деплоя веб-приложений



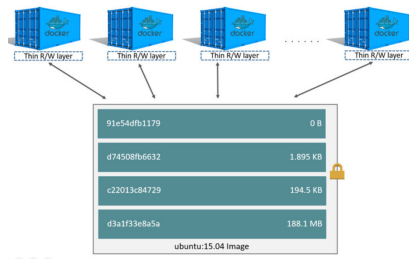
Docker Image

- ▶ Окружение и приложение
- ▶ Состоит из слоёв
 - ▶ Все слои read-only
 - ▶ Образы делят слои между собой как процессы делят динамические библиотеки
- ▶ На основе одного образа можно создать другой



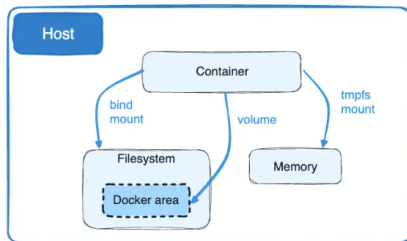
Docker Container

- ▶ Образ с дополнительным write слоем
- ▶ Содержит один запущенный процесс
- ▶ Может быть сохранен как новый образ



Хранение данных

- ▶ По умолчанию все данные хранятся только в памяти
- ▶ Для БД и т.п. — volumes и bind mounts



- ▶ `docker run -d --mount source=myvol,target=/app nginx:latest`

Базовые команды

- ▶ `docker run` — запускает контейнер (при необходимости делает pull)
 - ▶ `-d` — запустить в фоновом режиме
 - ▶ `-p host_port:container_port` — прокинуть порт из контейнера на хост
 - ▶ `-i -t` — запустить в интерактивном режиме
 - ▶ Пример: `docker run -it ubuntu /bin/bash`
- ▶ `docker ps` — показывает запущенные контейнеры
 - ▶ Пример: `docker run -d nginx; docker ps`
- ▶ `docker stop` — останавливает контейнер (шлёт SIGTERM, затем SIGKILL)
- ▶ `docker exec` — запускает дополнительный процесс в контейнере

Dockerfile

```
FROM mcr.microsoft.com/dotnet/aspnet:8.0 AS base
WORKDIR /app
EXPOSE 80
EXPOSE 443
```

```
FROM mcr.microsoft.com/dotnet/sdk:8.0 AS build
WORKDIR /src
COPY ["ConferenceRegistration.csproj", "."]
RUN dotnet restore "./ConferenceRegistration.csproj"
COPY . .
WORKDIR "/src/."
RUN dotnet build "ConferenceRegistration.csproj" -c Release -o /app/build
```

```
FROM build AS publish
RUN dotnet publish "ConferenceRegistration.csproj" -c Release -o /app/publish /p:UseAppHost=false
```

```
FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "ConferenceRegistration.dll"]
```

Сборка контейнера

- ▶ `docker build`
 - ▶ `docker build -t conferenceregistration:latest .`
- ▶ `-t` — имя образа, через двоеточие — тэг, или версия
 - ▶ В production не стоит использовать `latest`

Пушим на Docker Hub

- ▶ Регистрируемся на Docker Hub
- ▶ `docker images ls`
- ▶ `docker image tag conferenceregistration:latest <ваш юзернейм на Docker Hub>/conferenceregistration:latest`
- ▶ `docker push <ваш юзернейм на Docker Hub>/conferenceregistration:latest`
- ▶ `docker run -d -p 80:80 <ваш юзернейм на Docker Hub>/conferenceregistration:latest`

Yandex.Cloud

- ▶ Облачный хостинг с некоторыми бесплатными возможностями и стартовым грантом
- ▶ Умеет много чего (виртуальные машины в облаке, Cloud Functions, облачные СУБД, включая их собственную, распознавание/синтез речи, ML-инструменты, CDN и т.п.)
- ▶ Нам тут интереснее всего Serverless Containers
- ▶ Возможно, потребуется привязать карточку

Загрузка образа в Yandex Container Registry

- ▶ Логинимся в <https://console.cloud.yandex.ru>
- ▶ Создаём облако и каталог (кнопкой "+" слева сверху)
- ▶ Ставим себе локально Yandex Cloud CLI
- ▶ Делаем `yc init`, авторизуемся
- ▶ Создаём реестр Docker-образов:
`yc container registry create --name my-first-registry`
- ▶ Говорим Docker про аутентификацию в Yandex Container Registry: `yc container registry configure-docker`
- ▶ Присваиваем нашему образу тэг вида
`cr.yandex/<ID реестра>/<имя Docker-образа>:<тег>`
- ▶ Пушим, например:
`docker push cr.yandex/crpc9qeoft236r8tfalm/ubuntu:hello`
- ▶ Идём в консоль Яндекса и проверяем

Запуск образа в Serverless Containers

- ▶ Идём в <https://console.cloud.yandex.ru>
- ▶ Выбираем Serverless Containers
- ▶ Жмём “Создать”, вводим имя и описание приложения
- ▶ Указываем потребные ресурсы (для учебных целей — по минимуму, чтобы попасть в free tier)
- ▶ Выбираем образ из YCR
- ▶ Создаём сервисный аккаунт
- ▶ Жмём “Создать ревизию”
- ▶ Оно некоторое время потормозит, но затем приложение появится в списке контейнеров
- ▶ Кликаем на него, “Ссылка для вызова” — это URL для запуска приложения

Docker Compose

- ▶ Большинство полезных приложений состоит более чем из одного контейнера
- ▶ Создавать кучу контейнеров руками и конфигурировать им каждый раз URL — сложно
- ▶ Есть оркестраторы, которые всё делают сами:
 - ▶ Docker Compose
 - ▶ Kubernetes
- ▶ Запуск с общей конфигурацией, перезапуск при необходимости, контроль ресурсов, масштабирование, внутренняя сеть, ...

Пример, compose.yaml

```
services:
  todo-app:
    build:
      context: ./app
    depends_on:
      - todo-database
    environment:
      NODE_ENV: production
    ports:
      - 3000:3000
      - 35729:35729
  todo-database:
    image: mongo:6
    volumes:
      - database:/data/db
    ports:
      - 27017:27017
```

© <https://github.com/docker/multi-container-app>

Запуск

- ▶ Запуск: `docker compose up -d`
- ▶ Остановка: `docker compose down`
- ▶ Посмотреть, что происходит: `docker compose logs -f`
- ▶ YSC это всё пока не умеет, но есть VM с Container Optimized Image