Практика 8: Развёртывание, Docker

Юрий Литвинов
yurii.litvinov@gmail.com
Спасибо Владиславу Танкову (vdtankov@gmail.com) за предоставленные материалы

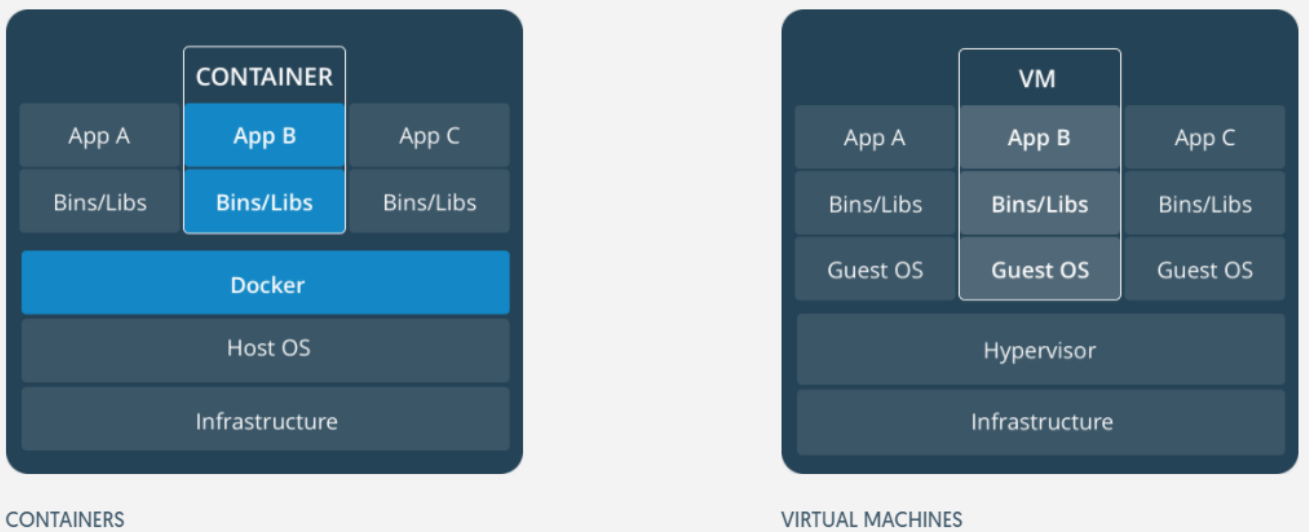
26.05.2020г

CI и CD

- Проекты разрабатываются большими командами
  - Разработчики
  - Тестировщики
  - Администраторы
- Разное окружение, разные форматы, разная история изменений
- Нужен артефакт, инкапсулирующий в себе окружение и приложение
  - Легко создавать и передавать
  - Не влияет на производительность

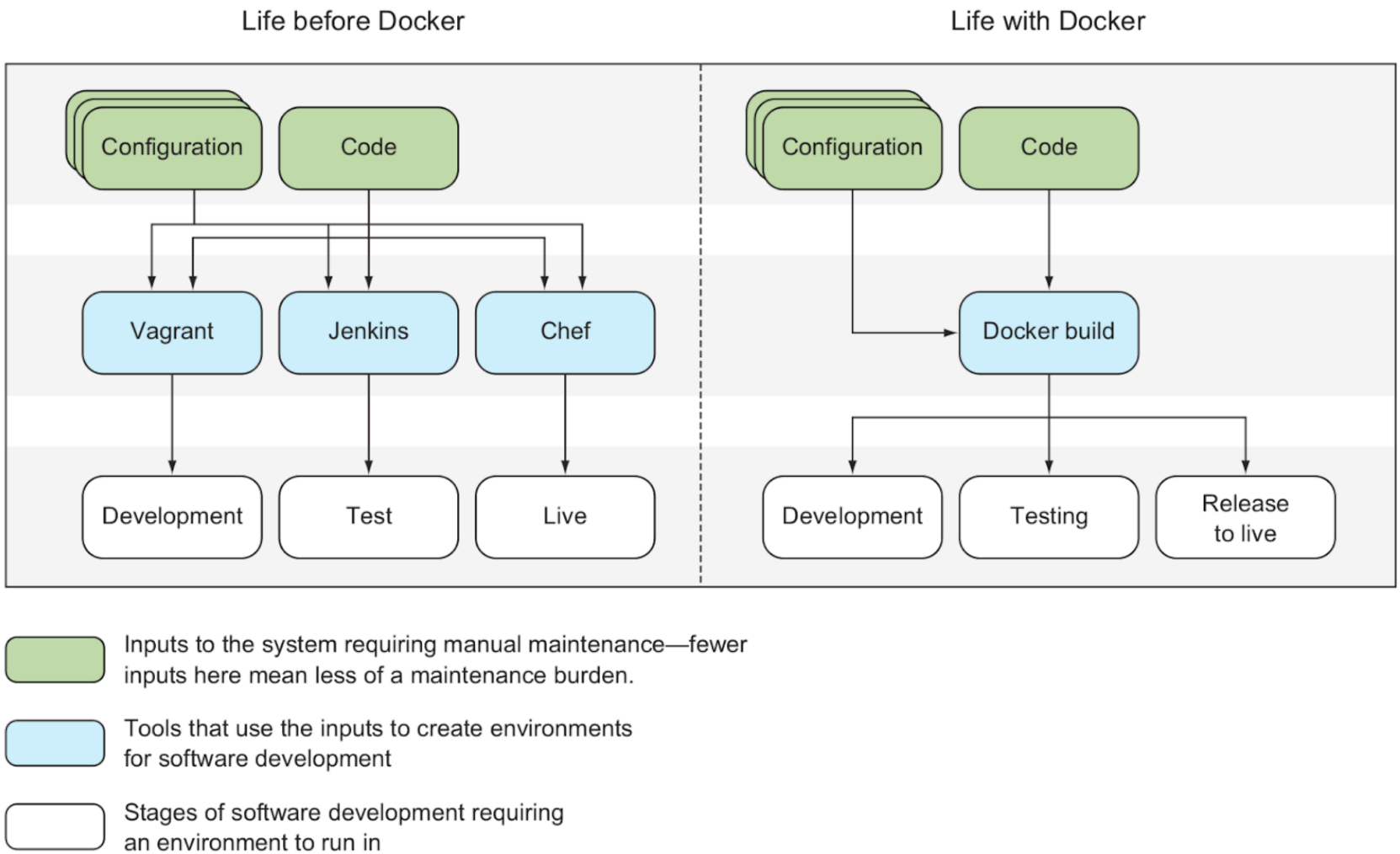
Docker

- Средство для “упаковки” приложений в изолированные контейнеры
- Что-то вроде легковесной виртуальной машины
- Широкий инструментарий: DSL для описания образов, публичный репозиторий, поддержка оркестраторами



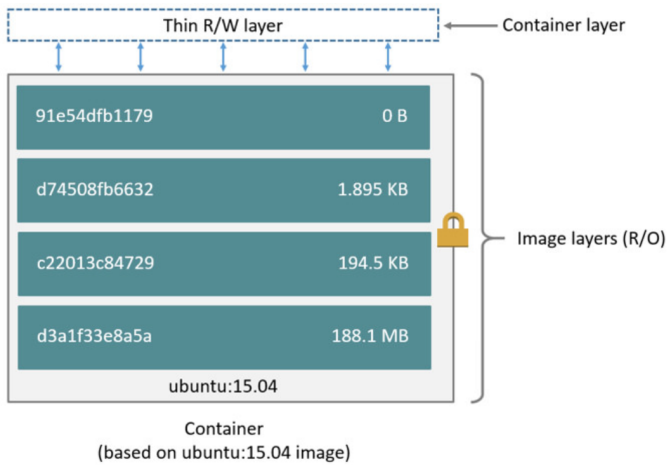
© https://www.docker.com

Чем он помогает



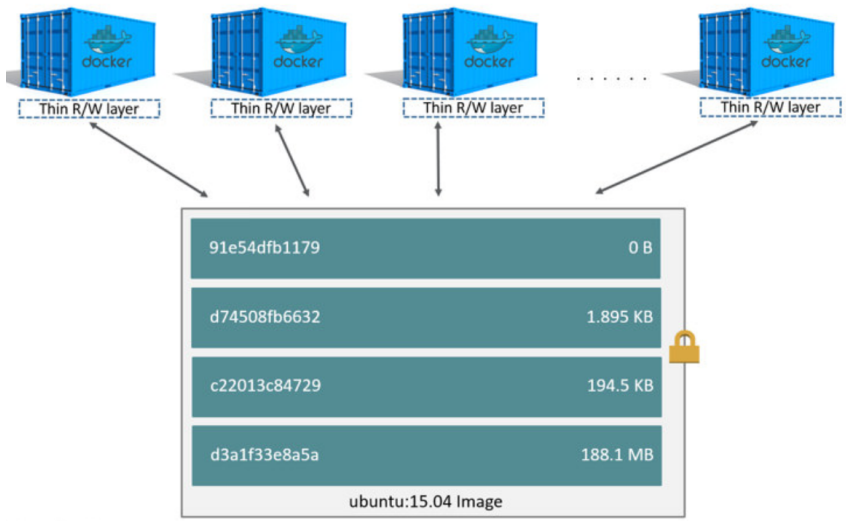
Docker Image

- 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

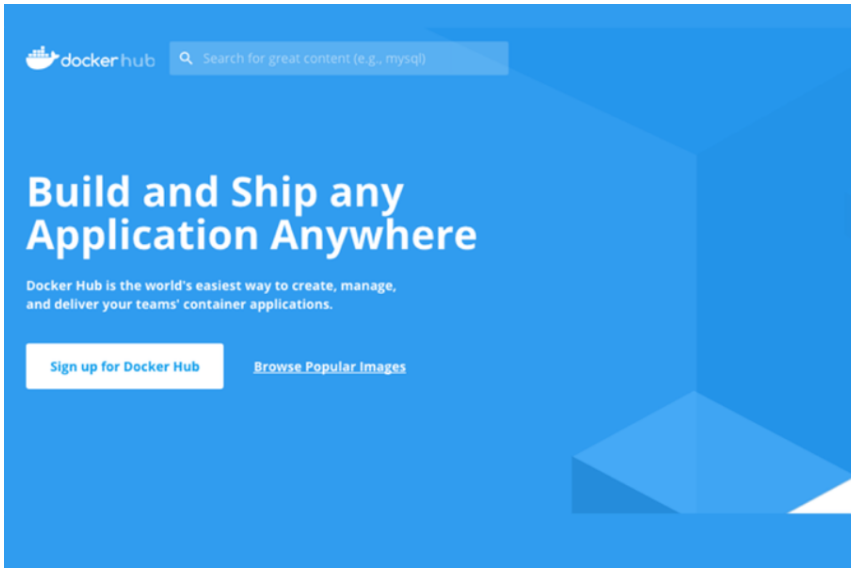


Клиент и сервер

- Docker — это клиент-серверное приложение
- Docker Daemon
  - REST API
  - Управляет запущенными контейнерами
  - Управляет локальным репозиторием образов
- Docker Client
  - CLI для Docker
  - Запросы по HTTP пересылает Docker Daemon

DockerHub

- Внешний репозиторий образов
  - Официальные образы
  - Пользовательские образы
  - Приватные репозитории
- Простой CI/CD
- Высокая доступность



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

```
# Use an official Python runtime as a parent image
FROM python:2.7-slim

# Set the working directory to /app
WORKDIR /app

# Copy the current directory contents into the container at /app
ADD . /app

# Install any needed packages specified in requirements.txt
RUN pip install --trusted-host pypi.python.org -r requirements.txt

# Make port 80 available to the world outside this container
EXPOSE 80

# Define environment variable
ENV NAME World

# Run app.py when the container launches
CMD ["python", "app.py"]
```

Команды

- FROM — взять как начальный образ указанный
- WORKDIR — выбрать папку в качестве текущей
- RUN — выполнить команду
- ADD — копировать файл/папку/архив
- ENV — установить environment переменную
- ENTRYPOINT — установить команду для запуска процесса
- CMD — установить принятые по умолчанию аргументы
- EXPOSE — разрешить доступ к контейнеру по порту
- VOLUME — определить mount point для volume
- ARG — build аргументы

Пример: Redis

```
FROM ubuntu:16.04

# Install Redis.
RUN <wget ...>\
<.......>\
<.......>

# Define mountable directories.
VOLUME ["/data"]
# Define working directory.
WORKDIR /data
# Define default command.
CMD ["redis-server", "/etc/redis/redis.conf"]
# Expose ports.
EXPOSE 6379
```

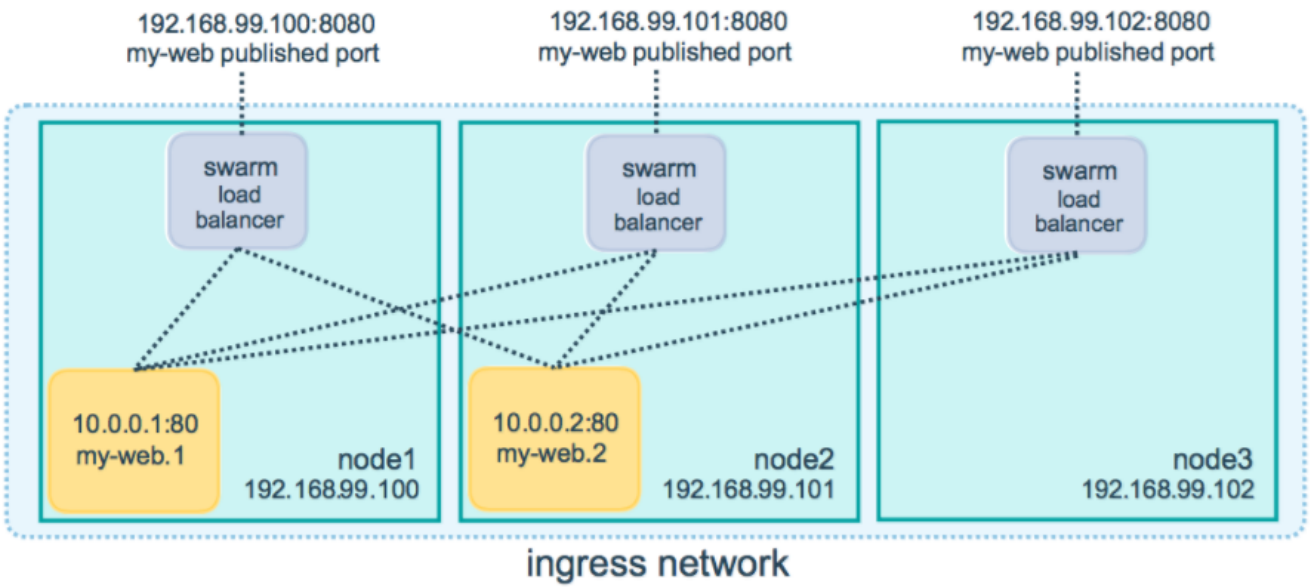
Балансировка нагрузки

docker-compose.yml

```yaml
version: "3"
services:
  web:
    # replace username/repo:tag with your name and image details
    image: username/repo:tag
    deploy:
      replicas: 5
      resources:
        limits:
          cpus: "0.1"
          memory: 50M
      restart_policy:
        condition: on-failure
    ports:
      - "80:80"
    networks:
      - webnet
networks:
  webnet:
```

Swarm-ы

- Машина, на которой запускается контейнер, становится главной
- Другие машины могут присоединяться к swarm-у и получать копию контейнера
- Docker балансирует нагрузку по машинам



© https://www.docker.com

Задание на пару

В командах по два человека оформить сетевой чат, разработанный на предыдущем занятии, в виде Docker-контейнера

- Убедиться, что при запуске клиента и сервера через Docker они могут установить соединение
- Выложить в свой репозиторий Docker-файл

Что делать

- Заполнить форму https://forms.gle/fTaEJ2YQaBHUNmnZ6 ссылкой на репозиторий
- Выложить на HwProj результаты к концу пары
- Желательно успеть показать, что всё работает
- Доделать “дома”, если не успели