

Лекция 11: Развёртывание распределённых приложений

Юрий Литвинов
y.litvinov@spbu.ru

26.05.2026

Docker

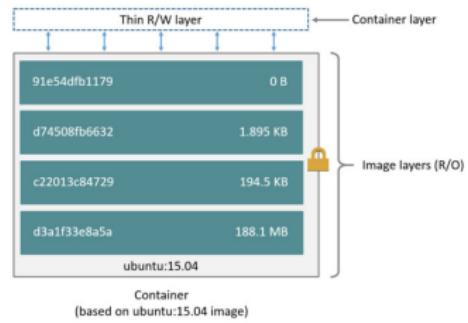
- ▶ Средство для “упаковки” приложений в изолированные контейнеры
- ▶ Что-то вроде легковесной виртуальной машины
- ▶ Широкий инструментарий: DSL для описания образов, публичный репозиторий, поддержка оркестраторами



© <https://www.docker.com>

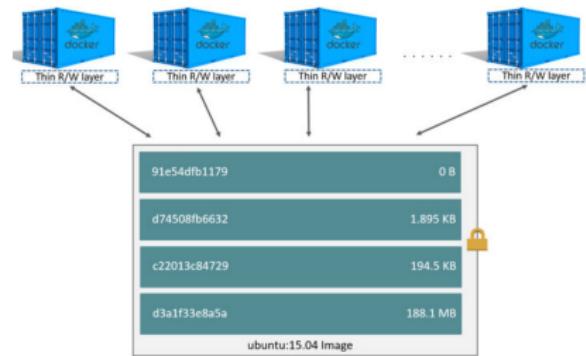
Docker Image

- ▶ 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



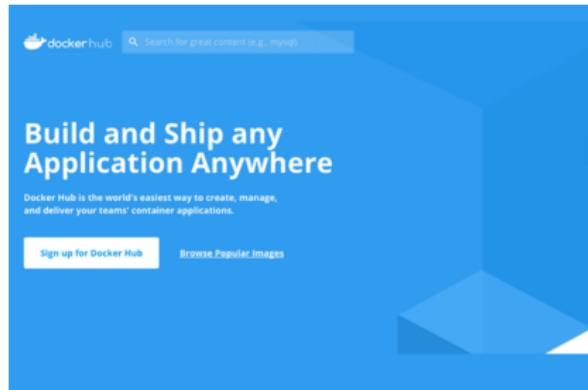
Docker Container

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



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"]
```

Двухфазная сборка

```
FROM mcr.microsoft.com/dotnet/aspnet:6.0 AS base
```

```
WORKDIR /app
```

```
EXPOSE 80
```

```
EXPOSE 443
```

```
FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build
```

```
WORKDIR /src
```

```
COPY ["ConferenceRegistration/ConferenceRegistration.csproj", "ConferenceRegistration/"]
```

```
RUN dotnet restore "ConferenceRegistration/ConferenceRegistration.csproj"
```

```
COPY ..
```

```
WORKDIR "/src/ConferenceRegistration"
```

```
RUN dotnet build "ConferenceRegistration.csproj" -c Release -o /app/build
```

```
FROM build AS publish
```

```
RUN dotnet publish "ConferenceRegistration.csproj" -c Release -o /app/publish
```

```
FROM base AS final
```

```
WORKDIR /app
```

```
COPY --from=publish /app/publish .
```

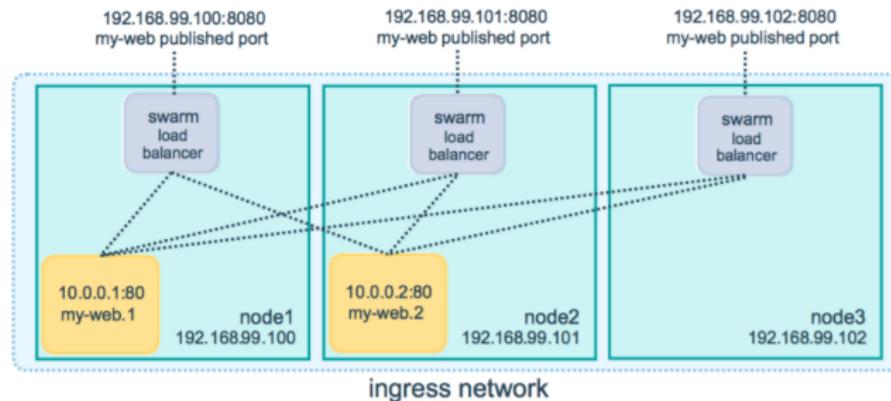
```
ENTRYPOINT ["dotnet", "ConferenceRegistration.dll"]
```

Docker Compose

```
version: "3"
services:
  web:
    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:
```

Docker Swarm

- ▶ 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>

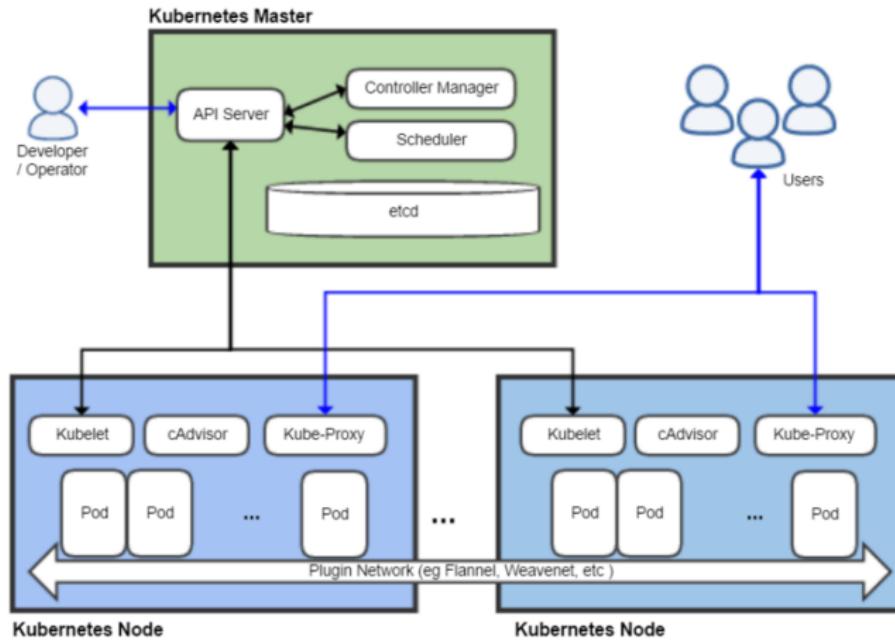
Kubernetes

- ▶ Оркестратор контейнеров
- ▶ Отвечает за раскидывание контейнеров по хостам, масштабирование, мониторинг и управление жизненным циклом
 - ▶ Сильно продвинутый Docker Compose
- ▶ Open-source, Google, Go



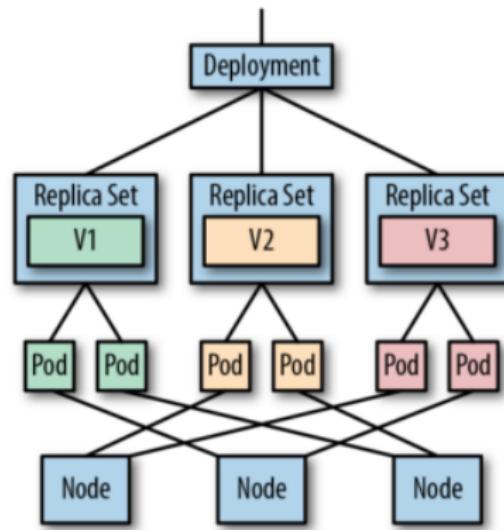
© <https://kubernetes.io/>

Архитектура Kubernetes



© <https://ru.wikipedia.org/wiki/Kubernetes>

Объекты Kubernetes



© J. Arundel, J. Domingus, Cloud Native DevOps with Kubernetes

Deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: demo
  labels:
    app: demo
spec:
  replicas: 1
  selector:
    matchLabels:
      app: demo
  template:
    metadata:
      labels:
        app: demo
    spec:
      containers:
        - name: demo
          image: cloudnativd/demo:hello
          ports:
            - containerPort: 8888
```

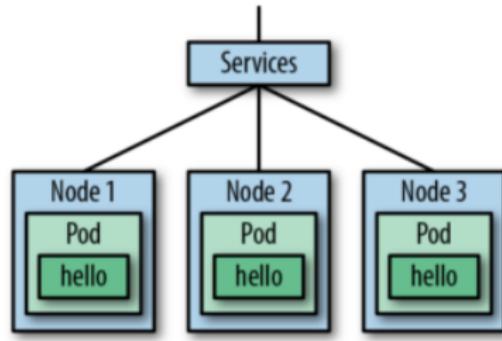
Запуск:

```
kubectl apply -f k8s/deployment.yaml
```

© J. Arundel, J. Domingus, Cloud Native DevOps with

Kubernetes

Сервисы



© J. Arundel, J. Domingus, Cloud Native DevOps with Kubernetes

Service

```
apiVersion: v1
kind: Service
metadata:
  name: demo
  labels:
    app: demo
spec:
  ports:
  - port: 9999
    protocol: TCP
    targetPort: 8888
  selector:
    app: demo
  type: ClusterIP
```

Запуск:

```
kubectl apply -f k8s/service.yaml
kubectl port-forward service/demo 9999:8888
```

© J. Arundel, J. Domingus, Cloud Native DevOps with Kubernetes

Рекомендации и техники

- ▶ Конфигурация — это код, не управляйте кластером вручную
- ▶ Мониторинг

`livenessProbe:`

`httpGet:`

`path: /healthz`

`port: 8888`

`initialDelaySeconds: 3`

`periodSeconds: 3`

- ▶ Blue/green deployment, rainbow deployment, canary deployment
 - ▶ Не используйте тэг `latest` для Docker-образов
- ▶ Используйте инструменты
 - ▶ Helm, Kubernetes Dashboard и аналоги, Prometheus, Clair, Velero, ...
- ▶ Метрики: Requests-Errors-Duration, Utilization-Saturation-Errors

Облачная инфраструктура

- ▶ Виды сервисов:
 - ▶ Infrastructure as a Service
 - ▶ Platform as a Service
 - ▶ Software as a Service
- ▶ Основные провайдеры:
 - ▶ Amazon Web Services (почти 50% рынка)
 - ▶ Microsoft Azure (порядка 10%)
 - ▶ Google Cloud
 - ▶ Всё остальное (Heroku, Yandex.Cloud, ...)

Пример: экосистема AWS

- ▶ Вычисления:
 - ▶ EC2 (Elastic Computations)
 - ▶ ECS (Elastic Container Service)
- ▶ Сеть:
 - ▶ VPC (Virtual Private Cloud)
 - ▶ ELB (Elastic Load Balancer)
 - ▶ API Gateway
- ▶ Устройства хранения:
 - ▶ EFS (Elastic File System)
 - ▶ EBS (Elastic Block Storage)
- ▶ SaaS, базы данных:
 - ▶ RDS (Relational Database Service)
 - ▶ DynamoDB
 - ▶ ElasticSearch Service

Infrastructure as Code

«The enabling idea of infrastructure as a code is that systems and devices which are used to run software can be treated as if they, themselves, are software» (Infrastructure as Code, Kief Morris)

- ▶ Платформонезависимое представление инфраструктуры
- ▶ Воспроизводимое развертывание
- ▶ Пример: Terraform

