

Homework_Lesson20_Report

1. Зарегистрировать аккаунт на kodekloud.com
2. Заинролиться на курс Docker Training Course for the Absolute Beginners
3. Пройти курс на 100%, сделать скриншот и добавить в папку домашки в качестве отчёта



This Certificate Is Proudly Presented To

Алексей Левченко

For Successfully Completing Our

Docker Training Course for the Absolute Beginner

January 21, 2025

ID: 928bdb74-8997-4bc5-9a4b-9395bdf0037f

Mumshad Mannambeth

Mumshad Mannambeth
Founder & Trainer, KodeKloud



[For Teams](#) [Dashboard](#) [Pricing](#) [Playgrounds](#) [Learning Paths](#) [Courses](#)



Course Progress

100% Complete

42/42 Lessons

[Continue Learning](#)

[Unenroll in This Course](#)

Work in a Team?

KodeKloud-For-Business gives your team the highest-tier access to everything on KodeKloud - plus exclusive content and powerful team management features.

[Find Out More](#)

`docker run nginx` - установит контейнер nginx если image нет на хосте будет искать на Docker hub

`docker run nginx:1.26.2` - установить контейнер nginx версии 1.26.2(после : префикс)

`docker run nginx sleep 5` - контейнер уснет на 5 секунд и не будет остановлен в это время если нету рабочего процесса

`-i` ---- сопоставить ввод хоста с запущенным контейнером

`-t` ---- псевдотерминал отображает вывод информации в терминале контейнера на хосте

`docker run -it ubuntu bash` - обычно используется комбинация ключей для команды `run -it` мы подключены к терминалу контейнера и находимся в интерактивном режиме можем писать команды, в терминал которые будут переданы в контейнер

`docker run -it centos bash` - выполним команду `bash` в созданном контейнере и войдем в него

`-d`

`docker run -d nginx` - запускает контейнер в отсоединённом режиме мы в него не провалимся, и он будет работать параллельно

`docker attach a043d` (id или имя контейнера) - подключиться обратно к работающему контейнеру

`-p` сопоставить порты на хосте и контейнере.

`docker run -p 80:8080 nginx`

`-v` смонтировать директорию с хоста внутрь контейнера

`docker run -v /opt/datadir:/var/lib/mysql mysql`

`-e` передать переменные среды в контейнер например ниже передается пароль `root` в `mysql`

`docker run --name some-mysql -e MYSQL_ROOT_PASSWORD=my-secret-pw -d mysql:tag`

`docker run --cpus=.5 ubuntu` - контейнер будет использовать максимум 50% cpu хоста.

`docker run --memory=100m ubuntu` - контейнер будет использовать максимум 100мб озу

`docker ps` - показывает все запущенные контейнеры и информацию по ним (Id контейнера, имя, текущее состояние)

`-a`

`docker ps -a` - покажет все запущенные и завершенные контейнеры

`docker stop my_nginx` (имя или ID контейнера) - остановить контейнер

`docker rm my_nginx` (имя или ID контейнера) - удалить остановленный или завершённый контейнер

`docker images` - отобразить все доступные images на хосту

`-q` - отобразить только идентификаторы id Docker Image

`docker rmi nginx` (имя image) - удалить image с хоста - не должно быть запущенных контейнеров на этом image

`docker rmi $(docker image -q)` - удалить все image с хоста

`docker pull nginx` (имя image) - скачать контейнер из docker hub

`docker exec my_nginx cat /etc/hosts` - выполнить команду в работающем контейнере

`docker inspect my_nginx` (имя image) - посмотреть информацию о контейнере

`docker container prune` - удалить все остановленные контейнеры

`-f` без подтверждения

`docker logs my_nginx` (имя или ID контейнера) - посмотреть логи контейнера

`docker port f544545edd60` (имя или ID контейнера) - просмотреть перенаправленные порты в контейнере

Network

при установке докер создается 3 сети

`bridge` - сеть по умолчанию куда подключается контейнер (обычно получают адрес 172.17.0.2 ...)

`null` - не подключен к сети

`host` - сопоставляется с хостом

`--network=host`

`docker run Ubuntu -- network=host` - подключить контейнер к сети host

создать сеть

```
docker network create \  
  --driver bridge \  
  --subnet 182.18.0.0/16 \  
  --gateway 182.18.0.1 \  
  custom-isolated-network
```

`docker network ls` - посмотреть список сетей докер

Storage

/var/lib/docker - файлы докера

aufs -

container- все файлы связанные с контейнером

image- все файлы связанные с image

volume - все тома созданные контейнерами

docker volume create data_volume создать volume

docker run -v data_volume:/var/lib/mysql mysql - подключить volume

docker run -v /data/mysql:/var/lib/mysql mysql - подключить подключить каталог вместо стандартного volume

-v старый стиль

--mount современным и предпочтительным способом монтирования нужно указать type, source, target

docker run --mount type=bind,source=/data/mysql,target=/var/lib/mysql mysql

Dockerfile

FROM Ubuntu - на основе какого образа у нас будет создаваться контейнер

RUN apt update - выполнить определенную команду

RUN apt install python -y

RUN pip install flask

RUN pip install flask-mysql

COPY . /opt/source-code - копируем приложение в контейнер в нашем случае оно находится в текущей папке и копируется в контейнер по пути /opt/source-code

ENTRYPOINT FLASK_APP=/opt/source-code/app.py flask run - выполняет команду после того как контейнер запущен

CMD - какая команда будут выполняться на контейнере

CMD command param1

CMD ["command", "param1"]

если в формате json массива то первый параметр должен быть исполняемый файл и параметры последующие параметры.

Если мы хотим запускать при старте контейнера и указывать из командной строки параметры стоит использовать ENTRYPOINT

В данном примере мы задаем команду ENTRYPOINT ["sleep"] и если мы не передадим параметр таймера из консоли CMD применит 5 по умолчанию
Так удобно их комбинировать
ENTRYPOINT ["sleep"]
CMD ["5"]

Сбилдить докерфайл

```
docker build -t alex1436183/my-custom-app Dockerfile
```

Загрузить в Dockerhub

```
docker push alex1436183/my-custom-app
```

Посмотреть информацию о слоях image. Мы увидим все слои и результаты выполнения каждой задачи.

```
docker history alex1436183d/my-custom-app
```

Все шаги кешируются и в случае изменений в Dockerfile предыдущие слои будут взяты из кеша сборка произойдет быстрее.

```
docker login - войти в свой аккаунт что бы загрузить image на dockerhub
```

Docker image

FROM node:lts-alpine - на основе какого образа у нас будет создаваться контейнер
WORKDIR /app рабочая директория
COPY . . копируем
RUN yarn install --production команда
CMD команда
EXPOSE

Docker compose

```
version: "3"  
services:  
  redis:  
    image: redis  
  
  db:  
    image: postgres:9.4
```

vote:

image: voting-app

ports:

- "5000:80"

links:

- redis

result:

image: result-app

ports:

- "5001:80" # Изменен порт, чтобы избежать конфликта с vote

links:

- db

worker:

image: worker

links:

- db

- redis

depends_on:

- redis

docker-compose up -d

-d отсоединенный режим

можно заменить строку что бы вместо поиска image мы сбилдили новый image

image: voting-app на build: ./vote