Docker для начинающих — Полный курс 2021

Что такое Docker

Docker — технология для создания и управления контейнерами.

Мы оборачиваем какой то код или приложение в контейнеры для того, чтобы он нам гарантировал одинаковое поведение в разных окружениях. Мы можем просто брать докер контейнеры и запускать их где угодно, где есть докер. Нам не важно, что это будет за ОС, его версия. Все поведение будет зафиксировано в контейнере.

Базовая информация

`docker` - какие вообще команды есть в докере

`docker version` - узнаем версию докера

Быстрый пример с Python

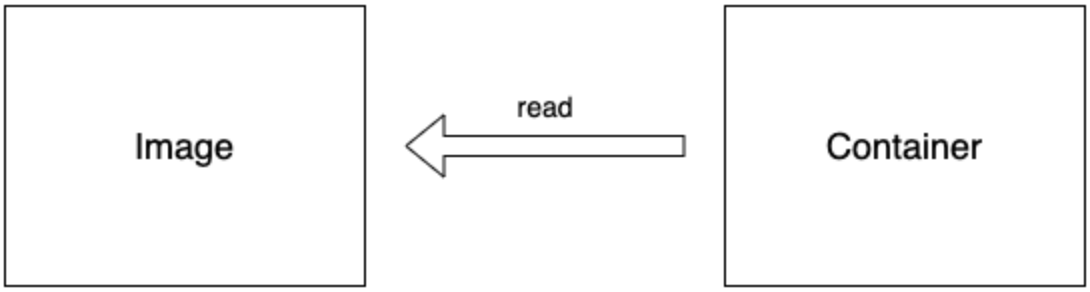
```
print('Hello Python!')
```

```
FROM python

WORKDIR /app

COPY . /app

CMD ["python", "index.py"]
```

1. `docker build .`
2. `docker image ls`
3. `docker run IMAGE_ID`

Образы и контейнеры (Images & Containers)

Containers & Images



Containers - запускаются на основе образов

Images - шаблоны, только для чтения для создания контейнеров

Качаем ноду

```
docker pull node
```

Она качается отсюда: https://hub.docker.com/_/node

Смотрим, что скачалось

```
docker images
```

Запускаем контейнер с нодой

```
docker run node
```

```
docker ps
```

Смотрим на параметры

```
docker ps –help
```

```
docker ps -a
```

Запускаем в интерактивном режиме и сравниваем версии

```
docker run -it node
```

> process.version

```
node -v
```

Удаляем контейнер

```
docker ps -a
```

```
docker rm CONTAINER_ID
```

Практика Node приложения

Качаем приложение с Git: https://github.com/vladilenm/logs-app

Создаем **Dockerfile**

```
FROM node # с какого image хотим сделать свой

WORKDIR /app # контекст проекта

COPY . . # копируем из локального проекта

EXPOSE 3000 # какой порт запускается

RUN npm install # запускаем команду когда собирается образ

CMD ["node", "app.js"] # запускаем команду, когда запускается образ
```

Создаем свой образ:

`docker build .`

`docker image ls`

`docker run IMAGE_ID`

`docker ps -a`

`docker stop CONTAINER_ID`

`docker run -p -d 3000:3000 IMAGE_ID`

Открываем localhost:3000

Изменяем код в приложении и заного строим образ

Оптимизация докера:

```
FROM node

WORKDIR /app

COPY package.json /app

RUN npm install

COPY . .

EXPOSE 3000

CMD ["node", "app.js"]
```

Основные команды

`docker stop CONTAINER_ID` - останавливает контенер

`docker run -p 3000:3000 IMAGE_ID`

`docker attach CONTAINER_ID` - присоединяется к контейнеру

`docker start` - запускает существующий контейнер

`docker images` - список образов

`docker run -p 3000:3000 -d –rm –name nodeapp IMAGE_ID` - запускает и удаляет контейнер с именем

`docker build -t nodeapp:latest .` - создает образ с именем

`docker image inspect IMAGE` - информация по образу

`docker logs CONTAINER` - смотрим, что происходит в контейнере

`docker rmi IMAGE` - удаляем образвы

`docker rm CONTAINER_IDS` - удаляем контейнеры

`docker container prune` - удаляем все неиспользуемые контейнеры

Деплой

Заходим в docker.com

`docker tag OLD_NAME NEW_NAME` - переименовывает образ

`docker push REPO_NAME` - Заливает образ

`docker pull` - забирает образ

Добавляем .dockerignore

```
node_modules
Dockerfile
.git
.idea
```

ENV переменные

```
ENV PORT 3000

EXPOSE $PORT
```

Задаем переменные из консоли

`docker run -p 3000:80 -d --rm --name logsapp -e PORT=80 logsapp:env`

Или через файл

```
PORT=3000
```

`docker run -p 3000:80 -d --rm --name logsapp --env-file ./.env logsapp:env`

Тома (Volumes)

Это просто папка на локальной машине, которая может монтироваться в докер контейнер. Служит для того, чтобы данные существовали вне зависимости от контейнеров.

Например данные для базы данных, или исходный код самого приложения.

```
VOLUME ["/app/data"] # Добавляем в докер
```

Собираем образ

`docker build -t logsapp:volumes .`

`docker run -p 3000:3000 -d –rm –name logsapp -v logs:/app/data logsapp:volumes`

`docker volume ls` - смотрим список

`docker volume inspect logs`

`docker volume create VOLUME`

`docker volume prune`

`docker volume rm VOLUME`

Монтирование каталога (Bind Mount)

Это нужно для разработки

Если собрать контейнер, то он не меняется, когда мы меняем исходники в редакторе

Добавляем еще один volume

```
docker run -p 3000:3000 -d --rm --name logsapp -v logs:/app/data -v /app/node_modules -v "/Users/vladilen/WebstormProjects/express-sample-app:/app" logsapp:volumes
```

Или с сокращением

```
docker run -p 3000:3000 -d --rm --name logsapp -v logs:/app/data -v /app/node_modules -v $(pwd):/app logsapp:volumes
```

Команды

`docker volume ls` - список

`docker stop CONTAINER` - удаляем контейнер

`docker volume ls` - удалились анонимные volumes

Deployment

На локальной машине заливаем образ в Docker Hub:

```
docker tag LOCAL_IMG vladilenm/nodeapp
```

```
docker push vladilenm/nodeapp
```

Я взял сервер на vscale.io . Можно использовать любой удобный VPS

> 💡 **Как создать SSH ключ**
>
> ssh-keygen -t rsa
> pbcopy < ~/.ssh/id_rsa.pub

На VPS

```
docker pull vladilenm/nodeapp
```

```
docker run -d -p 80:3000 --name nodeapp --rm vladilenm/nodeapp
```

Переходим по адресу:

Открываем в браузере IP

Практикум

Как продолжение работы с докером, рекомендую познакомиться с моим практикумом.

В нем вы узнаете, как работать с несколькими контейнерами на примере MERN приложения (Mongo Express React Node).

В рамках практикума я покажу как создать dev & prod сборку, как пользоваться docker-compose и многое другое. Все подробности по ссылке ниже:

https://clc.to/docker