

# Dockerfile

# Dockerfile базис

- Docker образы собираются из базового образа
- Базовые образы сделаны, используя инструкции:
  - Запуск команд
  - Добавить файл в директорию
  - Создать переменную среды
  - Какой процесс запустить при запуске контейнера

Dockerfile X

Dockerfile > FROM

```
1 FROM node:13-alpine
```

```
2
```

```
3 ENV MONGO_DB_USERNAME=admin \  
4     MONGO_DB_PWD=password
```

```
5
```

```
6 RUN mkdir -p /home/app
```

```
7
```

```
8 COPY ./app /home/app
```

```
9
```

```
10 # set default dir so that next commands executes in /home/app dir
```

```
11 WORKDIR /home/app
```

```
12
```

```
13 # will execute npm install in /home/app because of WORKDIR
```

```
14 RUN npm install
```

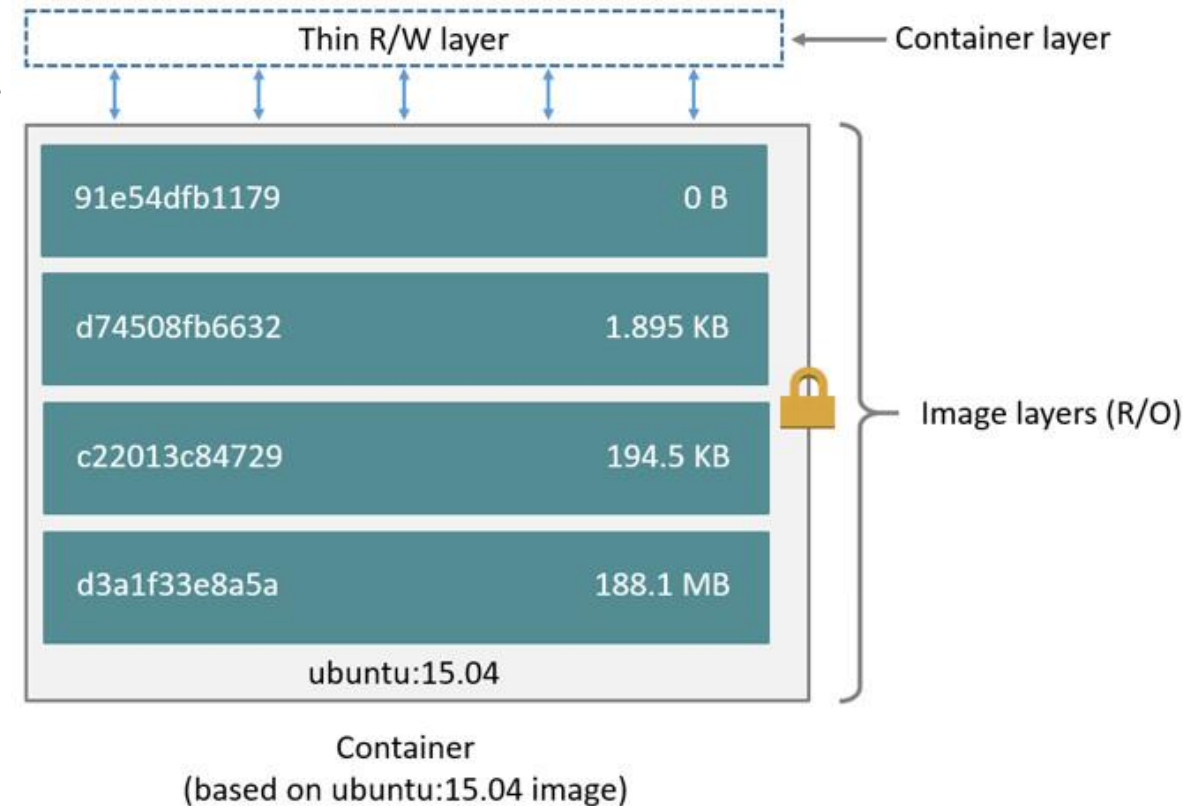
```
15
```

```
16 # no need for /home/app/server.js because of WORKDIR
```

```
17 CMD ["node", "server.js"]
```

# Docker образы

- Контейнер основывается на одном образе
- Каждый образ основывается на многоуровневой файловой системе
- Когда вы меняете образ, новый слой добавляется



# FROM

- задаёт родительский (главный) образ. На каком Dockerfile основывается текущий образ

FROM ubuntu:latest

# env

- Полезно для создания новых переменных сред

```
ENV JAVA_HOME /etc/bin/jdk/default-jdk
```

```
ENV TOMCAT_MAJOR 8
```

# RUN

- запускает команды, создаёт слой образа. Используется для установки пакетов и библиотек внутри контейнера

`RUN apt update && apt install -y default-jdk git`

# ADD\COPY

- COPY — копирует файлы и директории в контейнер
- ADD — делает всё то же, что и инструкция COPY. Но ещё может распаковывать локальные .tar файлы и загружать файлы с URL

COPY /home/app\*.jar # Добавляет файлы, которые начинаются с app и заканчиваются jar

COPY /home/val?e # Заменяет ? На одну букву

# EXPOSE

- EXPOSE говорит Docker, что приложение в контейнере должно использовать определенный порт в контейнере. Это не означает, что вы можете автоматически получать доступ к сервису, запущенному на порту контейнера

EXPOSE 8080



# WORKDIR

- С помощью WORKDIR можно установить рабочую директорию, откуда будут запускаться команды

WORKDIR /app

# CMD

- Указывает команду и аргументы для выполнения внутри контейнера. Параметры могут быть переопределены. Использоваться может только одна инструкция CMD

CMD[ 'java', '-jar', 'app\*.jar']

# Сборка образа

- Dockerfile содержит набор инструкций для того чтобы собрать образ
- Docker-daemon делает сборочный процесс. Весь контекст текущей директории передаётся демону. Можно уменьшить оверхед путём `.dockerignore` .
- Каждая инструкция создаёт новый слой образа
- Docker по максимуму использует кеш при сборке

`Docker build -t ex/ex .`

# Публикация образа

- Вы можете опубликовать ваш образ в Docker репозиторий. Образы могут быть скачаны с другой машины

```
Docker tag public_container:v2 private.repo/test:v2
```

```
docker push private.repo/test:v2
```

```
Docker pull private.repo/test:v2
```

# Пример

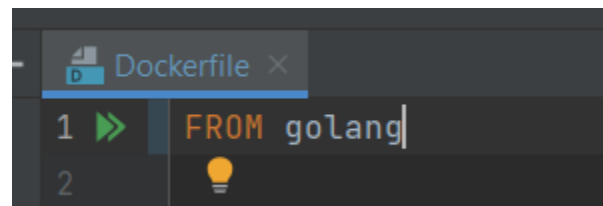
- Git: <https://git.miem.hse.ru/rshaniiazov1/devops-summer-school-example-java/-/tree/master/demo>
- Команды:
  - `docker run -it -v "$(pwd):/home/gradle" gradle:jdk17 docker build -t ex/ex:1 .`
  - Создать сеть lesson
  - `docker run -d -p 8081:8080 --name first --network lesson ex/ex:1`
  - `docker run -d -p 8080:8080 --name second --network lesson --env TARGET_URL=http://first:8080 ex/ex:1`

# Volumes

- Docker volume — это просто папка хоста, примонтированная к файловой системе контейнера.
- Может быть расшарена между контейнерами
- При удалении контейнера, не удаляет volume

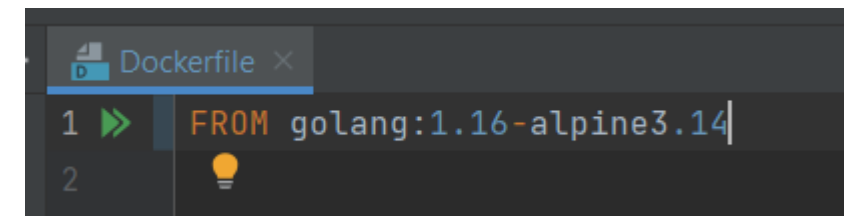
# Хороший тон

- Использовать `.dockerignore` для того, чтобы избежать подтягивания ненужных файлов
- Не используйте `apt update` в виде отдельной команды (связано с багами контейнера)
- Не устанавливайте лишних пакетов
- Всегда используйте тэг в `From` (игнорируйте `:latest`)



```
Dockerfile x
1 >> FROM golang|
2
```

This screenshot shows a code editor window titled 'Dockerfile'. The first line of code is 'FROM golang', which is highlighted with a green arrow icon. A lightbulb icon is visible in the margin next to the second line.



```
Dockerfile x
1 >> FROM golang:1.16-alpine3.14|
2
```

This screenshot shows a code editor window titled 'Dockerfile'. The first line of code is 'FROM golang:1.16-alpine3.14', which is highlighted with a green arrow icon. A lightbulb icon is visible in the margin next to the second line.

- Не используйте `run + commit`, используйте `Dockerfile`
- Не устанавливайте `ssh` сервер в контейнер
- Один процесс приложения на контейнер



# SDK отдельно, исполняемые файлы отдельно

```
# syntax=docker/dockerfile:1
FROM golang:1.16-alpine
WORKDIR /app
COPY go.mod ./
RUN go mod download
COPY *.go ./
RUN go build -o app
EXPOSE 8080
# source
https://docs.docker.com/language/golang/build-
images/
CMD [ "/app/app" ]
```

```
# syntax=docker/dockerfile:1
FROM golang:1.16-alpine as builder
WORKDIR /app
COPY go.mod ./
RUN go mod download
COPY *.go ./
RUN go build -o app

FROM frovlad/alpine-glibc:alpine-3.16
WORKDIR /app
COPY --from=builder /app/app /app/app
EXPOSE 8080
CMD [ "/app/app" ]
```

NAME		TAG	IMAGE ID	CREATED ↓	SIZE
example/two_image	IN USE	latest	c92385a925d8	5 minutes ago	23.58 MB
example/one_image	IN USE	latest	0cfe00baec6d	11 minutes ago	308.06 MB

# Больше хорошего тона

- [https://docs.docker.com/develop/develop-images/dockerfile\\_best-practices/](https://docs.docker.com/develop/develop-images/dockerfile_best-practices/)