PHASE 2
WEEK 1

DAY 4

ELBRUS/BOOTCAMP

План

1. Контейнеризация
2. Docker
3. Создание образа Docker

Контейнеры

Контейнеризация

Контейнеризация — один из видов изоляции приложений.

Контейнеризация позволяет запускать несколько приложений на одной машине так, чтобы они не могли друг с другом взаимодействовать.

Контейнеризация: зачем?

- Решает проблему повторяемого окружения
- Решает проблему конфликта глобальных зависимостей
- Решает проблему контроля потребляемых приложением ресурсов

Docker

Docker

Docker — платформа для разработки, доставки и запуска контейнерных приложений.

Docker позволяет создавать контейнеры, автоматизировать их запуск и развертывание, управляет жизненным циклом. Он позволяет запускать множество контейнеров на одной хост-машине.

Docker: установка

Шаги установки под каждую ОС детально описаны здесь:

https://docs.docker.com/engine/install/

Docker: контейнеры

Docker Containers, контейнер Docker — конкретное приложение, запущенное в Docker.

Любой контейнер запускается на основе образа. Каждый контейнер изолирован от любого другого.

Docker: образы

Docker Image, образ Docker — заранее подготовленное приложение в формате Docker.

На основе образа строится контейнер. Нет ограничения на количество контейнеров, созданных из одного образа.

Docker: образы и контейнеры

Можно представить образы как классы, а контейнеры — как экземпляры класса.

Несколько контейнеров, запущенных из одного образа, никак друг с другом не связаны.

DockerHub

DockerHub — публичное хранилище готовых образов. По умолчанию Docker ищет образы в Hub.

Большинство приложений, например базы данных, уже опубликованы в DockerHub и могут быть легко запущены.

Команда docker run

Для запуска контейнера <container_name> на основе образа <image>:

```
docker run -d --name <container_name> -p
<external_port>:<inner_port> <image>
```

Команда docker ps

Чтобы просмотреть список запущенных контейнеров:

```
docker ps
```

Чтобы просмотреть список всех контейнеров:

```
docker ps -a
```

Команда docker stop

Для остановки контейнера по имени или ID:

```
docker stop <container_name_or_id>
```

Команда docker rm

Для удаления контейнера(-ов):

```
docker rm <container_name_or_id>
```

Для удаления ВСЕХ остановленных контейнеров:

```
docker container prune
```

Dockerfile

Создание собственных образов

`docker build` — команда для построения собственных образов на основе файла Dockerfile и папки с файлами приложения.

```
docker build .
```

Построить образ на основе Dockerfile, который лежит в текущей папке.

Dockerfile

`Dockerfile` — специальный файл, описывающий шаги, выполняемые во время “сборки” образа.

В этом файле описывается установка зависимостей, удаление ненужных файлов и создание нужных, установка команды по умолчанию для запуска контейнера.

Dockerfile, example code

```
FROM node:16

ENV NODE_ENV production

COPY package.json .
COPY package-lock.json .
RUN npm ci

COPY . .
CMD ["npm", "start"]
```

.dockerignore

Файл со списком файлов и директорий, которые Docker должен игнорировать

.dockerignore, example code

```
# Используйте # в начале строки для обозначения комментария в файле
.dockerignore

# Исключать файлы и каталоги, имена которых начинаются с temp любого
непосредственного подкаталога корня
*/temp*

# Исключайте файлы и каталоги, начинающиеся с temp, из любого
подкаталога, который находится на два уровня ниже корня
*/*/temp*

# Исключить файлы и каталоги в корневом каталоге, имена которых являются
односимвольным расширением temp. Например, /tempa и /tempb исключены
temp?

*/node_modules
*/.git
```