***Dockerfile*** – специальный файл, описывающий настройки для создания *docker-image*. Команда *“docker build”* собирает из него *docker-image*. Сборка образа происходит послойно и кэшируется после каждого изменения.

***Docker-image*** – это образ-шаблон, который содержит все необходимое для автономной работы контейнера. В образ может быть упакована ОС, сервер, интерпретатор языка, виртуальное окружение проекта, файлы приложения.

***Docker*-контейнер** – это окружение, которое создается из образа, как по шаблону. Он запускается командой *“docker run имя\_образа”*. Например, *“docker run hello-world”*.

Рассмотрим процесс создания и запуска *docker*-контейнера для *react* приложения:

1. Добавим в корень проекта текстовый файл без расширения и назовем его .d*ockerignore*. Добавим в него файлы, которые не будут использоваться docker при создании образа:

*node\_modules  
Dockerfile  
.env  
.dockerignore  
.DS\_Store*

1. Добавим в корень проекта текстовый файл без расширения и назовем его *.env*. Добавим в него настройки для переменных сред:

*VITE\_API\_BASE\_URL=http://localhost:8080*

1. Для работы на локальном хосте в файле *package.json* нужно добавить ***--host*** в поле

*“scripts”:“dev”: “vite --host”*

1. Добавим в корень проекта текстовый файл без расширения и назовем его *Dockerfile*. Он будет служить схемой, по которой Docker будет собирать образ для контейнеров.
2. Добавим в *Dockerfile* инструкции для сборки. Каждая инструкция пишется заглавными буквами, а их выполнение происходит последовательно.

*FROM node:19-alpine // ОС Linux с установленной node*

*ARG api\_base\_url // принимаем извне это значение  
WORKDIR /app // создадим целевую рабочую директорию  
COPY package\*.json . // скопируем json файлы с зависимостями в рабочую директорию  
RUN npm i --silent // скачаем и установим все библиотеки-зависимости  
COPY . . // копируем весь код проекта, кроме запрещенного .dockerignore  
RUN echo "VITE\_API\_BASE\_URL=${ api\_base\_url }” > .env // переопределим переменные среды  
EXPOSE 5173 // открыть порт 5173  
CMD ["npm", "run", "dev"] // запустить фронтенд*

Команда ***FROM <image>:<tag>*** определяет базовый образ, на основе которого создается локальный образ. Необязательный параметр ***tag*** указывает конкретную версию образа. По умолчанию используется версия с тегом ***latest***. Подробнее о тегах можно почитать [тут](https://hub.docker.com/_/amazoncorretto?tab=tags).

Команда ***ARG*** позволяет задать значение для аргумента сразу при сборке образа.

Команда ***COPY <локальная директория> <путь внутри контейнера>*** копирует файлы и директории из указанной локальной директории в директорию внутри контейнера. Если целевой директории не существует, то она создается.

Команда ***WORKDIR <директория>*** устанавливает рабочую директорию в образе.

1. Соберем собственный образ *dock\_image*.

*docker build . -t straigt/straigt-react --build-arg=http:localhost:8088*

Команда ***build*** запускает сборку образа по инструкциям из *Dockerfile*.

Ключ ***-t*** позволяет задать имя образу.

***Точка*** внутри команды – это путь до *Dockerfile*, на основе которого производится сборка.

Ключ ***--build-arg*** позволяет задать аргумент образу.

1. Загрузим docker-образ в DockerHub:

*docker push straigt/straigt-react*

1. Запустим собственный контейнер.

*docker run --name dock\_container -p 3000:5173 dock\_image*

Команда ***run*** запускает контейнер.

Ключ ***--name*** позволяет задать имя контейнеру. В случае его отсутствия, *Docker* сгенерирует его самостоятельно.

Ключ ***-p <внешний порт>:<внутренний порт приложения>*** позволяет настроить порты.

Ключ ***-d*** позволяет не занимать текущий процесс консоли запущенным приложением.

Ключ ***--rm*** позволяет удалить контейнер при выходе.

***dock\_image*** – образ, из которого будет запущен контейнер.

Для управления взаимодействием нескольких контейнеров используется ***Docker Compose***. Например, в одном контейнере работает сервис, а в другом – база данных.

Инструкции по развертыванию проекта в нескольких контейнерах пишут в файле ***docker-compose.yml***. Например:

*# версия Docker Compose*

*version: '3.8'*

*# имена и описания контейнеров, которые должны быть развёрнуты*

*services:*

*# описание контейнера straigt-react*

*straigt-react:*

*# имя контейнера*

*container\_name: straigt -react*

*# ЛИБО загрузить образ из DockerHub и запустить контейнер*

*image: straigt/straigt-react*

*# ЛИБО создать образ на основе локального dockerfile и запустить контейнер*

*build:*

*# путь до dockerfile*

*context: frontend/react*

*# передать аргументы для переменных сред внутрь dockerfile*

*args:*

*api\_base\_url:* [*http://localhost:8088*](http://localhost:8088)

*ports:*

*- "3000:5173"*

*# "зависит от"*

*depends\_on:*

*- straigt-api*

*# перезапускать если упадет БД*

*restart: unless-stopped*

Команда ***services*** задает список имен и описаний контейнеров, которые должны быть развернуты. Имена могут быть любыми.

Команда ***depends\_on*** определяет, после какого контейнера должен быть запущен описываемый.

Команды ***docker-compose up*** и ***docker-compose stop*** запускает и останавливает проект соответственно.

Команды ***docker-compose down*** и ***docker-compose pull*** останавливает и загружает актуальную версию докер-образа проекта из *DockerHub*.

Подробнее про модульные приложения можно почитать [тут](https://spring.io/guides/gs/multi-module/).