Docker

Здесь описан краткий план по демонстрации работы с docker контейнерами и инфраструктурой докера

1. Docker-hub. Загрузка и запуск простейшего, готового контейнера (docker-library/hello-world)

```
docker pull hello-world
docker image ls
docker run hello-world
docker container ls -a
docker rm "<CONTAINER_ID>"
```

2. Docker-hub. Пример с контейнером ubuntu. Первый контейнер, который можно применять на практике. Рассказ про выполнение команд внутри контейнера (на примере ls). Обзор флагов запуска (--rm, -i, -t, -d, -p, -v).

```
docker pull ubuntu
docker image ls
docker run ubuntu ls
docker rm "<CONTAINER_ID>"
# Запуск с удалением по завершению
docker run --rm ubuntu ls
# Запуск с подключением к псевдоконсоли
docker run -i -t ubuntu bash
```

3. Демонстрация изменений временной файловой системы внутри контейнера. Уничтожение пространства процесса

```
# Запуск с комбинацией -i -t позволяет выполнять detach комбинацией ^P^Q
внутри контейнера
docker run -i -t ubuntu bash
>> echo "Hello World" > ~/test.txt
>> cat ~/test.txt
# Если был указан флаг --rm то контейнер будет удалён вместе с его файловой
системой
# При отсутсвии этого флага уничтожается только пространство процессов, а
файловая система остаётся нетронутой
>> exit
docker start -i "<CONTAINER_ID>"
# Видно, что состояние файловой системы не поменялось
>> cat ~/test.txt
```

4. Демонстрация работы с пространством процессов. Отсоединение от контейнера. Фоновая работа.

```
>> apt update && apt install -y tmux
# Создаём фоновое приложение
>> tmux new -s run
>> while true; do echo >> test.txt; sleep 1; done;
>> ^B D
>> exit
```

```
docker start -i "<CONTAINER_ID>"
# Видим, что выход из контейнера таким образом действительно уничтожает
пространство процессов
>> tmux ls
# Создаём фоновое приложение ещё раз
>> tmux new -s run
>> while true; do echo >> test.txt; sleep 1; done;
>> ^B D
# Отсоединяемся от контейнера
>> ^P ^Q
docker start -i "<CONTAINER_ID>"
# Видим, что пространство процессов остальнось нетронутым
>> tmux ls
>> exit

# Для запуска в фоне можно использовать флаг -d
docker run --rm -d ubuntu bash -c "while true; do echo '0'; sleep 1; done;"
```

5. Сборка контейнеров из DockerFile

```
# Для сборки можно использовать DockerFile
docker build -t flask_server .
# Можно пробрасывать порты (-p) и директории (-v)
docker run -p 5000:5000 -v "$PWD/FlaskExample/data:/root/FlaskExample/data"
--rm -i flask_server
```