Семинар № 4
Dockerfiles

https://docs.docker.com/engine/reference/builder/ **- дока**

1. Блок 3. Задача 1

Задание:

Предлагается повторить пример, отраженный на лекции. Необходимо самостоятельно создать образ, используя докерфайл:

```
FROM ubuntu:22.10
RUN apt-get update && \
    apt-get install -y cowsay && \
    ln -s /usr/games/cowsay /usr/bin/cowsay && \
    rm -rf /var/lib/apt/lists/*
CMD ["cowsay"]
```

Запустить его и убедиться, что все работает.

Пример решения:

Это примерный вариант решения, главное – чтобы логика была верной.

```
docker build -t cowsaytest .
docker run cowsaytest cowsay "GeekBrains"
```

2. Блок 4. Задача 2

Задание:

Теперь, основываясь на докерфайле, полученном на предыдущем задании, давайте проводить опыты. Посмотрите на практике на слои, изменения их и как эти изменения будут влиять на сборку контейнеров.

Создать файл **example.txt** и положить его рядом с докерфайлом.

Наполните файл простыми данными (любыми) и сохраните.

Итак. Вот вам первый докерфайл:

```
FROM ubuntu:22.10
```

```
RUN apt-get update && \
    apt-get install -y cowsay && \
    ln -s /usr/games/cowsay /usr/bin/cowsay && \
    rm -rf /var/lib/apt/lists/*
COPY example.txt /
CMD [“cowsay”]
```

Соберите из него образ. Запускать его не нужно.
Теперь измените файл **example.txt** и соберите Докерфайл заново.
Операцию можно повторить дважды для понимания, что происходит.

И второй докерфайл:

```
FROM ubuntu:22.10
COPY example.txt /
RUN apt-get update && \
    apt-get install -y cowsay && \
    ln -s /usr/games/cowsay /usr/bin/cowsay && \
    rm -rf /var/lib/apt/lists/*
CMD [“cowsay”]
```

Также необходимо собрать образ из докерфайла. После первой сборки необходимо изменить файл **example.txt** и повторить сборку.

3. Блок 5. Задача 3

Задание:

В этом задании студентам предлагается создать образ и запустить его. Цель задания - дать студентам понимание, что как только CMD будет выполнен, то работа контейнера прекратиться. До остановки можно повзаимодействовать с контейнером и например выполнить docker exec -it xxxxxx bash

Есть два докерфайла. Первый соберите как timer600:

```
FROM ubuntu:22.10
COPY example.txt /
CMD sleep 600
```

Второй как timer5:

```
FROM ubuntu:22.10
COPY example.txt /
CMD sleep 5
```

Сначала скомпилировать первый. Запустить из образа контейнер в фоновом режиме (почитайте про флаг -d):

```
docker run --name timer600_cont -d timer600
```

После запуска первого контейнера в фоновом режиме, войти в него, используя команду:

```
docker exec -it timer600_cont bash
сделать внутри ps aux
```

4. Блок 8. Задача 4

Задание:

Поэкспериментируйте с CMD и ENTRYPOINT.

Контейнеры в докере не виртуалки. Они не для запуска ОС, а для запуска приложений внутри. Т.е. как только приложение отработает, контейнер завершится.

Например если мы сделаем просто

docker run ubuntu

Мы можем определить команду (свое приложение), которое будет запускаться

docker run ubuntu sleep 5

Чтобы каждый раз не писать sleep 5 создадим свой образ timer

FROM ubuntu

CMD [“sleep”, “5”]

Зупустим контейнер и все ок, но если сделаем

docker run timer sleep 10

- с каким параметром отработает 5 или 10 ? Проверьте себя.

Например нам теперь хочется менять параметр у sleep и не указывать каждый раз 10 20 40 и пр. На помощь приходит ENTYPOINT (точка входа)

меняем CMD на ENTRYPOINT[“sleep”]

И теперь можно запускать так

docker run timer 10 - параметр упадет внутрь и вызовется sleep 10

но если запустить без параметров, то будет ошибка, вот тут нам и понадобится CMD после ENTRYPOINT , который отработает по дефолту, если никаких параметров не будет. Попробуйте такой докерфайл

FROM ubuntu

ENTRYPOINT ["sleep"]

CMD ["4"]

тогда после запуска без параметров ошибки не будет и sleep отработает 5 секунд.

5. Блок 8. Задача 5

Задание:

В этом задании студентам предлагается собрать образ по выданным докерфайлам и открыть порты для доступа в контейнер извне

Заострить внимание студентов!! В данном задании демонстрируется работа инструкции EXPOSE.

Собственно, докерфайл:

```
FROM ubuntu:22.10
RUN apt-get update && \
    apt-get install -y nginx && \
    ln -s /usr/games/cowsay /usr/bin/cowsay && \
    rm -rf /var/lib/apt/lists/*
RUN echo 'Hi, I am in your container' \
        >/usr/share/nginx/html/index.html

EXPOSE 80
```

Попытаться запустить из образа контейнер и получить информацию утилитой curl.

6. Блок 10. Домашнее задание Условие:

Формат сдачи ДЗ: предоставить доказательства выполнения задания посредством ссылки на google-документ с правами на комментирование/редактирование.

Результатом работы будет: текст объяснения, логи выполнения, история команд и скриншоты (важно придерживаться такой последовательности).

В названии работы должны быть указаны ФИ, номер группы и номер урока.

Задание: необходимо собрать образ и запустить из него контейнер.

1) основой образа должна быть alpine
2) установить необходимо mariaDB
3) уменьшить размер образа (способ обсуждался на лекции)
4) необходимо открыть порт для коммуникации с другими сущностями
5) для проверки решения необходимо подключить к такому контейнеру phpmyadmin (нужно, чтобы в нем вы увидели данные из вашей БД)
6) необходимо смонтировать внешнюю папку для хранения данных БД вне контейнера