## Задача 1 (обязательно)

### 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” |

## Задача 2 (обязательно)

### Задание

Для начала я вас попрошу создать файл **example.txt** и положить его рядом с докерфайлом.

Наполните файл простыми данными (любыми) и сохраните.

Итак. Вот 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:

|  |
| --- |
| 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. Также проделать несколько раз для понимания, что происходит.

## Задача 3 (обязательно)

### Задание

Необходимо понять в чем разница между сборками и выяснить что происходит с контейнерами на основе двух контейнеров.Вот два докерфайла. Первый:

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

Второй:

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

Сначала скомпилировать первый. Запустить из образа контейнер в фоновом режиме

|  |
| --- |
| docker run -n testcontainer -d testcontainer |

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

|  |
| --- |
| docker run -it testcontainer bash |

Нужно найти отличия и догадаться почему в первом случае у вас получается входить в контейнер без проблем и работать с ним, а во втором - нет.

|  |
| --- |
| vi Dockerfile  docker build -t testcontainer .  docker ps -a #тут увидеть, что контейнер существует и запущен  docker run -n testcontainer -d testcontainer #вход в контейнер будет без проблем тк система запущена и работает, докер не убивает контейнер  docker build -t testcontainer . #пересборка докерфайла с малым временем ожидания  docker ps -a #тут увидеть, что контейнер почти сразу убивается системой |

## Задача 4 (обязательно)

### Задание

В этом задании вам предлагается собрать образ по выданным докерфайлам и снова определить различия.

Как и обещалось на лекции, теперь мы прочувствуем разницу между entrypoint и cmd.

Собственно, первый докерфайл:

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

И второй:

|  |
| --- |
| FROM ubuntu:22.10  COPY example.txt /  ENTRYPOINT sleep 600 |

Попытайтесь запустить из образа контейнер и войти в командный интерпретатор.

### Пример решения

|  |
| --- |
| vi Dockerfile  docker build -t testcontainer .  docker run -it testcontainer bash #вход в контейнер будет без проблем тк произойдет переопределение аргументов  docker build -t testcontainer . #пересборка докерфайла с малым временем ожидания  docker run -it testcontainer bash #вход в контейнер будет недоступен тк переопределение аргументов не произойдет |

## Задача 5 (обязательно)

### Задание

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

В данном задании демонстрируется работа инструкции EXPOSE.

Собственно, докерфайл:

|  |
| --- |
| FROM ubuntu:22.04  RUN apt-get update && \  apt-get install -y nginx  RUN rm /var/www/html/index.\*  RUN echo "Hi, I am in your container" >> /var/www/html/index.html  EXPOSE 80  ENTRYPOINT ["nginx", "-g", "daemon off;"] |

Попытайтесь запустить из образа контейнер и получить информацию утилитой curl.

### Пример решения

|  |
| --- |
| vi Dockerfile  docker build -t testcontainer .  docker run -d -p 8081:80 testcontainer  curl localhost:8081 |