Методические указания по выполнению практической работы №3: Docker

В практической работе необходимо выполнить все шаги из разделов 1–7.

В отчёт должны быть включены ответы на вопросы, выделенные *курсивом*, результаты выполнения команд из разделов 1–7, а также выполненное индивидуальное задание (раздел 8): листинг `Dockerfile`, а также команды сборки и запуска контейнера.

Для установки Docker в Windows, воспользуйтесь инструкцией из Приложения А. Если нет возможности установить Docker, то воспользуйтесь сервисом Play With Docker (см. Приложение Б).

1 Образы

Посмотрите на имеющиеся образы: `docker images`.

Загрузите образ: `docker pull ubuntu` — будет загружен образ `ubuntu:latest` — последняя доступная версия. Для загрузки конкретной версии, нужно указать тег, например, `12.04`: `docker pull ubuntu:12.04`.

Посмотрите на имеющиеся образы ещё раз: `docker images` — должны появиться новые загруженные образы. Посмотрите список контейнеров, выполнив команду: `docker ps -a`

2 Изоляция

Посмотрим информацию о хостовой системе, выполнив команду `hostname`. Выполните её ещё один раз.
Вопрос: одинаковый ли результат получился при разных запусках?

Попробуем выполнить то же самое в контейнерах. Выполните два раза команду `docker run ubuntu hostname`.
Вопрос: Одинаковый ли результат получился при разных запусках?

В случае запуска команды в контейнерах, ответ будет немного отличаться, будет разный `hostname`.
Так происходит, потому что из одного образа `ubuntu` были запущены два изолированных контейнера, поэтому у них и был разный `hostname`.
Заново выполните `docker ps -a` — там должны появиться запущенные ранее контейнеры.

Запуск контейнеров производится командой:
`docker run --флаги --докера имя_контейнера команда для запуска -и --флаги --запуска --программы`.

Запустите `bash` в контейнере: `docker run ubuntu bash`. Ничего не произошло. Это не баг. Интерактивные оболочки выйдут после выполнения любых скриптовых команд, если только они не будут

запущены в интерактивном терминале — поэтому для того, чтобы этот пример не завершился, вам нужно добавить флаги `-i -t` или сгруппированно `-it`: `docker run -it ubuntu bash`.

Выполняя запуск контейнера, указывая образ `ubuntu`, неявно указывался образ `ubuntu:latest`. Следовательно, следующие команды равнозначны:

- `docker run ubuntu hostname`
- `docker run ubuntu:latest hostname`

Если бы мы хотели запустить `ubuntu:12.04`, то нужно было бы выполнить команду
`docker run ubuntu:12.04 hostname`

3 Работа с портами

Для начала, загрузите образ `python` командой `docker pull python`.

В качестве примера, запустите встроенный в Python модуль веб-сервера из корня контейнера, чтобы отобразить содержание контейнера. `docker run -it python python -m http.server`

При запуске пишется, что сервер доступен по адресу http://0.0.0.0:8000/. Однако, если открыть этот адрес, то ничего не будет видно, потому что порты не проброшены. Завершите работу веб-сервера, нажав комбинацию клавиш `Ctrl+C`.

Для проброса портов используется флаг `-p hostPort:containerPort`

Добавьте его, чтобы пробросить порт `8000`:
`docker run -it -p8000:8000 python python -m http.server` — теперь по адресу http://0.0.0.0:8000/ (если не открывается на Windows, то вместо `0.0.0.0` нужно указать `localhost`) открывается содержимое корневой директории в контейнере.

Для того, чтобы доступный в контейнере на порту `8000` веб-сайт в хостовой системе открывался на порту `8888`, необходимо указать флаг `-p 8888:8000`:
`docker run -it -p8888:8000 python python -m http.server`.

Завершите работу веб-сервера, нажав комбинацию клавиш `Ctrl+C`.

4 Именованные контейнеры, остановка и удаление

Запустите контейнер: `docker run -it -p8000:8000 python python -m http.server`. Нажмите `Ctrl+C` — выполнение завершится. Для того, чтобы запустить контейнер в фоне, нужно добавить флаг `-d`/`--detach`. Также определим имя контейнеру, добавив флаг `--name`.

```
docker run -p8000:8000 --name pyserver -d python python -m http.server
```

Убедитесь, что контейнер всё ещё запущен: `docker ps | grep pyserver` — вывод команды не должен быть пустым. Для просмотра логов контейнера, воспользуйтесь командой `docker logs pyserver`.

Для того, чтобы остановить выполнение контейнера, существует команда `docker stop pyserver`. Однако, если снова попробовать запустить командой
`docker run -it -p8000:8000 --name pyserver -d python python -m http.server`, то возникнет ошибка: контейнер с таким именем существует. Его нужно удалить `docker rm pyserver`.
Для остановки и удаления контейнера можно воспользоваться командой `docker rm -f pyserver` вместо выполнения двух отдельных команд `stop` и `rm`. После удаления контейнер с таким именем можно будет создать заново.

Для того, чтобы контейнер удалялся после завершения работы, нужно указать флаг `--rm` при его запуске — далее в работе мы будем использовать данный флаг:
`docker run --rm -p8000:8000 --name pyserver -d python python -m http.server`

5 Постоянное хранение данных

Запустите контейнер, в котором веб-сервер будет отдавать содержимое директории `/mnt`:
`docker run -p8000:8000 --name pyserver --rm -d python python -m http.server -d /mnt`,
где `-d /mnt` указывает модулю `http.server` какая директория будет корневой для отображения.
Вопрос: Что значат остальные флаги запуска? Где здесь команда, которая выполнится в контейнере?

Для того, чтобы попасть в уже запущенный контейнер, существует команда `docker exec -it pyserver bash` — вы попадёте в оболочку *bash* в контейнере. Попав в контейнер, выполните команду
`cd mnt && echo "hello world" > hi.txt`, а затем выйдите из контейнера, введя команду `exit` или нажав комбинацию клавиш `Ctrl+D`.

Если открыть http://0.0.0.0:8000/, там будет доступен файл `hi.txt`.

Остановим контейнер: `docker stop pyserver`, а затем снова запустим:
`docker run -p8000:8000 --name pyserver --rm -d python python -m http.server -d /mnt`.

Как мы видим, файл `hi.txt` пропал — это неудивительно, ведь мы запустили другой контейнер, а старый был удалён после завершения работы (флаг `--rm`). Остановим контейнер: `docker stop pyserver`.

Для того, чтобы не терялись какие-то данные (например, если запущен контейнер с СУБД, то чтобы не терялись данные из неё) существует механизм монтирования.

5.1 Тома

Первый способ — это создать отдельный том с помощью ключа `-v myvolume:/mnt`, где `myvolume` — название тома, `/mnt` — директория в контейнере, где будут доступны данные.

Попробуйте снова создать контейнер, но уже с примонтированным томом:

```
docker run -p8000:8000 --rm --name pyserver -d \
    -v $(pwd)/myfiles:/mnt python python -m http.server -d /mnt
```

Затем, если создать файл (выполнить `docker exec -it pyserver bash` и внутри контейнера выполнить `cd mnt && echo "hello world" > hi.txt`), то даже после удаления контейнера данные в этом томе будут сохранены.

Чтобы узнать где хранятся данные, выполните команду
`docker inspect -f "{{json .Mounts }}" pyserver`, в поле `Source` будет храниться путь до тома на хостовой машине.

Для управления томами существует команда `docker volume`, ознакомиться с которой предлагается самостоятельно.

5.2 Монтирование директорий и файлов

Сперва, остановите контейнер, созданный на предыдущем шаге: `docker stop pyserver`.

Иногда требуется пробросить в контейнер конфигурационный файл или отдельную директорию. Для этого используется монтирование директорий и файлов.

Создадим директорию и файлы, которые будем монтировать. Часть из них нам понадобится дальше: создайте директорию: `mkdir myfiles`, в ней создайте файл `host.txt`: `touch myfiles/host.txt`

Запустте контейнер:
```
docker run -p8000:8000 --rm --name pyserver -d -v $(pwd)/myfiles:/mnt python \
   python -m http.server -d /mnt
```

Команда `pwd` — выведет текущую директорию, например: `/home/user/dome-directory`, в итоге получился абсолютный путь до файла: `/home/user/dome-directory/myfiles`.
Обратный слеш (\) перед переводом строки экранирует символ перевода строки и позволяет написать одну команду в несколько строк.

Затем, зайдите в контейнер: `docker exec -it pyserver bash`, перейдите в директорию `/mnt` командой `cd /mnt`. Если вывести список файлов командой `ls`, то там будет файл `host.txt`, примонтированный вместе с директорией `myfiles`

Создайте файл `echo "hello world" > hi.txt`, а затем выйдите из контейнера: `exit`. Теперь на хостовой машине в директории `myfiles/` появится файл `hi.txt`. Проверить можно командой `ls myfiles`.

Остановите контейнер: `docker stop pyserver`.

Для того, чтобы примонтировать один файл, нужно указать ключ `-v`, например:

`-v $(pwd)/myfiles/host.txt:/mnt/new-name-of-host.txt` – файлу в контейнере присвоится другое имя: `new-name-of-host.txt`.

Если на Windows возникают ошибки при монтировании, убедитесь, что вы используете `bash` *, а не* `cmd.exe`.

6 Переменные окружения

Для передачи переменных окружения внутрь контейнера используется ключ `-e`. Например, чтобы передать в контейнер переменную окружения *MIREA* во значением «*ONE LOVE*», нужно добавить ключ `-e MIREA="ONE LOVE"`.

Проверьте, выведя все переменные окружения, определённые в контейнере с помощью утилиты `env`: `docker run -it --rm -e MIREA="ONE LOVE" ubuntu env`. Среди списка переменных будет и `MIREA`

7 Dockerfile

Соберите образ, в который будут установлены дополнительные пакеты, примонтируйте директорию и установите команду запуска. Для этого создаётся файл `Dockerfile` (без расширения).

```
1 FROM ubuntu:20.04
2 RUN apt update \
3     && apt install -y python3 fortune \
4     && cd /usr/bin \
5     && ln -s python3 python
6 RUN /usr/games/fortune > /mnt/greeting-while-building.txt
7 ADD ./data /mnt/data
8 EXPOSE 80
9 CMD ["python", "-m", "http.server", "-d", "/mnt/", "80"]
```

Будьте внимательны при копировании, могут скопироваться неправильные минусы и лишние пробелы.
В строке (1) указывается базовый образ, на основе которого будет строиться новый образ.
В строках (2-5) указана команда, которая выполнится в процессе сборки. На самом деле, там выполняются несколько команд, соединённых `&&` для того, чтобы создавать меньше слоёв в образе.
В строках (6) тоже указана команда, которая сгенерирует случайную цитату и перенаправит вывод в файл `/mnt/greeting-while-building.txt`. Файл будет сгенерирован во время сборки образа.
В строке (7) копируется всё содержимое директории `./data` хостовой машины в директорию `/mnt`, которая будет доступна в контейнере.
В строке (8) указывается, какой порт у контейнера будет открыт.
В строке (9) указывается команда, которая будет выполнена при запуске, где `80` — порт, который будет слушать веб-сервер.

Соберите образ с тегом `mycoolimage` с помощью команды `docker build -t mycoolimage .` Точка в конце указывает на текущую директорию, где лежит `Dockerfile`.

Запуск производится командой `docker run --rm -it -p8099:80 mycoolimage`, где порт `8099` — порт на хостовой машине.

Хорошая статья с описанием `Dockerfile` на Хабре: Изучаем Docker, часть 3: файлы Dockerfile.

8 Индивидуальные задания

Написать `Dockerfile`, собрать образ, запустить контейнер (и записать команду для его запуска).

Для монтирования создайте директорию `data` и в ней файл `student.txt`, содержащий ФИО, название группы и номер варианта.

Для установки пакетов использовать команду `apt install -y название-пакета`. В качестве примера можно использовать `Dockerfile` из раздела 7.

Чётные варианты:

- необходимо использовать базовый образ `ubuntu:20.10`
- примонтировать файл `data/student.txt` как `/mnt/files/student.txt` в контейнере.

Нечётные варианты:

- необходимо использовать базовый образ `ubuntu:20.04`
- примонтировать директорию `data` в директорию `/mnt/files/` в контейнере.

Запустить веб-сервер, отображающий содержимое `/mnt/files`, в хостовой системе должен открываться на порту (8800 + номер варианта). Например, для 22-го варианта это порт 8822.

Установить пакет, согласно варианту:

1. cowsay
2. figlet
3. zip
4. imagemagick
5. git
6. patch
7. php-cli
8. postgresql-client
9. mysql-client
10. jq
11. gpg
12. wget
13. nginx
14. nano
15. emacs-nox

Приложение А. Установка Docker в Windows

1. Установить подсистему Windows для Linux в Windows 10;
2. Установить Docker Desktop.

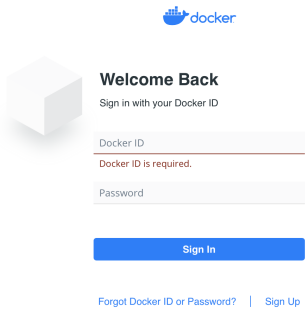
Приложение Б. Использование PlayWithDocker

Если нет возможности установить Docker, можно воспользоваться сервисом labs.play-with-docker.com.

1. Регистрируемся на id.docker.com.

2. Заходим на labs.play-with-docker.com:

3. Входим через Docker ID `id.docker.com`:



4. Нажимаем «Старт»:

5. Создаём новый инстанс (будет доступен с момента создания в течении 4-х часов), клинкув по «Add new instance»:

6. В разделах, где есть текст про открыть `http://0.0.0.0:8000/` и т.п., нужно будет кликнуть на порт вверху страницы. Пример на картинке — 8000: