Стригунов Эрик Сергеевич
ФИТУ 3 курс 09.03.03-ПИб-о21

Практическая работа №2

Основы работы с технологиями контейнеризации и ботами Telegram

Цель работы. Выполнение практической работы направлено на:

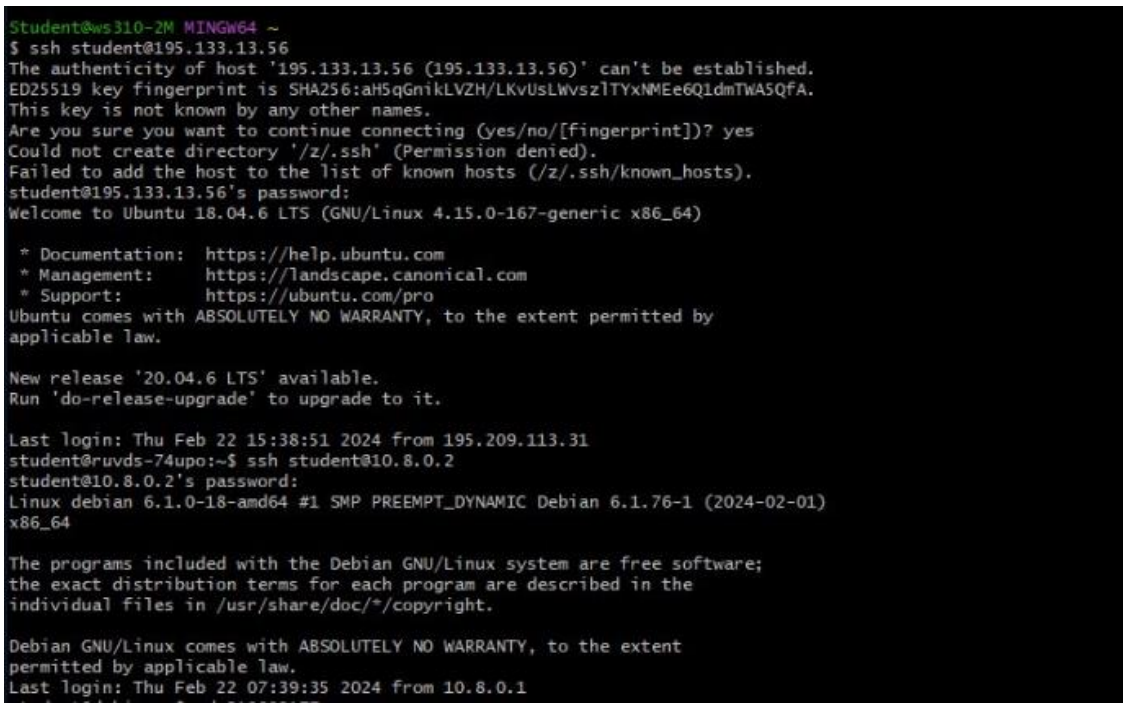
1. изучение базовых команд ubuntu
2. создание на сервере работающего Telegram бота с простейшим функционалом
3. использование контейнеризации приложений Docker

Ход работы

Для работы будем использовать Git Bash, работа которого описана в прошлой лабораторной, он будет нам заменой Windows PowerShell.

Для начала подключимся к серверу, на котором и будем производить всю основную работу. Используем заранее созданный облачный сервер как шлюз к основному рабочему серверу.

Для этого, вводим поочередно команды ssh student@195.133.13.56 и ssh student@10.8.0.2, попутно указывая пароль 24, там где это необходимо



Все дальнейшие действия по работе с контейнеризацией и ботом мы будем производить именно на этом сервере

Теперь создадим рабочий каталог по номеру зачетки

При создании каталога, мой номер зачетки, а именно 210803177, был уже занят, поэтому в результате ввода команды mkdir 210803177, выдавалась ошибка, в связи с этим было принято решение создать новый каталог, а именно 210803177_

Далее переходим только в что созданный созданный каталог

```
student@debian:~$ cd 210803177_
```

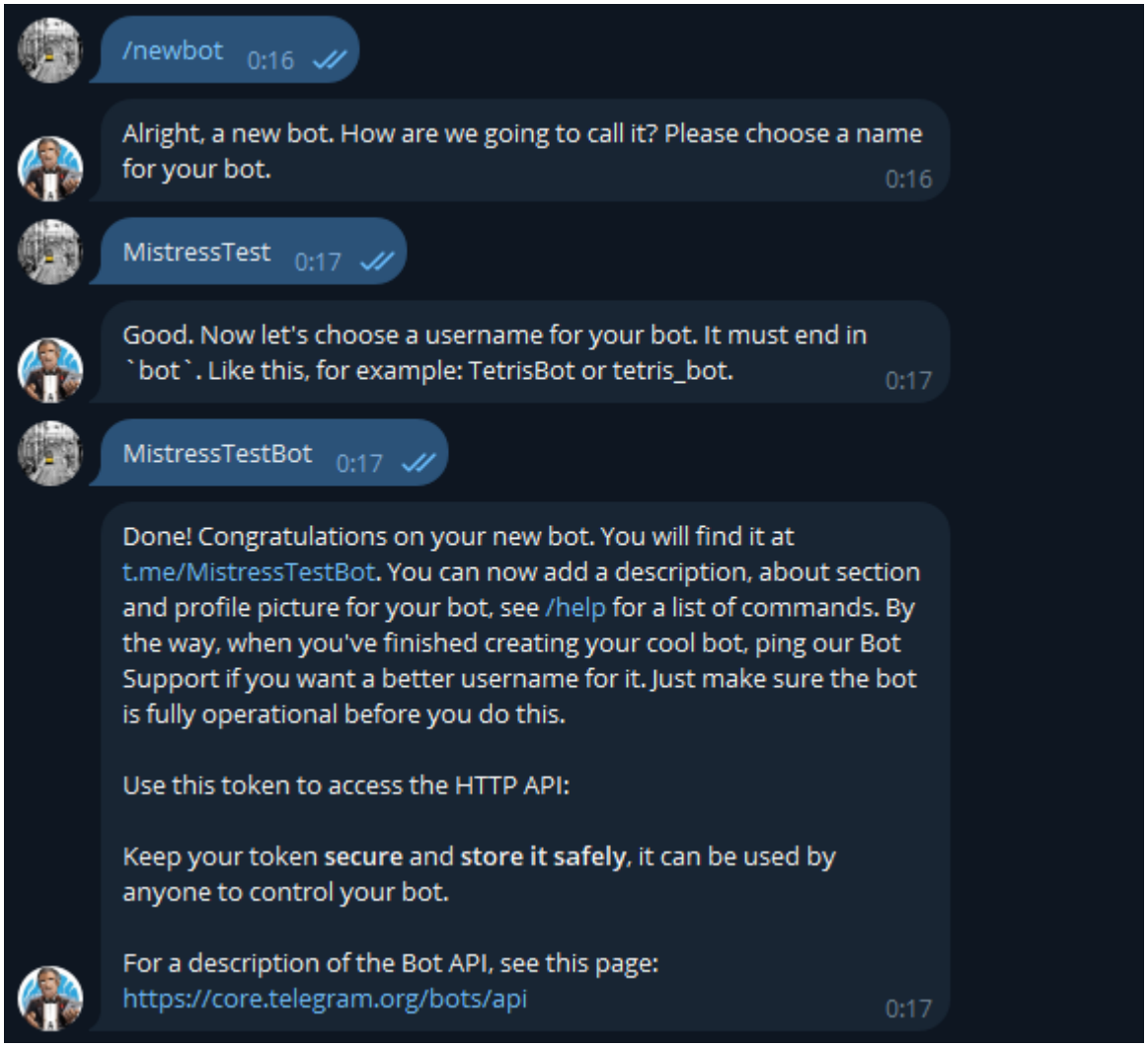
Проверим работу python3, введя команду python3, и после успешного результата, выходим из командного пространства python3, введя команду exit()

Далее, чтобы не мешать дальнейшими действиями другим пользователям сервера и администраторам, создадим окружение Python, поочередно введя следующие команды

```
student@debian:~/210803177_$ python3 -m venv env
student@debian:~/210803177_$ source env/bin/activate
(env) student@debian:~/210803177_$
```

как мы видим, после успешного срабатывания команд, мы активировали, то есть перешли в виртуальное окружение, о чем свидетельствует приписка (env) в начале

Для работы с Telegram ботом, нам нужно его сначала создать, в чем нам поможет BotFather, или другими словами отец всех ботов. Итак, находим в телеграмме данного бота @BotFather, и в результате диалога получаем специальный токен, который понадобится нам позже



Продолжим работу с командной строкой. Определимся с функционалом нашего бота. По началу, для проверки сделаем простейшие команды

/command1 – вывод Oks

/command2 – вывод Ok

создадим файл и запустим текстовый редактор внутри командной строки командой nano bot.py куда введем текст программы

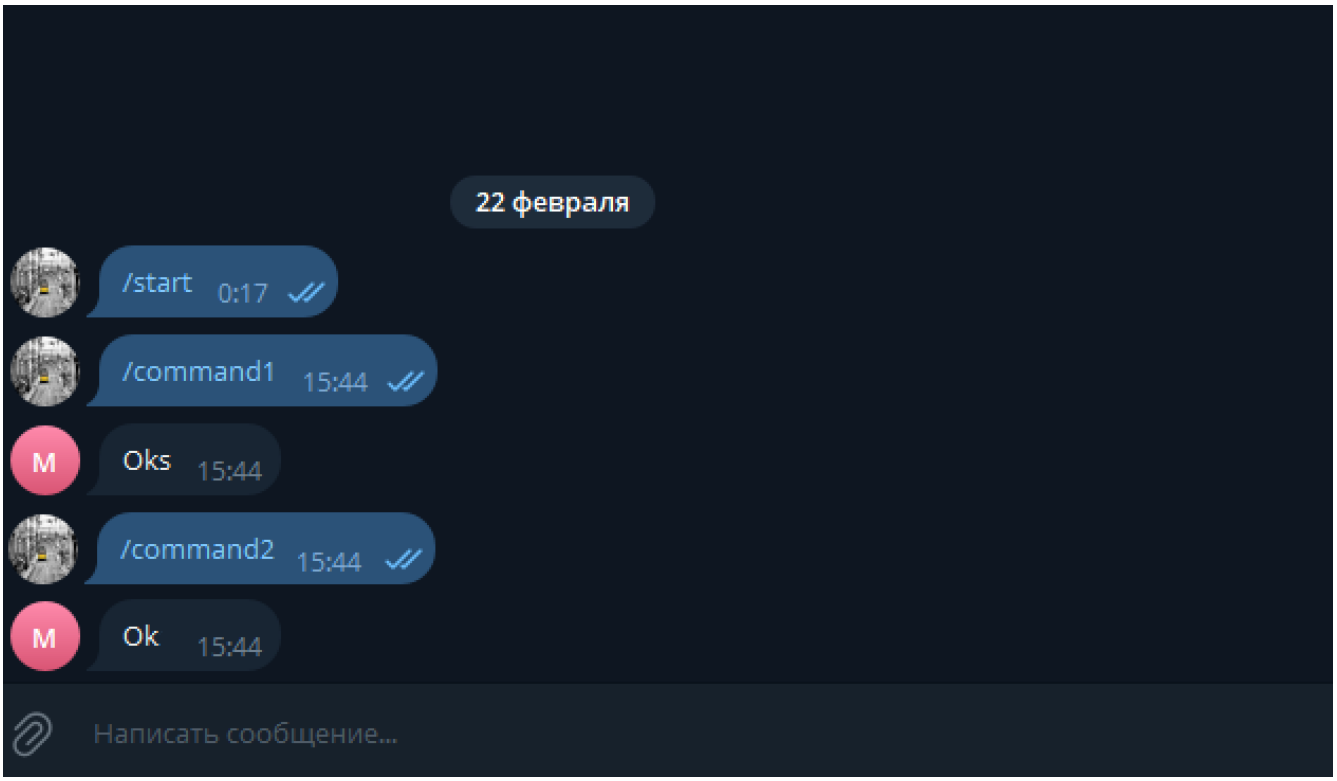
```
  GNU nano 7.2                                bot.py
import telepot
import time
def handle(msg):
 chat_id = msg['chat']['id']
 command = msg['text']
 print('Got command: %s' % command)
 print('From : %s' % chat_id)
 if command == '/command1':
  bot.sendMessage(chat_id, 'Oks')
 elif command == '/command2':
  bot.sendMessage(chat_id, 'Ok')
bot = telepot.Bot('7179946955:AAEwLx164CfOIGHpy2ch2H6yVaNnl8ch6-U')
bot.message_loop(handle)
print('I am listening ...')
while 1:
 time.sleep(10)


                [ Read 16 lines ]
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

После чего сохраняем файл, сочетанием клавиш ctrl+O, после чего подтверждаем имя, нажав на Enter и выходим из редактора сочетанием ctrl+X

Теперь запускаем программу командой python3 bot.py

Переходим в нашего созданного бота и проверим работоспособность



В командной строке же:

```
(env) student@debian:~/210803177_$ python3 bot.py
I am listening ...
Got command: /command1
From : 772888420
```

Что так же указывает на работоспособность программы

Выключаем бота, нажав сочетание ctrl+C

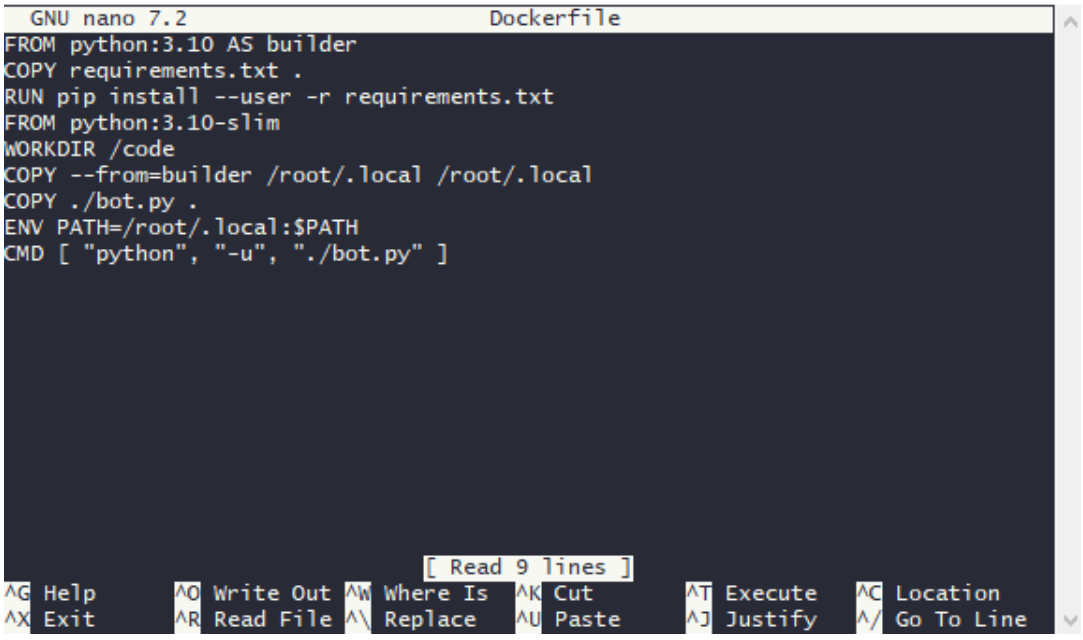
Создаем файл requirements.txt со списком pip-библиотек, необходимых для работы нашей программы командой nano requirements.txt и пишем внутрь лишь одну строчку:

telepot==12.7

Теперь перейдем к контейнирезации и создании docker образа

Для этого создаем файл для сборки docker образа (nano Dockerfile)

и вставляем внутрь код:



Далее собираем docker образ с именем – номером зачетки командой

docker build -t 210803177 .

Но так как мы помним, что ранее уже кто-то работал под этим номером, используем другой, 210803177_, однако, создание такого образа невозможно

```
(env) student@debian:~/210803177_$ docker build -t 210803177_ .
[+] Building 0.0s (0/0)                                          docker:default
ERROR: invalid tag "210803177_": invalid reference format
```

Поэтому было принято решение, назвать его 2108031770

```
(env) student@debian:~/210803177_$ docker build -t 2108031770 .
[+] Building 0.2s (13/13) FINISHED                                docker:default
 => [internal] load build definition from Dockerfile                       0.0s
 => => transferring dockerfile: 295B                                       0.0s
 => [internal] load metadata for docker.io/library/python:3.10-slim        0.1s
 => [internal] load metadata for docker.io/library/python:3.10             0.1s
 => [internal] load .dockerignore                                          0.0s
 => => transferring context: 2B                                            0.0s
 => [builder 1/3] FROM docker.io/library/python:3.10@sha256:b54e76c629a98  0.0s
 => [stage-1 1/4] FROM docker.io/library/python:3.10-slim@sha256:4bd9a0e5  0.0s
 => [internal] load build context                                          0.0s
 => => transferring context: 527B                                          0.0s
 => CACHED [stage-1 2/4] WORKDIR /code                                     0.0s
 => CACHED [builder 2/3] COPY requirements.txt .                           0.0s
 => CACHED [builder 3/3] RUN pip install --user -r requirements.txt        0.0s
 => CACHED [stage-1 3/4] COPY --from=builder /root/.local /root/.local     0.0s
 => [stage-1 4/4] COPY ./bot.py .                                          0.0s
 => exporting to image                                                     0.0s
 => => exporting layers                                                    0.0s
 => => writing image sha256:3e3afe6196348405c5d34abbfc1640a5732688f489d53  0.0s
 => => naming to docker.io/library/2108031770                              0.0s
```

Запускаем docker образ в режиме работы в фоне (-d) и даем команду запуска при перезапуске docker (--restart=always)

Вводим команду:

```
(env) student@debian:~/210803177_$  docker run -d --restart=always 2108031770
9554094ad3767274ba65c012108fb95d825e056124266d9aef9907d6628cc79e
docker: Error response from daemon: connection error: desc = "transport: Error w
hile dialing: dial unix /run/containerd/containerd.sock: connect: connection ref
used": unavailable.
```

В результате, команда выдала ошибку, которая указывает на проблему с подключением Docker к его даемону, проблема может быть вызвана тем что Docker даемон не запущен, проблемами с правами доступа, с сокетом или конфликтами с другими сервисами.

Но не смотря на это, нам все равно был выдан container ID

Из-за ошибки с Docker, последующая часть выполнения методических рекомендаций невозможна.

Поэтому выполним лишь сохранение контейнера для последующей возможности его выгрузки на другой компьютер командой

```
(env) student@debian:~/210803177_$ docker save -o ./docker_image_2108031770.tar 2108
031770
```

, где 2108031770 – номер зачетки с нулем в конце.

Вывод: в результате проделанной работы, были изучены концепции работы с Docker, был создан telegram bot с простейшим функционалом, способный отвечать на определенные команды: /command1 и /command2. Так же были получены навыки работы с виртуальным окружением python3, рабочей средой nano. В ходе работы возникли проблемы с названиями каталога и образа, но они были решены альтернативными аналогичными названиями. Так же возникли проблемы с подключением Docker, в связи с чем, некоторые команды в процессе не были реализованы.

Выполнил	Стригунов Э.С.
Проверил	Шайхутдинов Д.В.