

Практическая работа №2

Основы работы с технологиями контейнеризации и ботами Telegram

Цель работы.

Создание сервера с постоянно работающим Telegram ботом

Программа работы.

Следует изучить базовые команды ubuntu https://help.ubuntu.ru/wiki/командная_строка и рекомендуется держать данную ссылку открытой, в т.ч. использовать для формирования отчета по работе.

Создание собственного облачного сервера описано в приложении 1, однако, доступные облачные серверы не имеют обычно достаточных ресурсов для обеспечения работы значительного количества пользователей, мы его используем только как шлюз к основному рабочему серверу (рис. 1).

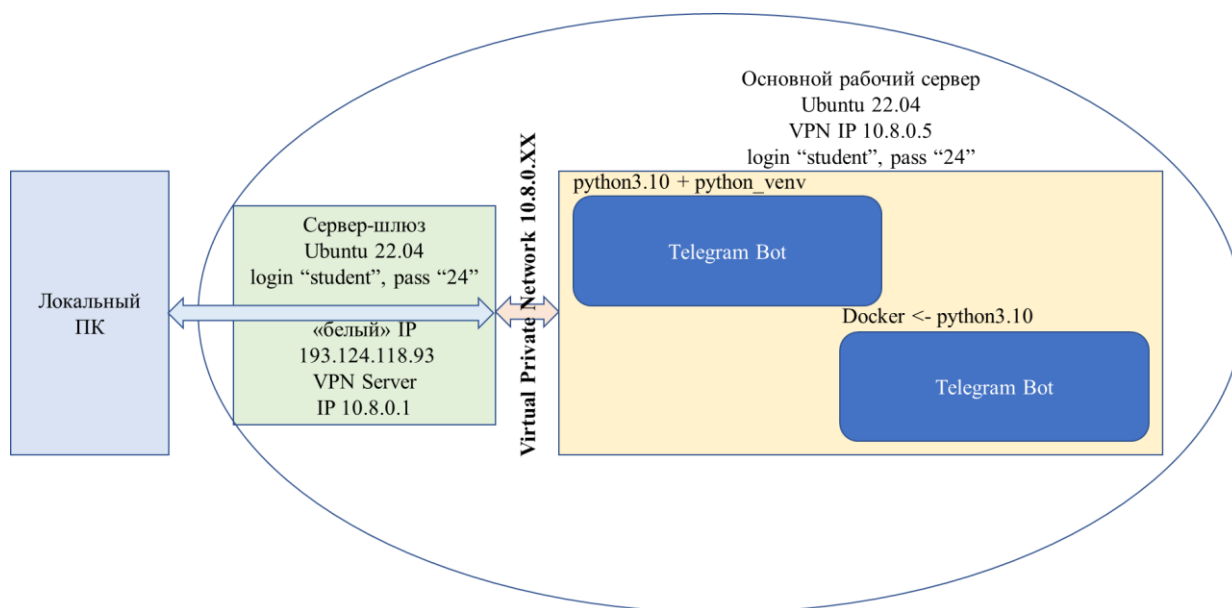


Рисунок 1 – Схема организации доступа к серверу для запуска Telegram-бота

1.1. Из окна Windows PowerShell подключаемся к серверу-шлюзу по ssh

```
> ssh student@193.124.118.93
```

```
> 24
```

1.2. От сервера-шлюза подключаемся к основному рабочему серверу

```
student@ruvds-x7i06:~$ ssh student@10.8.0.5
```

```
24
```

1.3. Создаем свой (по номеру зачетки) рабочий каталог и переходим в него

```
student@user-IPMSB-H61:~$ mkdir 000000 && cd 000000
```

1.4. Проверяем работу python3.10

```
student@user-IPMSB-H61:~/000000$ python3.10
```

```
>>> exit()
```

1.5. Для того, чтобы не нарушать структуру базового python, не мешать своими установками администраторам серверов и коллегам, создаем «окружение» python env и активируем его [https://netpoint-dc.com/blog/python-venv-ubuntu-1804/]

```
student@user-IPMSB-H61:~/000000$ python3.10 -m venv env
```

```
student@user-IPMSB-H61:~/000000$ source env/bin/activate
```

В результате в начале командной строки появляется указание на использование окружения:

```
(env) student@user-IPMSB-H61:~/000000$
```

1.6. Устанавливаем необходимые pip-пакеты, в частности, нам понадобится

```
(env) student@user-IPMSB-H61:~/000000$ pip install telepot==12.7
```

1.7. Создаем собственную учетную запись – нового бота, как это указано в Приложении 2 (дополнительно см. "Step 2: Text /newbot to BotFather" <https://www.instructables.com/Set-up-Telegram-Bot-on-Raspberry-Pi/>).

Получаем токен, его будет достаточно для работы простейшего приложения, которое на /command1 будет отвечать Oks, а на /command2 - Ok.

```
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('***** PUT YOUR TOKEN HERE *****')
```

```
bot.message_loop(handle)
```

```
print('I am listening ...')
```

```
while 1:
```

```
    time.sleep(10)
```

Копируем текст программы в буфер обмена, запускаем текстовый редактор

```
(env) student@user-IPMSB-H61:~/000000$ nano bot.py
```

Нажимаем в открывшемся пустом документе правой кнопкой мыши, текст программы вставляется.

Клавишами доходим до текста

```
***** PUT YOUR TOKEN HERE *****
```

удаляем его, копируем токен, полученный от BotFather на его место.

Сохраняем файл (Ctrl+O), выходим (Ctrl+X).

Запускаем программу

```
(env) student@user-IPMSB-H61:~/000000$ python bot.py
```

Проверяем работу бота, находя его через telegram.

- 1.8. Настраиваем работу собственной python-программы в виде docker-контейнера с автозапуском после старта ОС [в соответствии с <https://habr.com/ru/companies/vdsina/articles/555540/> и <https://habr.com/ru/companies/ruvds/articles/439980/>]

- 1.8.1. Отключаем python env, так как теперь в качестве закрытого окружения будет docker-контейнер:

```
(env) student@user-IPMSB-H61:~/000000$ deactivate
```

```
-> student@user-IPMSB-H61:~/000000$
```

- 1.8.2. Создаем файл requirements.txt со списком pip-библиотек, необходимых для работы нашей программы

```
student@user-IPMSB-H61:~/000000$ nano requirements.txt
```

```
telepot==12.7
```

Сохраняем файл (Ctrl+O), выходим (Ctrl+X).

- 1.8.3. Создаем файл для сборки docker образа

```
student@user-IPMSB-H61:~/000000$ nano Dockerfile
```

Вставляем в файл

```
FROM python:3.10 AS builder
```

```
COPY requirements.txt .
```

```
RUN pip install --user -r requirements.txt
```

```
FROM python:3.10-slim
```

```
WORKDIR /code
```

```
COPY --from=builder /root/.local /root/.local
```

```
COPY ./bot.py .
```

```
ENV PATH=/root/.local:$PATH
```

CMD ["python", "-u", "./bot.py"]

Сохраняем файл (Ctrl+O), выходим (Ctrl+X).

Следует обратить внимание на версию python: указанная в данном примере соответствует python, установленному на сервере 10.8.0.5, если Вы используете свой сервер, но следует указать версию, установленную на нем, например, для Debian 12 по умолчанию устанавливается python:3.11

1.8.4. Собираем docker образ с именем – номером зачетки

```
student@user-IPMSB-H61:~/000000$ docker build -t 000000 .
```

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

```
student@user-IPMSB-H61:~/000000$ docker run -d --restart=always 000000
```

В ответ на данную команду, docker сообщает CONTAINER ID вида

```
5df687ebc2f6380abd23e4ac5f7899c5f9a8a0e414cfa633ffefb0c372e40fcd
```

Также данный CONTAINER ID можно понять из списка, выдаваемого в ответ на команду

```
student@user-IPMSB-H61:~/000000$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
5df687ebc2f6	000000	"python -u ./bot.py"	About a minute ago	Up About a minute		bold_bhabha

В данном случае CONTAINER ID значительно короче; можно пользоваться любым номером.

Например, для просмотра log'ов (в данном случае - результатов работы функций "print(" программы).

```
student@user-IPMSB-H61:~/000000$ docker logs 5df687ebc2f6
```

```
Got command: /command1
```

```
From : 384540256
```

1.8.6. После проверки следует сохранить docker image в виде архива. Это может быть полезно для передачи Вашим заказчикам, например, если нет желания и возможности воспользоваться Docker Hub

1.8.6.1. Сохраняем образ командой [<https://stackoverflow.com/questions/24482822/how-to-share-my-docker-image-without-using-the-docker-hub>]

```
$ docker save -o <path for created tar file> <image name>
```

Например,

```
student@user-IPMSB-H61:~/000000$ docker save -o ./docker_image_000000.tar 000000
```

1.8.6.2. Отключаемся от основного рабочего сервера (оказываемся на сервере-шлюзе):

```
student@user-IPMSB-H61:~/000000$ exit
```

1.8.6.3. Копируем архив docker-образа на шлюз:

```
student@ruvds-x7i06:~$ scp student@10.8.0.5:/home/student/docker_image_000000.tar .
```

(необходимо будет указать пароль пользователя - 24)

1.8.6.4. Отключаемся от шлюз-сервера

```
student@ruvds-x7i06:~$ exit
```

и копируем этот файл к себе на ПК выполнив уже на локальном ПК

```
> scp student@193.124.118.93:/home/student/docker_image_000000.tar .
```

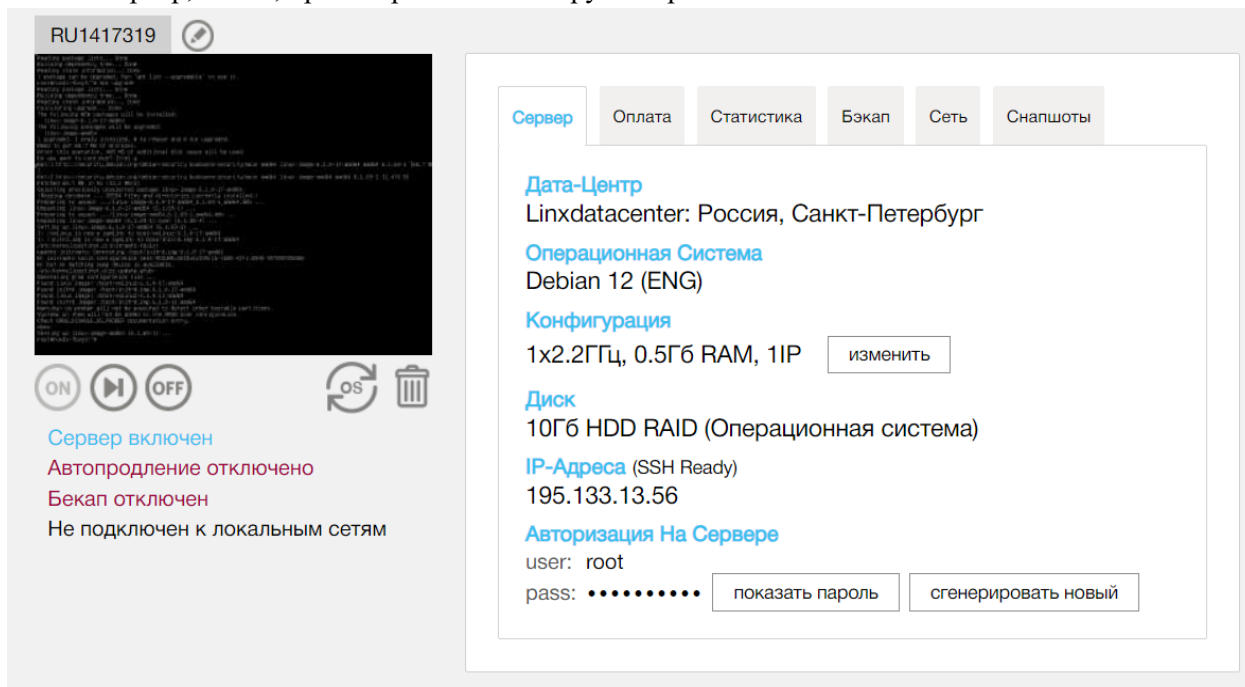
(необходимо будет указать пароль пользователя - 24)

Приложение 1

Подготовка собственного облачного сервера для выполнения программ в docker

1.1. Покупаем себе сервер VPS, например, вот тут <https://ruvds.com/ru-rub/my/orders>, выбирая в качестве ОС, например, Debian 12.

1.2. Ждем, пока завершится установка, видим в <https://ruvds.com/ru-rub/my/servers> новый сервер, его IP, просматриваем и копируем пароль.



1.3. На рисунке показан существующий сервер 195.133.13.56 и его можно использовать. Из Windows PowerShell подключаемся к нему удаленно под пользователем root

```
> ssh root@195.133.13.56
```

1.4. Стандартные команды проверки последних обновлений для Ubuntu/Debian после установки:

```
# apt-get update
```

```
# apt-get upgrade
```

Кроме того, устанавливаем htop

```
# apt install htop
```

1.5. Создаем нового пользователя student с собственным каталогом

```
# useradd -m student -s /bin/bash
```

и задаем ему пароль

```
# passwd student
```

```
24
```

```
24
```

1.6. Добавляем его в группу, которая может подключаться по ssh к серверу
[<https://ostechnix.com/allow-deny-ssh-access-particular-user-group-linux/>]

```
# nano /etc/ssh/sshd_config
```

в конце файла добавляем

```
AllowUsers student root
```

Сохраняем файл (Ctrl+O), выходим (Ctrl+X).

Перезапускаем службу ssh

```
# systemctl restart sshd
```

Попытаемся из второго окна Windows PowerShell подключиться с указанными учетными данными

```
> ssh student@195.133.13.56
```

> 24

* Иногда проявляется ошибка в подключении к серверу по ssh (долгое ожидание сообщение о невозможности подключения). В таком случае следует воспользоваться

<https://serverfault.com/a/918810> https://www.seei.biz/ssh-fails-to-connect-with-debug1-expecting-ssh2_msg_kex_ecdh_reply/

1.7. В первом окне Windows PowerShell из-под учетной записи root устанавливаем docker engine <https://docs.docker.com/engine/install/debian/>

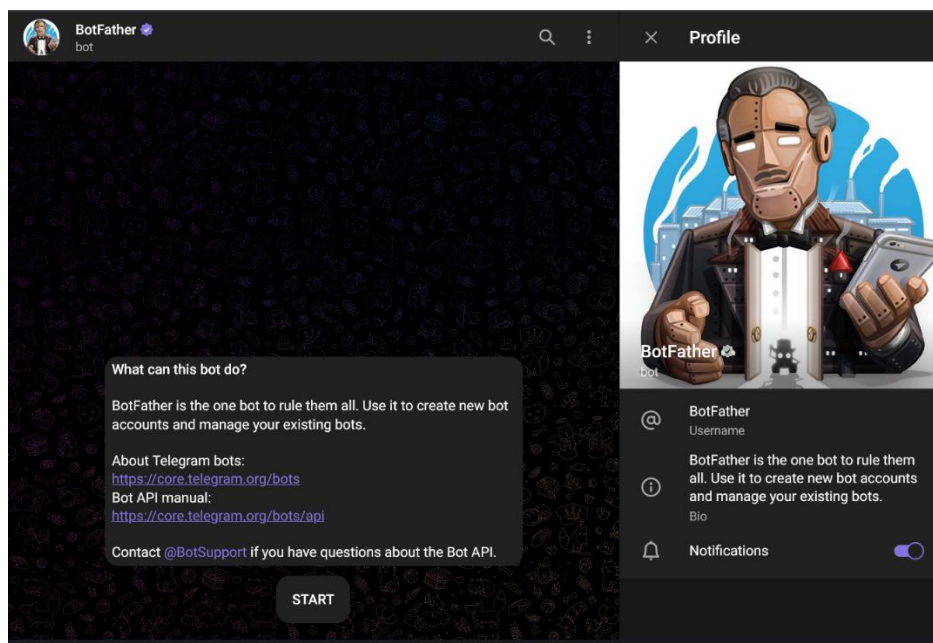
```
# apt-get install ca-certificates curl
# install -m 0755 -d /etc/apt/keyrings
# curl -fsSL https://download.docker.com/linux/debian/gpg -o /etc/apt/keyrings/docker.asc
# chmod a+r /etc/apt/keyrings/docker.asc
# echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/debian \
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
# apt-get update
# apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
Проверяем работу под root (после указанной команды ошибок быть не должно)
# docker run hello-world
Добавляем группу docker
# groupadd docker
Добавляем в эту группу student'a
# usermod -aG docker student
В окне с учетной записью student сначала переобновляем свои данные в группе
$ newgrp docker
потом проверяем работу (после указанной команды ошибок быть не должно)
$ docker run hello-world
```

1.8. Устанавливаем python из учетной записи root

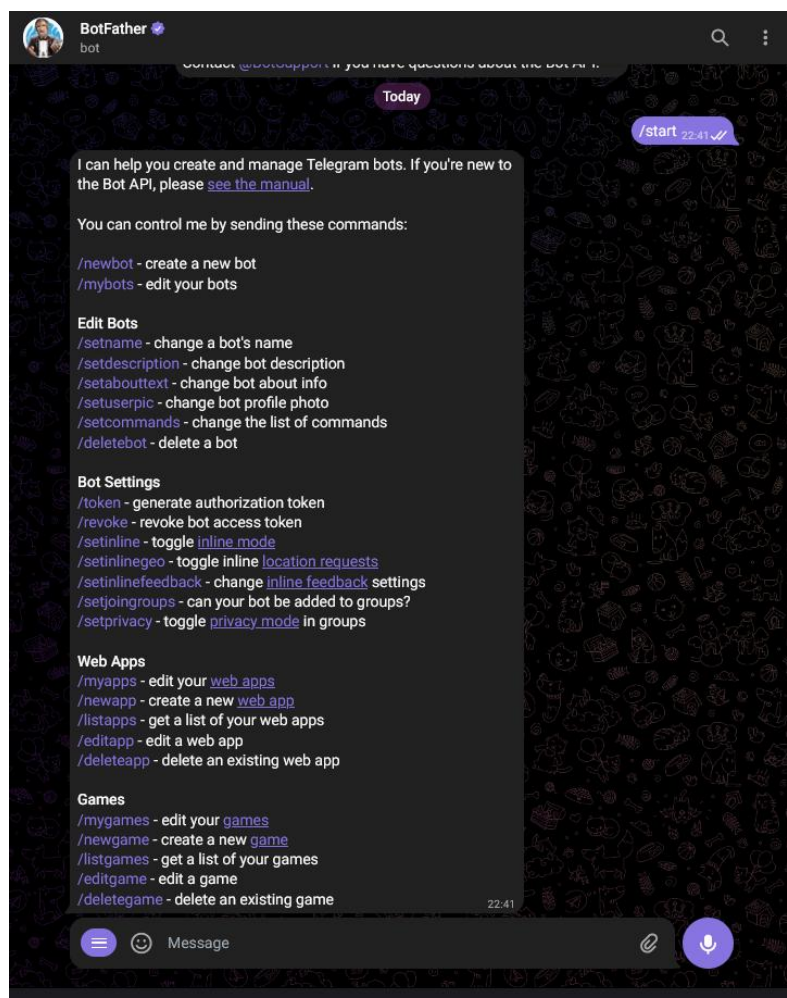
```
# apt install python3 python3-pip python3-venv
Затем снимаем запрет student'у устанавливать пакеты через pip
# rm /usr/lib/python3.11/EXTERNALLY-MANAGED
```


Приложение 2

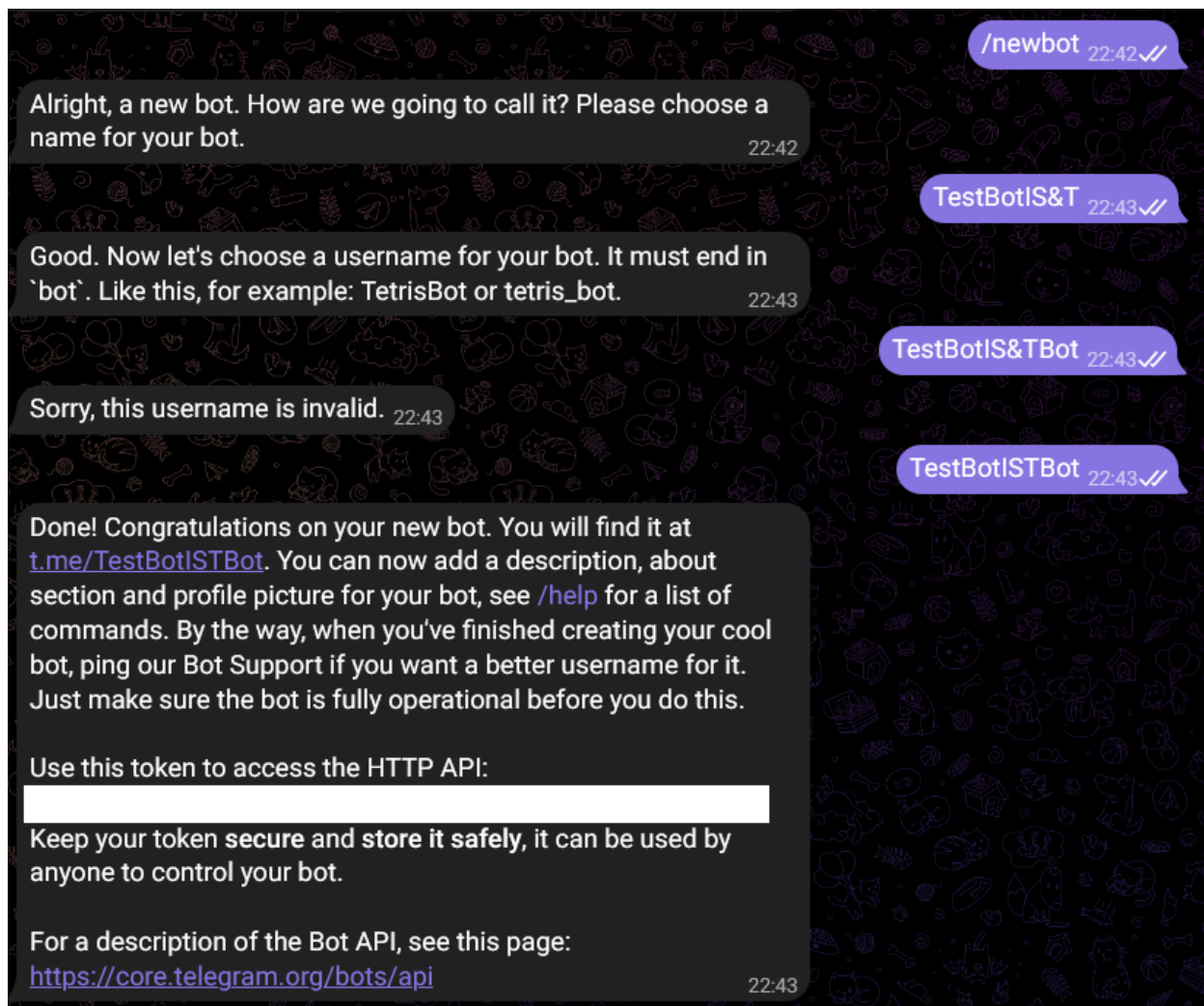
Откройте Telegram на своем телефоне, найдите пользователя по имени BotFather (поиск формирует несколько аналогов, нужен именно @BotFather). Как следует из названия, он является «отцом» всех ботов.



Он принимает специальные команды.



Чтобы получить учетную запись бота, отправьте ему сообщение /newbot. Он задаст пару вопросов. В конце процесса вам будет предоставлен токен, имеющий вид аналогичный 123456789:ABCdefGhIJKlmNoPQRsTUVwxyz. Этот токен вам потребуется для работы программы на python. Скопируйте его в текстовый файл.



Приложение 3

Дополнительная информация по работе с Docker

Если Вы создавали docker-образ на своем ПК и хотите передать его на какой-либо сервер, то команда должна иметь вид

```
$ scp ./docker_image_000000.tar student@193.124.118.93:/home/student/
```

На целевом для запуска контейнера ПК распаковываем tar-файл в каталог docker

```
$ docker load -i ./docker_image_000000.tar
```

```
571ade696b26: Loading layer [=====>] 77.82MB/77.82MB
e96fe707bd25: Loading layer [=====>] 9.551MB/9.551MB
4b58789a003f: Loading layer [=====>] 35.28MB/35.28MB
2655468a1a9f: Loading layer [=====>] 5.12kB/5.12kB
0b61847d9826: Loading layer [=====>] 12.99MB/12.99MB
801e13511f2f: Loading layer [=====>] 1.536kB/1.536kB
808b39b933eb: Loading layer [=====>] 10.97MB/10.97MB
ea286d404790: Loading layer [=====>] 3.072kB/3.072kB
Loaded image: 000000:latest
```

Запуск образа (создается контейнер)

```
$ docker run -d --restart=always 000000
```

```
50046704457e9745897ba2c36e99e9c115ef89f3c41fa443beca5a7668668342
```

Остановка контейнера

```
$ docker stop 50046704457e9745897ba2c36e99e9c115ef89f3c41fa443beca5a7668668342
```

```
50046704457e9745897ba2c36e99e9c115ef89f3c41fa443beca5a7668668342
```

Удаление контейнера

```
$ docker rm 50046704457e9745897ba2c36e99e9c115ef89f3c41fa443beca5a7668668342
```

Удаление образа image

```
$ docker image rm 000000
```

```
Untagged: 000000:latest
```

```
Deleted: sha256:d863657cdb75bd42ae87353533ab6bd2201cfce16a7be2ec3c7b028b6341e5ab
```

```
Deleted: sha256:302c36b8d0f3add37345463c5235c21088c047fa0d6f19693bd9f66db7442bc6
```

```
Deleted: sha256:e87a9bf251080da4af8e7ed51e4048a453b1deb7f9d73a526b08f96f343b366b
```

```
Deleted: sha256:8466b1a13177626112a3b27b33bb6dd8bb013e4bdedf27c2d29c52b31c26d908
```

```
Deleted: sha256:4ec2e811eb3bca56124349186481b085bfa8f26b073777fc94af8b4ac785649
```

```
Deleted: sha256:e8f29a68c53010eb307c1a07df90028fe4b7e0766bd9d7be0fc80287fa93a116
```

```
Deleted: sha256:e2b880ce8192f572ac31d7db0071633982c965863927fe8e5048d32f88cc6218
```

```
Deleted: sha256:cd6b0451a6fea2b8ab85302915856393122ec8db6399363a01d273a98c819ac2
```

Deleted: sha256:571ade696b261f0ff46e3cdac4635afc009c4ed3429950cb95cd7e5f70ba0a07