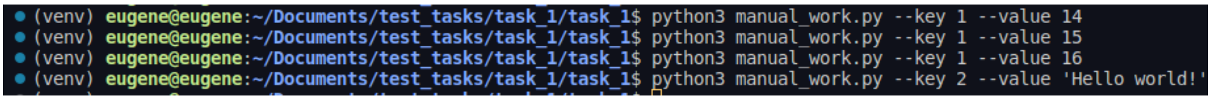
Гит репозиторий

https://github.com/BoyFromDubai/test_tasks/tree/master/task_1

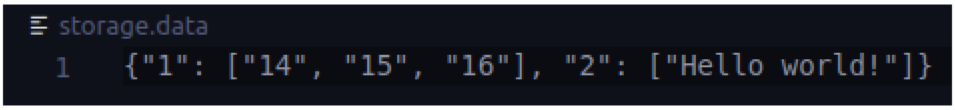
Задание 1.1

Добавление значений:



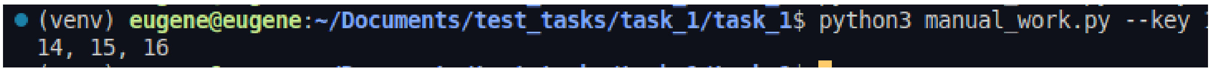
```
● (venv) eugene@eugene:~/Documents/test_tasks/task_1/task_1$ python3 manual_work.py --key 1 --value 14
● (venv) eugene@eugene:~/Documents/test_tasks/task_1/task_1$ python3 manual_work.py --key 1 --value 15
● (venv) eugene@eugene:~/Documents/test_tasks/task_1/task_1$ python3 manual_work.py --key 1 --value 16
● (venv) eugene@eugene:~/Documents/test_tasks/task_1/task_1$ python3 manual_work.py --key 2 --value 'Hello world!'
```

Данные в файле:



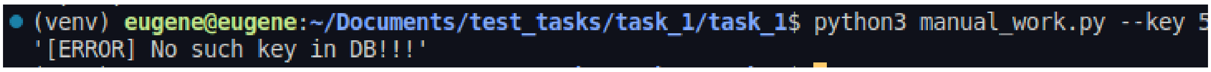
```
storage.data
  1   {"1": ["14", "15", "16"], "2": ["Hello world!"]}
```

Получение по ключу:



```
● (venv) eugene@eugene:~/Documents/test_tasks/task_1/task_1$ python3 manual_work.py --key 1
  14, 15, 16
```

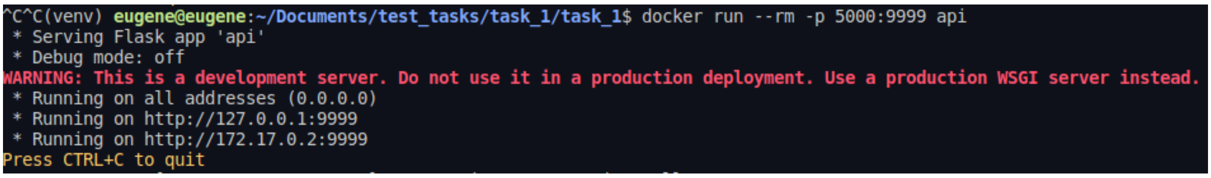
Несуществующий ключ:



```
● (venv) eugene@eugene:~/Documents/test_tasks/task_1/task_1$ python3 manual_work.py --key 5
  '[ERROR] No such key in DB!!!'
```

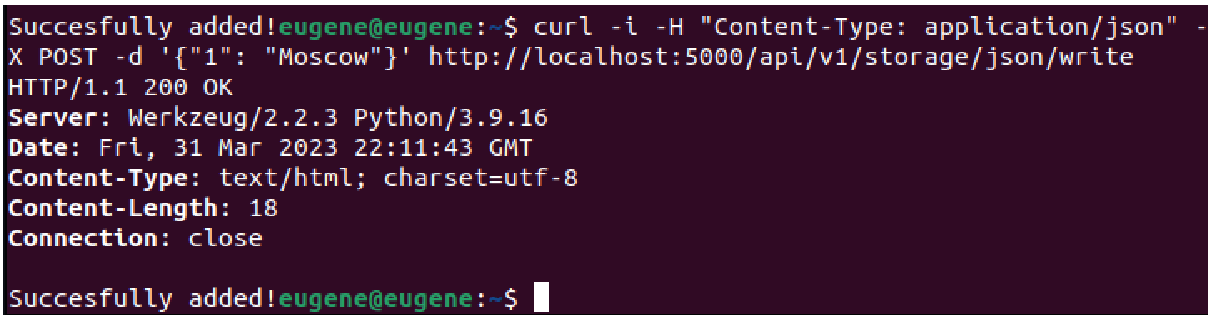
Задание 1.2

Запуск контейнера:

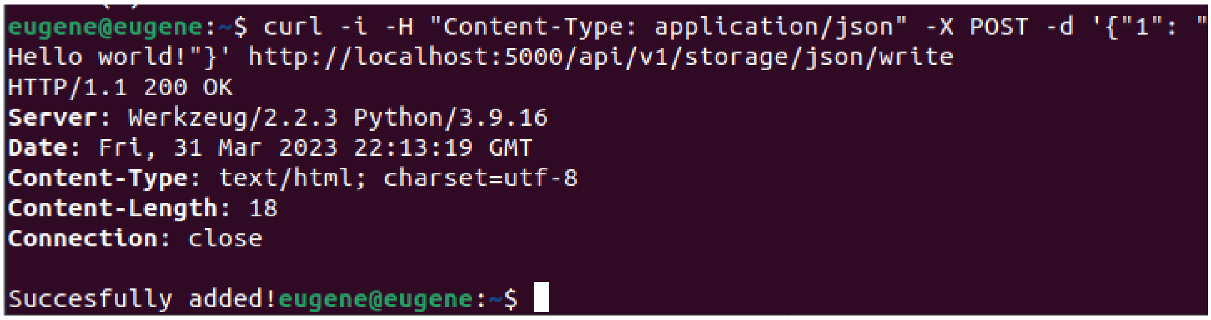


```
^C^C(venv) eugene@eugene:~/Documents/test_tasks/task_1/task_1$ docker run --rm -p 5000:9999 api
 * Serving Flask app 'api'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:9999
 * Running on http://172.17.0.2:9999
Press CTRL+C to quit
```

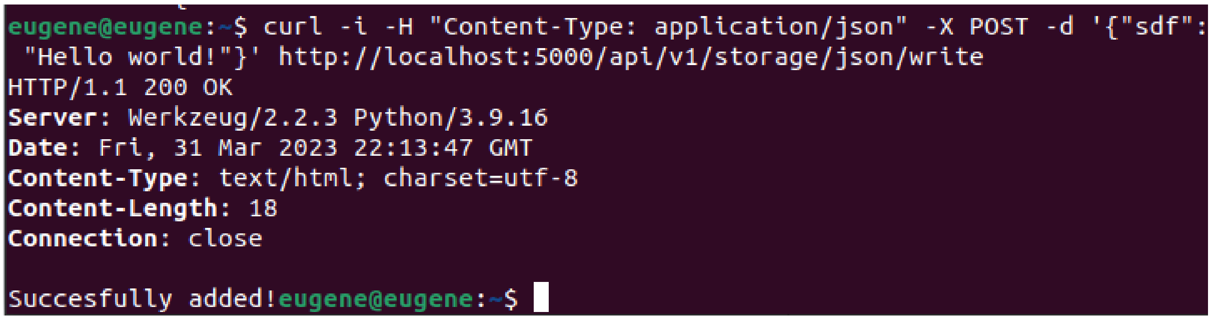
POST-запрос:



```
Succesfully added!eugene@eugene:~$ curl -i -H "Content-Type: application/json" -
X POST -d '{"1": "Moscow"}' http://localhost:5000/api/v1/storage/json/write
HTTP/1.1 200 OK
Server: Werkzeug/2.2.3 Python/3.9.16
Date: Fri, 31 Mar 2023 22:11:43 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 18
Connection: close

Succesfully added!eugene@eugene:~$ 
```

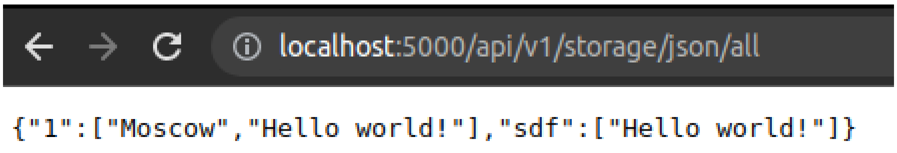


```
eugene@eugene:~$ curl -i -H "Content-Type: application/json" -X POST -d '{"1": "
Hello world!"}' http://localhost:5000/api/v1/storage/json/write
HTTP/1.1 200 OK
Server: Werkzeug/2.2.3 Python/3.9.16
Date: Fri, 31 Mar 2023 22:13:19 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 18
Connection: close

Succesfully added!eugene@eugene:~$ 
```



```
eugene@eugene:~$ curl -i -H "Content-Type: application/json" -X POST -d '{"sdf":
 "Hello world!"}' http://localhost:5000/api/v1/storage/json/write
HTTP/1.1 200 OK
Server: Werkzeug/2.2.3 Python/3.9.16
Date: Fri, 31 Mar 2023 22:13:47 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 18
Connection: close

Succesfully added!eugene@eugene:~$ 
```

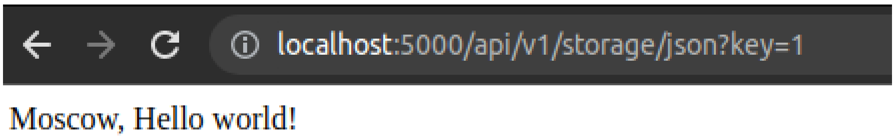
Все данные:



localhost:5000/api/v1/storage/json/all

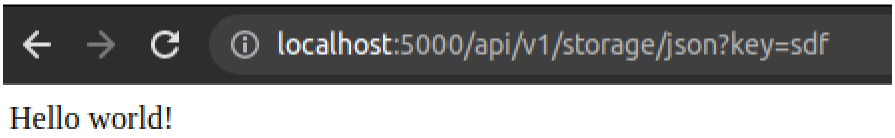
```
{"1":["Moscow","Hello world!"],"sdf":["Hello world!"]}
```

Данные по ключу:



localhost:5000/api/v1/storage/json?key=1

Moscow, Hello world!



localhost:5000/api/v1/storage/json?key=sdf

Hello world!

Задание 2.1

Сделал сразу через плейбук.

Но само задание делается путем изменения файла /etc/nginx/nginx.conf

Для веб-серверов:

```
events {
   worker_connections 768;
}

http {
   server {
       location / {
           return 200 "Hello Word! \n Server 1";
       }
   }
}
```

Для балансировщика на:

```
events {
   worker_connections 768;
}

http {
   upstream web {
       server 192.168.0.136;
       server 192.168.0.137;
   }

   server {
       location / {
           proxy_pass http://web;
       }
   }
}
```

Далее обращаемся к балансировщику, который редиректит на один из серверов.

Скрин результата такой же, как и в задании 2.2.

Задание 2.2

Хосты:

```
[webservers]
```

```
web1 ansible_host=192.168.0.136 ansible_user=vagrant conf_file=nginx1
web2 ansible_host=192.168.0.137 ansible_user=vagrant conf_file=nginx2

[loadbalancers]
lb1 ansible_host=192.168.0.139 ansible_user=vagrant
```

Создаем две роли: set_web и set_balanceloader
Основной плейбук:

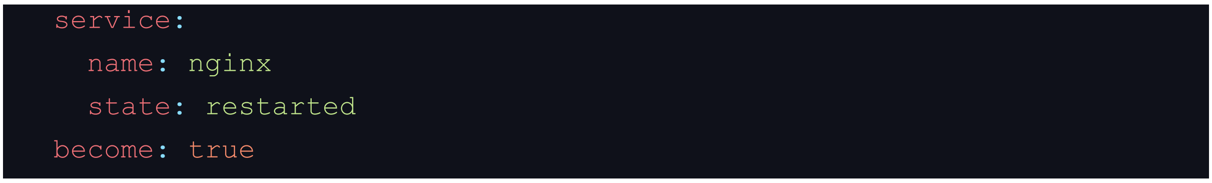
```
---
- hosts: webservers:loadbalancers
 tasks:
   - name: Install nginx
     apt:
       name: nginx
       state: latest
       update_cache: yes
     become: true

- name: Configure webservers
 hosts: webservers
 roles:
   - set_web

- name: Configure loadbalancers
 hosts: loadbalancers
 roles:
   - set_loadbalancer
```

Пример таска из роли set_web:

```
---
 - name: Set nginx.conf on webservers
   copy:
     src: "{{ conf_file }}.conf"
     dest: "{{ dest_path }}"
   become: true

   notify:
     - Restart nginx
```

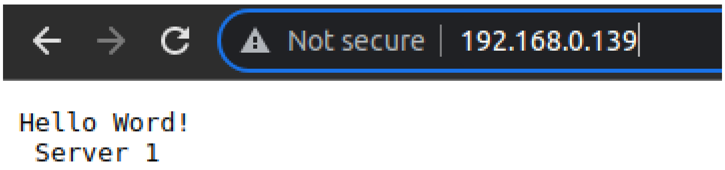
Хендлер:

```
---
 - name: Restart nginx
```

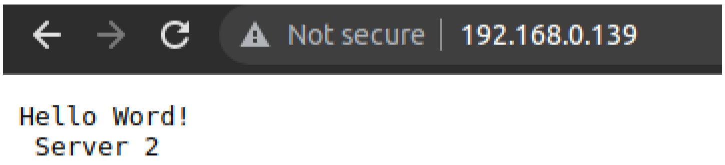


```
  service:
    name: nginx
    state: restarted
  become: true
```

Реузультат обращения к балансировщику:



```
Hello Word!
 Server 1
```



```
Hello Word!
 Server 2
```