

Лабораторная работа №4
Архитектура вычислительных систем

Выполнил:
студент 3 курса Раков Артем

```
server.py
1  from flask import Flask
2  import socket
3
4  app = Flask(__name__)
5
6  @app.route('/')
7  def index():
8      return f"container hostname: {socket.gethostname()}.\\n"
9
10 if __name__ == "__main__":
11     app.run(host="0.0.0.0", port = 5000)
12
13
```

flask webserver, слушает все доступные адреса на 5000 порту

```
Dockerfile
1  FROM python:3.9
2
3  RUN pip install flask
4
5  COPY server.py /server.py
6  WORKDIR /
7
8  CMD ["python", "server.py"]
9
10 EXPOSE 5000
```

Dockerfile для оборачивания flask webserver в контейнер, контейнер разворачивается на порту 5000, как и наш flask webserver, то есть запросы на 5000 порт нашего контейнера будут приходить на flask webserver

Создадим отдельную сеть в докере, внутри нее будут находиться наши контейнеры и прокси-сервер nginx

выполним следующую команду:

`docker network create my_flask_claster_network`, где **my_flask_claster_network** название нашей сети

Соберем docker-образ, для этого нужно находиться в папке с Dockerfile и написать следующую инструкцию:

`docker build -t flask_build`, где **flask_build** будем названием нашего образа

Запустим 2 контейнера:

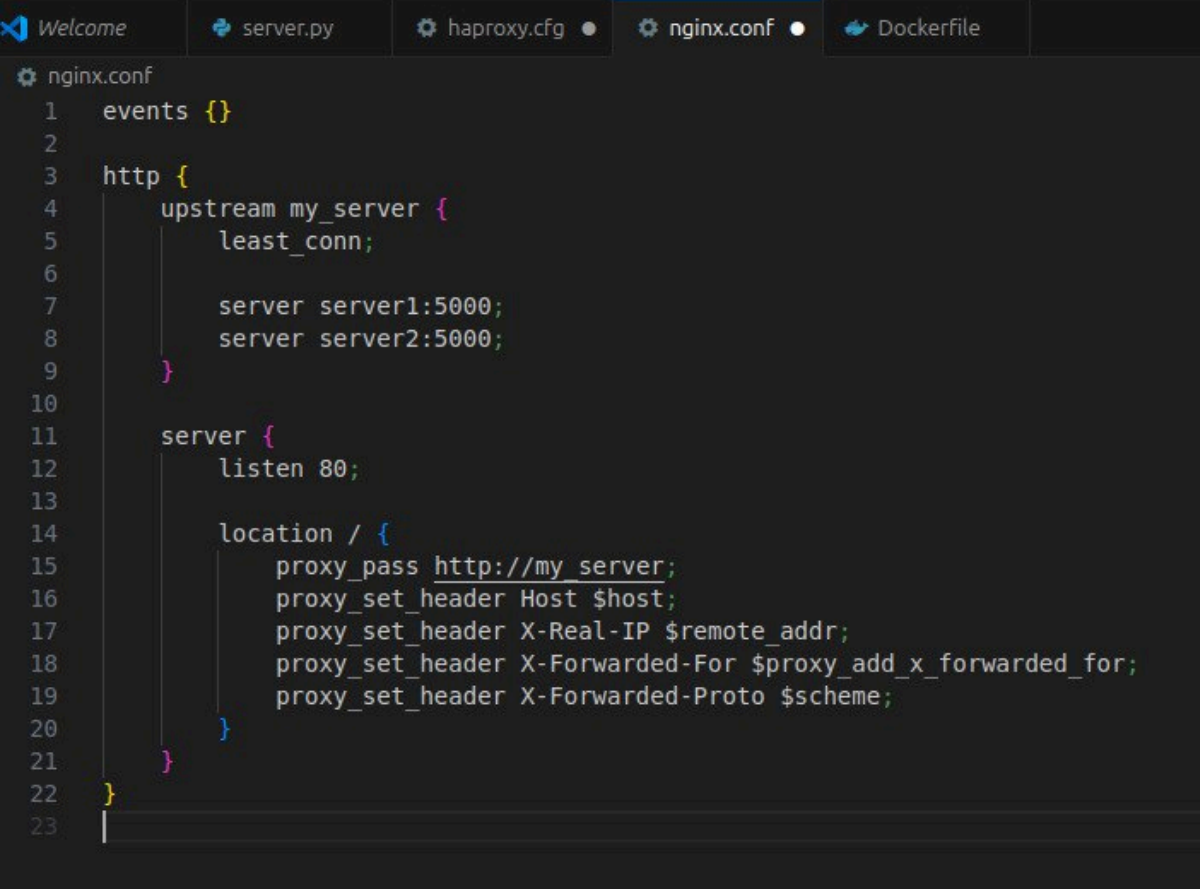
```
root@wireless-monkey-IdeaPad-Gaming-3-151MH05:/home/wireless-monkey/Desktop/acn/Lab4# docker run -d --hostname server1 --name server1 --net my_flask_claster_network flask_build
c786f7992e319d5642ab11dd629d983755bceb7870b1d2a98688a337fe8bbbf
root@wireless-monkey-IdeaPad-Gaming-3-151MH05:/home/wireless-monkey/Desktop/acn/Lab4# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS
c786f7992e31   flask_build  "python server.py"      1 second ago  Up 1 second  5000/tcp
2886f1f0452   nginx       "/docker-entrypoint..." 28 minutes ago Up 28 minutes  0.0.0.0:80->80/tcp, :::80->80/tcp
root@wireless-monkey-IdeaPad-Gaming-3-151MH05:/home/wireless-monkey/Desktop/acn/Lab4# docker run -d --hostname server2 --name server2 --net my_flask_claster_network flask_build
4cf309cfa727a80866089636c3bad3c71082986dc8f7cd0873c747bcs902448e8
root@wireless-monkey-IdeaPad-Gaming-3-151MH05:/home/wireless-monkey/Desktop/acn/Lab4# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS
4cf309cfa727   flask_build  "python server.py"      2 seconds ago  Up 1 second  5000/tcp
c786f7992e31   flask_build  "python server.py"      36 seconds ago Up 35 seconds  5000/tcp
2886f1f0452   nginx       "/docker-entrypoint..." 29 minutes ago Up 29 minutes  0.0.0.0:80->80/tcp, :::80->80/tcp
```

выполним следующие команды:

`docker run -d --hostname server1 --name server1 --net my_flask_claster_network flask_build` , для первого контейнера

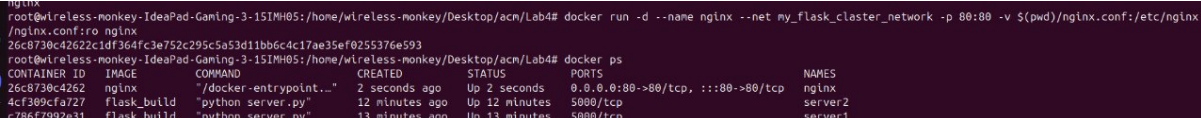
`docker run -d --hostname server2 --name server2 --net my_flask_claster_network flask_build` , для второго контейнера

Теперь напишем файл-конфигурации nginx.conf для нашего прокси сервера:



```
nginx.conf
1  events {}
2
3  http {
4      upstream my_server {
5          least_conn;
6
7          server server1:5000;
8          server server2:5000;
9      }
10
11     server {
12         listen 80;
13
14         location / {
15             proxy_pass http://my_server;
16             proxy_set_header Host $host;
17             proxy_set_header X-Real-IP $remote_addr;
18             proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
19             proxy_set_header X-Forwarded-Proto $scheme;
20         }
21     }
22 }
23
```

Создадим контейнер с нашим прокси-сервером nginx



```
root@wireless-monkey-IdeaPad-Gaming-3-15IMH05:/home/wireless-monkey/Desktop/acn/Lab4# docker run -d --name nginx --net my_flask_cluster_network -p 80:80 -v $(pwd)/nginx.conf:/etc/nginx/nginx.conf:ro nginx
26c8730c42622c1df364fc3e752c295c5a53d11bb6c4c17ae35ef0255376e593
root@wireless-monkey-IdeaPad-Gaming-3-15IMH05:/home/wireless-monkey/Desktop/acn/Lab4# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                               NAMES
26c8730c4262   nginx    "/docker-entrypoint..." 2 seconds ago  Up 2 seconds  0.0.0.0:80->80/tcp, :::80->80/tcp   nginx
4cf309cfa727   flask_bu "python server.py"       12 minutes ago Up 12 minutes  5000/tcp                          server2
c786f7992e31   flask_bu "python server.py"       13 minutes ago Up 13 minutes  5000/tcp                          server1
```

выполним следующую команду:

```
docker run -d --name nginx --net my_flask_claster_network -p 80:80 -v $(pwd)/nginx.conf:/etc/nginx/nginx.conf:ro nginx
```

Теперь отправляя запросы на nginx мы видим работу балансировщика нагрузки по алгоритму round-robin

```
root@wireless-monkey-IdeaPad-Gaming-3-15IMH05:/hone/wireless-monkey/Desktop/acm/Lab4# for i in {1..20}; do curl -s http://localhost:80; done;
container hostname: server2.
container hostname: server1.
container hostname: server2.
container hostname: server1.
container hostname: server2.
container hostname: server1.
container hostname: server2.
container hostname: server1.
container hostname: server2.
container hostname: server1.
container hostname: server2.
container hostname: server1.
container hostname: server2.
container hostname: server1.
container hostname: server2.
container hostname: server1.
container hostname: server2.
container hostname: server1.
container hostname: server2.
container hostname: server1.
```

запустим контейнер с haproxy:

для этого напишем файл конфигурации haproxy.cfg

```
Welcome  server.py  haproxy.cfg  nginx.conf  Dockerfile
haproxy.cfg
1  global
2      daemon
3      maxconn 256
4
5  defaults
6      mode http
7      timeout connect 5000ms
8      timeout client 10000ms
9      timeout server 10000ms
10
11 frontend http_front
12     bind *:8081
13     default_backend flask_cluster
14
15 backend flask_cluster
16     balance roundrobin
17     server fserver1 172.18.0.2:5000 check
18     server fserver2 172.18.0.3:5000 check
19
```

запустим контейнер с haproxy:

```
root@wireless-monkey-IdeaPad-Gaming-3-15IMH05:/home/wireless-monkey/Desktop/acn/Lab4# docker run -d --name haproxy --net my_flask_cluster_network -p 8081:8081 -v $(pwd)/haproxy.cfg:/usr/local/etc/haproxy/haproxy.cfg:ro haproxy
5ad81b84ce2665eb792991ecc0fa9b8d26187b2a89666cf669f2bcd5b51a2a7
root@wireless-monkey-IdeaPad-Gaming-3-15IMH05:/home/wireless-monkey/Desktop/acn/Lab4# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                               NAMES
5ad81b84ce26   haproxy   "/docker-entrypoint.s..." 2 seconds ago  Up 1 second   0.0.0.0:8081->8081/tcp, :::8081->8081/tcp  haproxy
26c8738c4262   nginx     "/docker-entrypoint..." 40 minutes ago  Up 40 minutes  0.0.0.0:80->80/tcp, :::80->80/tcp          nginx
4cf309cfa727   flask_bui "python server.py"        53 minutes ago  Up 53 minutes  5000/tcp                                server2
c786f7992e31   flask_bui "python server.py"        54 minutes ago  Up 54 minutes  5000/tcp                                server1
```

для этого нужно выполнить следующую команду:

```
docker run -d --name haproxy --net my_flask_cluster_network -p
8081:8081 -v $(pwd)/haproxy.cfg/usr/local/etc/haproxy/haproxy.cfg:ro
haproxy
```

Проверим работу балансировщика отправив на его адрес несколько запросов:

[illegible]