Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Кафедра вычислительных систем

Лабораторная работа № 4

Выполнил:

студент группы ИВ-121:

Ермаков А. В.

Работу проверил:

Романюта А.А.

Новосибирск 2023 г.

Содержание

Задание 3
Разработка файла docker-compose 4
Создание отдельного сервиса Nginx 11
Увеличение числа экземпляров docker-приложений 17

Задание

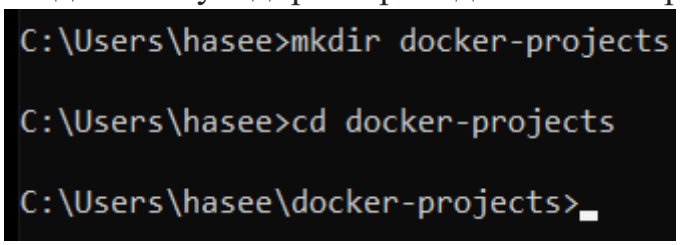
1. Разработать docker-compose файл запускающий несколько экземпляров docker-приложений с веб-сервером (например, traefik/whoami). Убедиться в работоспособности. Провести нагрузочное тестирование, используя сетевой бенчмарк (например, Apache Benchmark, ab).

2. Добавить в docker-compose файл из п.1 прокси-сервер на базе Nginx, реализующий переадресацию запросов и отдачу контента от docker-приложений, при обращении по домену/IP адресу на порт 80 хоста. Убедиться в работоспособности. Провести нагрузочное тестирование.

3. Увеличить число экземпляров docker-приложений с веб-сервером до максимально возможного. Убедиться в соответствии теоретического оценки практическим результатам. Продемонстрировать работу.

Разработка файла docker-compose

Docker Compose по умолчанию устанавливается с Docker Desktop, поэтому создаем новую директорию для нашего проекта:



mkdir docker-projects

cd docker-projects

В этой директории создаем файл docker-compose.yaml, в который запихиваем содержимое:

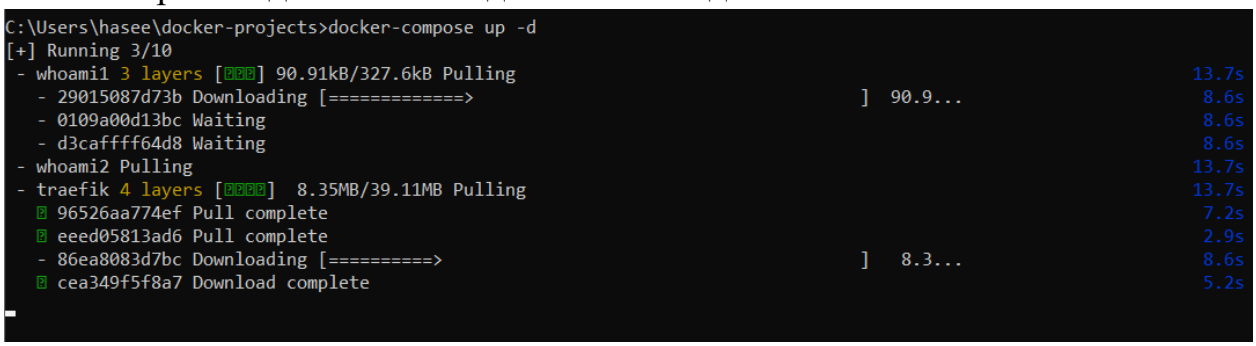
```
version: '3'


services:
  web1:
    image: nginx
    ports:
      - "5000:80"

  web2:
    image: nginx
    ports:
      - "5001:80"

  web3:
    image: nginx
    ports:
      - "5002:80"
```

```
  web4:
    image: nginx
    ports:
      - "5003:80"

  web5:
    image: nginx
    ports:
      - "5004:80"
```

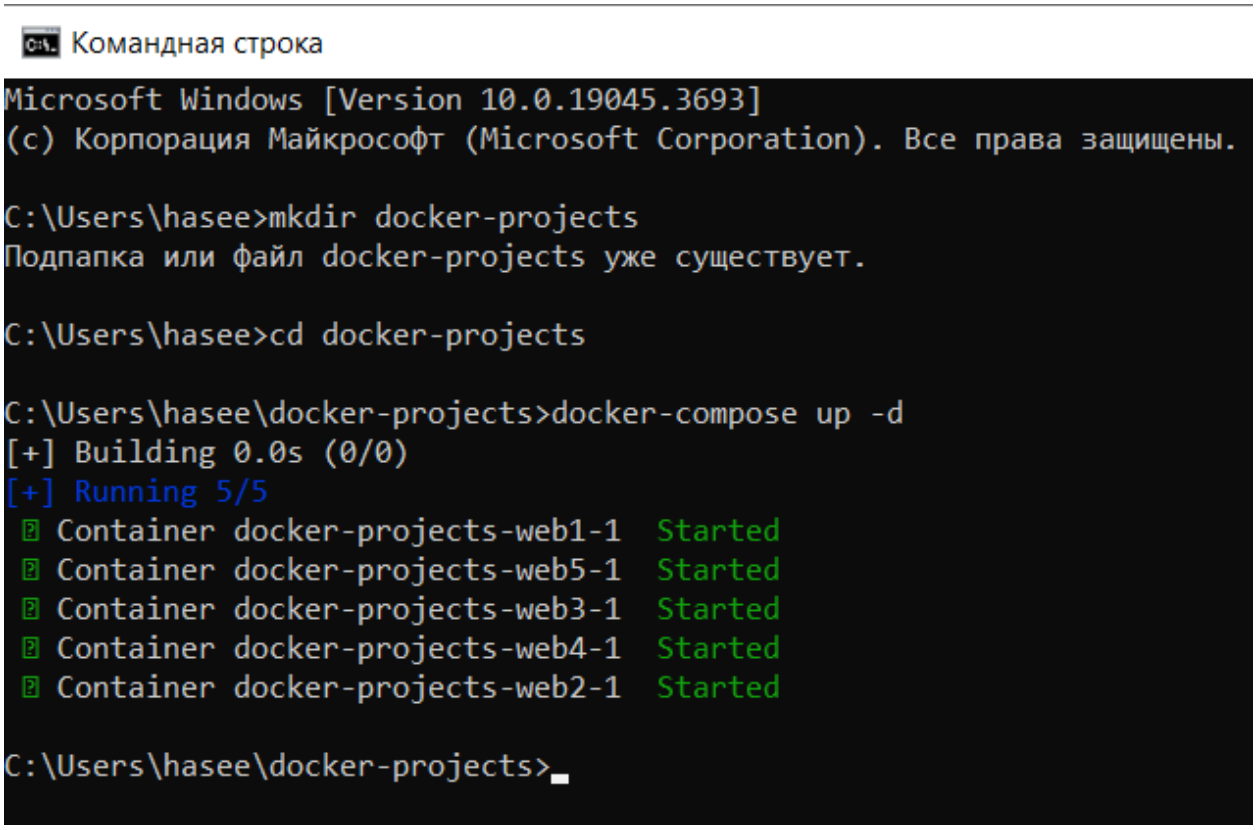
В этой же директории выполняем следующую команду:
docker-compose up –d

Эта команда создаст и запустит контейнеры на основе файла docker-compose.yaml. Флаг -d означает, что контейнеры будут запущены в фоновом режиме.

Вот что происходит после ввода этой команды:

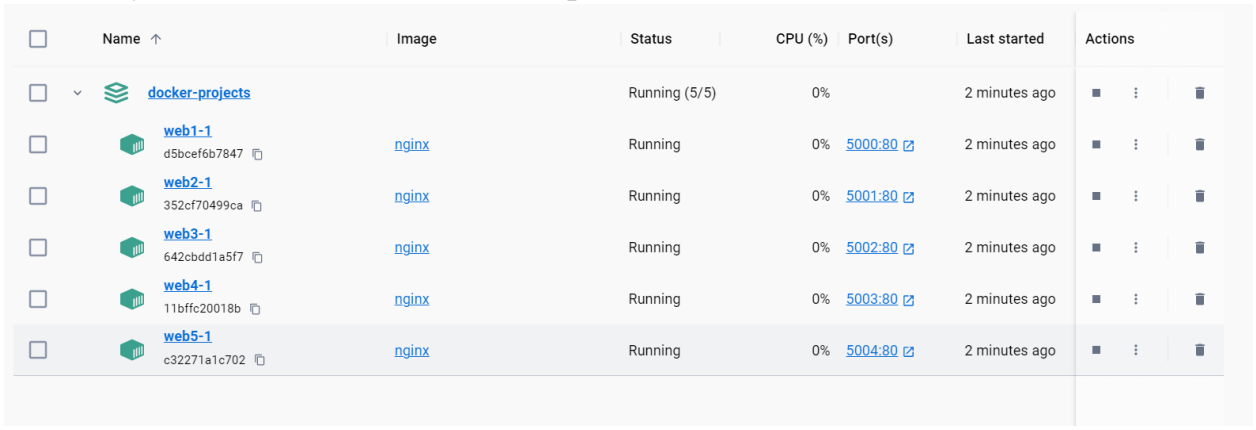


это создание рандомного контейнера.



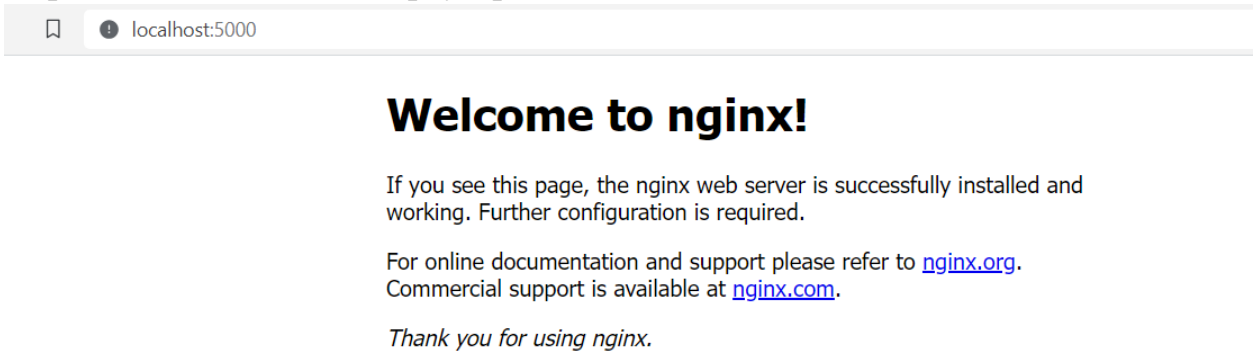
это уже нужные в лабе запущенные контейнеры.

По итогу глядим на наши контейнеры:



Видим, что все 5 контейнеров успешно запустились.

Переходим по ссылке в браузер:



На других контейнерах аналогично.

Нагрузим наши контейнеры немножечко:

```
C:\Users\hasee\docker-projects>ab -n 1000 -c 10 http://localhost:5003/
This is ApacheBench, Version 2.3 <$Revision: 1903618 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests


Server Software:
Server Hostname:        localhost
Server Port:            5003

Document Path:          /
Document Length:        182 bytes

Concurrency Level:      10
Time taken for tests:   2.982 seconds
Complete requests:      1000
Failed requests:        0
Total transferred:      300000 bytes
HTML transferred:       182000 bytes
Requests per second:    335.31 [#/sec] (mean)
Time per request:       29.823 [ms] (mean)
Time per request:       2.982 [ms] (mean, across all concurrent requests)
Transfer rate:          98.24 [Kbytes/sec] received
```

```
Percentage of the requests served within a certain time (ms)
  50%     33
  66%     36
  75%     38
  80%     47
  90%     68
  95%     71
  98%     78
  99%     95
 100%    135 (longest request)
```

И у каждого по 12 процессов:

```
2023-11-29 22:33:12 2023/11/29 15:33:12 [notice] 1#1: start worker processes
2023-11-29 22:33:12 2023/11/29 15:33:12 [notice] 1#1: start worker process 29
2023-11-29 22:33:12 2023/11/29 15:33:12 [notice] 1#1: start worker process 30
2023-11-29 22:33:12 2023/11/29 15:33:12 [notice] 1#1: start worker process 31
2023-11-29 22:33:12 2023/11/29 15:33:12 [notice] 1#1: start worker process 32
2023-11-29 22:33:12 2023/11/29 15:33:12 [notice] 1#1: start worker process 33
2023-11-29 22:33:12 2023/11/29 15:33:12 [notice] 1#1: start worker process 34
2023-11-29 22:33:12 2023/11/29 15:33:12 [notice] 1#1: start worker process 35
2023-11-29 22:33:12 2023/11/29 15:33:12 [notice] 1#1: start worker process 36
2023-11-29 22:33:12 2023/11/29 15:33:12 [notice] 1#1: start worker process 37
2023-11-29 22:33:12 2023/11/29 15:33:12 [notice] 1#1: start worker process 38
2023-11-29 22:33:12 2023/11/29 15:33:12 [notice] 1#1: start worker process 39
2023-11-29 22:33:12 2023/11/29 15:33:12 [notice] 1#1: start worker process 40
2023-11-29 22:33:17 2023/11/29 15:33:17 [notice] 1#1: signal 3 (SIGQUIT) received, shutting down
2023-11-29 22:33:17 2023/11/29 15:33:17 [notice] 29#29: gracefully shutting down
2023-11-29 22:33:17 2023/11/29 15:33:17 [notice] 31#31: gracefully shutting down
2023-11-29 22:33:17 2023/11/29 15:33:17 [notice] 30#30: gracefully shutting down
```

C:\Users\hasee\docker-projects>ab -n 30000 -c 13000 http://localhost/

This is ApacheBench, Version 2.3 <$Revision: 1903618 $>

Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/

Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)

Completed 3000 requests

Completed 6000 requests

Completed 9000 requests

Completed 12000 requests

Test aborted after 10 failures

apr_socket_connect(): ╧юфъы■ўхэшх эх єёЄрэютыхэю, Є.ъ. ъюэхўэ√щ ъюья№■ЄхЁ юЄтхЁу чряЁюё эр яюфъы■ўхэшх. (730061)

Total of 13645 requests completed

C:\Users\hasee\docker-projects>

Отсюда видно, что 13645 процессов завершились успешно, что совпадает с теоретической оценкой.

Создание отдельного сервиса Nginx

Для реализации прокси-сервера на базе Nginx внутри Docker Compose, создадим отдельный сервис для Nginx и настроим его для обработки запросов и переадресации к вашим Docker-приложениям:

```
version: '3'

services:
  web1:
    image: nginx
    ports:
      - "5000:80"

  web2:
    image: nginx
    ports:
      - "5001:80"

  web3:
    image: nginx
    ports:
      - "5002:80"

  web4:
    image: nginx
    ports:
      - "5003:80"

  web5:
    image: nginx
```

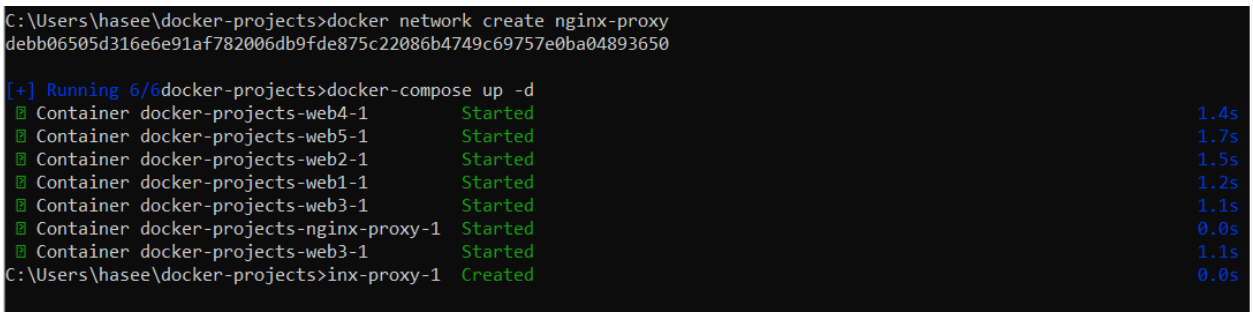
```
    ports:
      - "5004:80"


  nginx-proxy:
    image: nginx
    ports:
      - "80:80"
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf
    depends_on:
      - web1
      - web2
      - web3
      - web4
      - web5
```

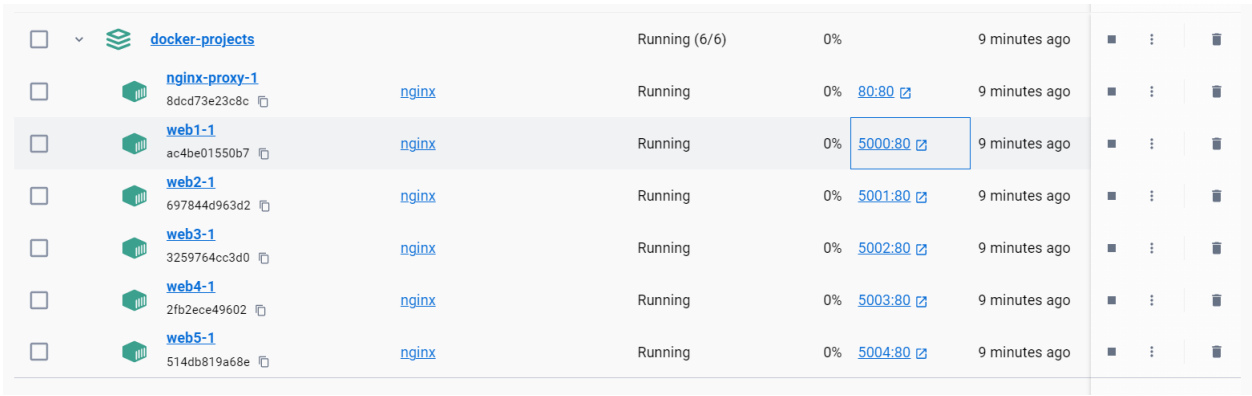
Таким образом мы создаем отдельный сервис nginx-proxy, который проксирует запросы с хоста на порт 80 ко всем контейнерам web1, web2, web3, web4 и web5.

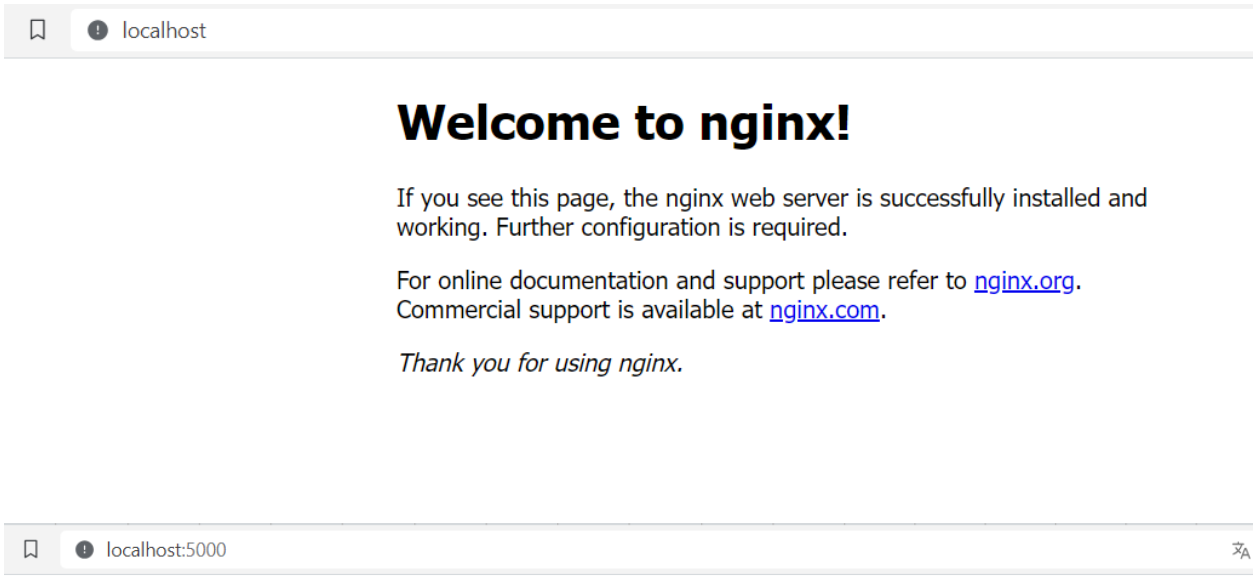
Далее создаем файл nginx.conf в папке nginx (в той же директории, где находится ваш docker-compose.yml), чтобы обеспечить правильное перенаправление запросов к соответствующему сервису:

```
http {
    sendfile on;


    upstream backend {
        server web1:80;
        server web2:80;
        server web3:80;
```

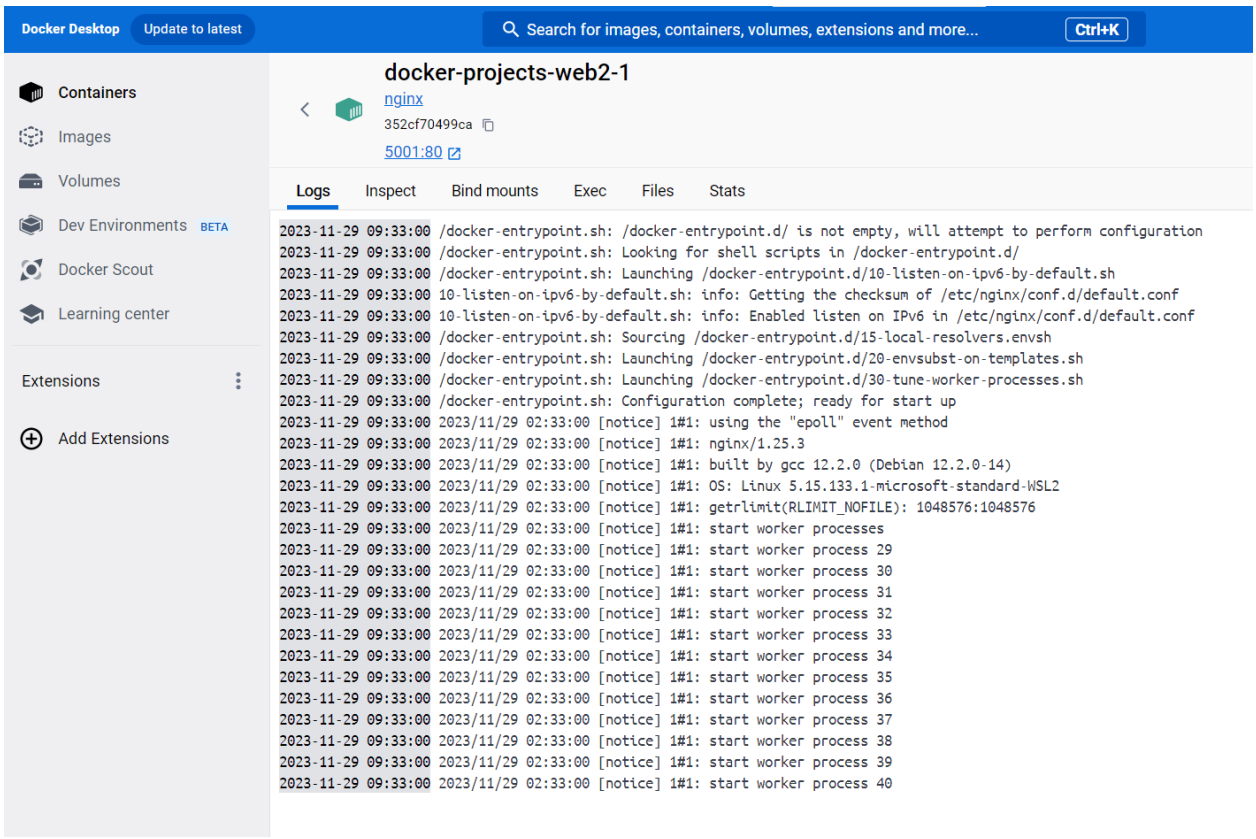
```
        server web4:80;
        server web5:80;
    }

    server {
        listen 80;

        location / {
            proxy_pass http://backend;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }
    }
}
```







Посмотрим количество процессов на каждом из контейнеров:



2023-11-29 09:33:00 2023/11/29 02:33:00 [notice] 1#1: start worker processes
2023-11-29 09:33:00 2023/11/29 02:33:00 [notice] 1#1: start worker process 29
2023-11-29 09:33:00 2023/11/29 02:33:00 [notice] 1#1: start worker process 30
2023-11-29 09:33:00 2023/11/29 02:33:00 [notice] 1#1: start worker process 31
2023-11-29 09:33:00 2023/11/29 02:33:00 [notice] 1#1: start worker process 32
2023-11-29 09:33:00 2023/11/29 02:33:00 [notice] 1#1: start worker process 33
2023-11-29 09:33:00 2023/11/29 02:33:00 [notice] 1#1: start worker process 34
2023-11-29 09:33:00 2023/11/29 02:33:00 [notice] 1#1: start worker process 35
2023-11-29 09:33:00 2023/11/29 02:33:00 [notice] 1#1: start worker process 36
2023-11-29 09:33:00 2023/11/29 02:33:00 [notice] 1#1: start worker process 37
2023-11-29 09:33:00 2023/11/29 02:33:00 [notice] 1#1: start worker process 38
2023-11-29 09:33:00 2023/11/29 02:33:00 [notice] 1#1: start worker process 39
2023-11-29 09:33:00 2023/11/29 02:33:00 [notice] 1#1: start worker process 40

Количество процессов на всех контейнерах 12, теперь в файле конфигурации nginx посмотрим на максимальное количество соединений для одного процесса:

```
events {
    worker_connections  1024;
}
```

Теперь у нас есть 5 экземпляров, 12 процессов на каждом и 1024 ограничение для одного процесса.
По итогу получаем 5 * 12 * 1024 = 61440 запросов теоретически.

Увеличение числа экземпляров docker-приложений

Теоретически, если мы в nginx-proxy имеем 4 работающих контейнера в 1 случае, 8 в другом, то при одинаковой нагрузке nginx-proxy на 4 контейнерах будет работать в 2 раза дольше, чем на 8 процессах.

Поэтому для начала проведем тестирование на 8 контейнерах:

User

C:\Users\hasee\docker-projects>ab -n 100000 -c 10000 http://localhost/

This is ApacheBench, Version 2.3 <$Revision: 1903618 $>

Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/

Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)

Completed 10000 requests

Completed 20000 requests

Completed 30000 requests

Completed 40000 requests

Completed 50000 requests

Completed 60000 requests

Completed 70000 requests

Completed 80000 requests

Completed 90000 requests

Completed 100000 requests

Finished 100000 requests

Server Software: Apache/2.4.58

Server Hostname: localhost

Server Port: 80

Document Path: /
Document Length: 45 bytes

Concurrency Level: 10000
Time taken for tests: 41.966 seconds
Complete requests: 100000
Failed requests: 0
Total transferred: 28900000 bytes
HTML transferred: 4500000 bytes
Requests per second: 2382.86 [#/sec] (mean)
Time per request: 4196.633 [ms] (mean)
Time per request: 0.420 [ms] (mean, across all concurrent requests)
Transfer rate: 672.51 [Kbytes/sec] received

Connection Times (ms)

	min	mean[+/-sd]	median	max
Connect:	0	0 0.4	0	3
Processing:	1223	4017 616.0	4047	4783
Waiting:	3	2058 1132.4	2019	4606
Total:	1223	4017 616.0	4048	4783

Percentage of the requests served within a certain time (ms)
50% 4048
66% 4389
75% 4552
80% 4594
90% 4647

95% 4702
98% 4760
99% 4774
100% 4783 (longest request)

C:\Users\hasee\docker-projects>

Время на 8 контейнерах составило 41.966 секунд.

Теперь проведем тестирование на 8 контейнерах:

C:\Users\hasee\docker-projects>ab -n 100000 -c 10000 http://localhost/
This is ApacheBench, Version 2.3 <$Revision: 1903618 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 10000 requests
Completed 20000 requests
Completed 30000 requests
Completed 40000 requests
Completed 50000 requests
Completed 60000 requests
Completed 70000 requests
Completed 80000 requests
Completed 90000 requests
Completed 100000 requests
Finished 100000 requests

Server Software: Apache/2.4.58
Server Hostname: localhost
Server Port: 80

Document Path: /
Document Length: 45 bytes

Concurrency Level: 10000
Time taken for tests: 87.067 seconds
Complete requests: 100000
Failed requests: 10
(Connect: 10, Receive: 0, Length: 0, Exceptions: 0)
Total transferred: 28900000 bytes
HTML transferred: 4500000 bytes
Requests per second: 1148.54 [#/sec] (mean)
Time per request: 8706.718 [ms] (mean)
Time per request: 0.871 [ms] (mean, across all concurrent requests)
Transfer rate: 324.15 [Kbytes/sec] received

Connection Times (ms)

	min	mean[+/-sd]	median	max
Connect:	0	0 11.1	0	1535
Processing:	1234	8518 7581.4	4885	31058
Waiting:	2	5248 7228.9	2693	30143
Total:	1235	8518 7581.6	4886	31059

Percentage of the requests served within a certain time (ms)
50% 4886

66% 7078

75% 9428

80% 11568

90% 16569

95% 29194

98% 30397

99% 30516

100% 31059 (longest request)

Время на 4 контейнерах составило 87.067 секунд.
Несложно посчитать, что с увеличением числа контейнеров в 2 раза время уменьшилось в 87.067 / 41.966 = 2.0747, что совпадает с значением, полученным теоретически.