

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

Кафедра вычислительных систем

Лабораторная работа № 3

Выполнил:

студент группы ИВ-121:

Ермаков А. В.

Работу проверил:

Романюта А.А.

Новосибирск 2023 г.

Содержание

Задание	3
Доступ к веб-серверу внутри контейнера из других сетей	4
Запуск веб-сервера	8
Нагрузочное тестирование веб-сервера.....	12

Задание

- а) Запустить в контейнере веб сервер. Допускается использовать готовые образы. Примеры веб-серверов - nginx, apache, примеры из boost.asio).
- б) Разрешить доступ к веб-серверу внутри контейнера из других сетей (--publish).
- в) Используя сетевой бенчмарк (например, Apache Benchmark, **ab**) провести нагрузочное тестирование веб-сервера.

Доступ к веб-серверу внутри контейнера из других сетей

Для начала убедимся, что у нас установлено необходимые программы:

```
Командная строка
Microsoft Windows [Version 10.0.19045.3570]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\hasee>docker --version
Docker version 24.0.6, build ed223bc

C:\Users\hasee>_
```

Создадим новую директорию для Docker, в которой создадим необходимый Dockerfile:

```
1 FROM nginx:latest
2 COPY . /usr/share/nginx/html
3 EXPOSE 80
```

Давайте рассмотрим его содержимое:

- 1) **FROM nginx:latest** - Эта строка указывает базовый образ для Docker-образа. В данном случае, я решил использовать образ Nginx с тегом "latest" (последней версии).
- 2) **COPY . /usr/share/nginx/html** - Эта строка копирует все файлы и каталоги из текущего контекста сборки (где находится Dockerfile) внутрь контейнера по пути /usr/share/nginx/html, в контексте Nginx это обычно используется для размещения статических веб-страниц.
- 3) **EXPOSE 80** - Эта строка указывает Docker, что контейнер будет слушать сетевой трафик на порту 80 (эта инструкция не открывает порт 80 на хосте, она просто документирует факт, что контейнер использует этот порт).

Далее запускаем приложение Docker Desktop и в командной строке создаем Docker образ с помощью команды:

```
1 docker build -t my-nginx .
```

```
Командная строка
C:\Users\hasee\Desktop\5sem\авс\lab3\Docker>docker build -t my-nginx .
[+] Building 16.7s (7/7) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 95B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/nginx:latest
=> [internal] load build context
=> => transferring context: 95B
=> [1/2] FROM docker.io/library/nginx:latest@sha256:86e53c4c16a6a276b204b0fd3a8143d86547c967dc8258b3d47c3a21bb68d3c6
=> => resolve docker.io/library/nginx:latest@sha256:86e53c4c16a6a276b204b0fd3a8143d86547c967dc8258b3d47c3a21bb68d3c6
=> => sha256:86e53c4c16a6a276b204b0fd3a8143d86547c967dc8258b3d47c3a21bb68d3c6 1.86kB / 1.86kB
=> => sha256:d2e65182b5fd330470eca9b8e23e8a1a0d87cc9b820eb1fb3f034bf8248d37ee 1.78kB / 1.78kB
=> => sha256:e398db710407fbc310b4bc0b0db1c94161480ac9b44638c6655939f426529780 41.38MB / 41.38MB
=> => sha256:c20060033e06f882b0fbedb7d974d72e0887a3be5e554efdb0dcf8d53512647 8.15kB / 8.15kB
=> => sha256:578acb154839e9d0034432e8f53756d6f53ba62cf8c7ea5218a2476bf5b58fc9 29.15MB / 29.15MB
=> => sha256:85c41ebe6d660b75d8e2e985314ebce75e602330cd325bc5cfbf9d9723c329a1 627B / 627B
=> => sha256:7170a263b582e6a7b5f650b7f1c146267f694961f837ffefc2505bb9148dd4b0 958B / 958B
=> => sha256:8f28d06e2e2ec58753e1acf21d96619aafcab87e63e06fb0590f56091db38014 367B / 367B
=> => sha256:6f837de2f88742f4e8083bff54bd1c64c1df04e6679c343d1a1c9a650078fd48 1.21kB / 1.21kB
=> => sha256:c1dfc7e1671e8340003503af03d067bae6846c12c30cbc1af3e589cb124fd45d 1.40kB / 1.40kB
=> => extracting sha256:578acb154839e9d0034432e8f53756d6f53ba62cf8c7ea5218a2476bf5b58fc9
=> => extracting sha256:e398db710407fbc310b4bc0b0db1c94161480ac9b44638c6655939f426529780
=> => extracting sha256:85c41ebe6d660b75d8e2e985314ebce75e602330cd325bc5cfbf9d9723c329a1
=> => extracting sha256:7170a263b582e6a7b5f650b7f1c146267f694961f837ffefc2505bb9148dd4b0
=> => extracting sha256:8f28d06e2e2ec58753e1acf21d96619aafcab87e63e06fb0590f56091db38014
=> => extracting sha256:6f837de2f88742f4e8083bff54bd1c64c1df04e6679c343d1a1c9a650078fd48
=> => extracting sha256:c1dfc7e1671e8340003503af03d067bae6846c12c30cbc1af3e589cb124fd45d
=> [2/2] COPY . /usr/share/nginx/html
=> exporting to image
=> => exporting layers
=> => writing image sha256:eeb661fca5b42a2e3c062a9e0aac69c636297fb88edca472d24b197f8198ae36
=> => naming to docker.io/library/my-nginx
What's Next?
View a summary of image vulnerabilities and recommendations -> docker scout quickview
C:\Users\hasee\Desktop\5sem\авс\lab3\Docker>
```

Таким образом создаем образ с именем my-nginx.

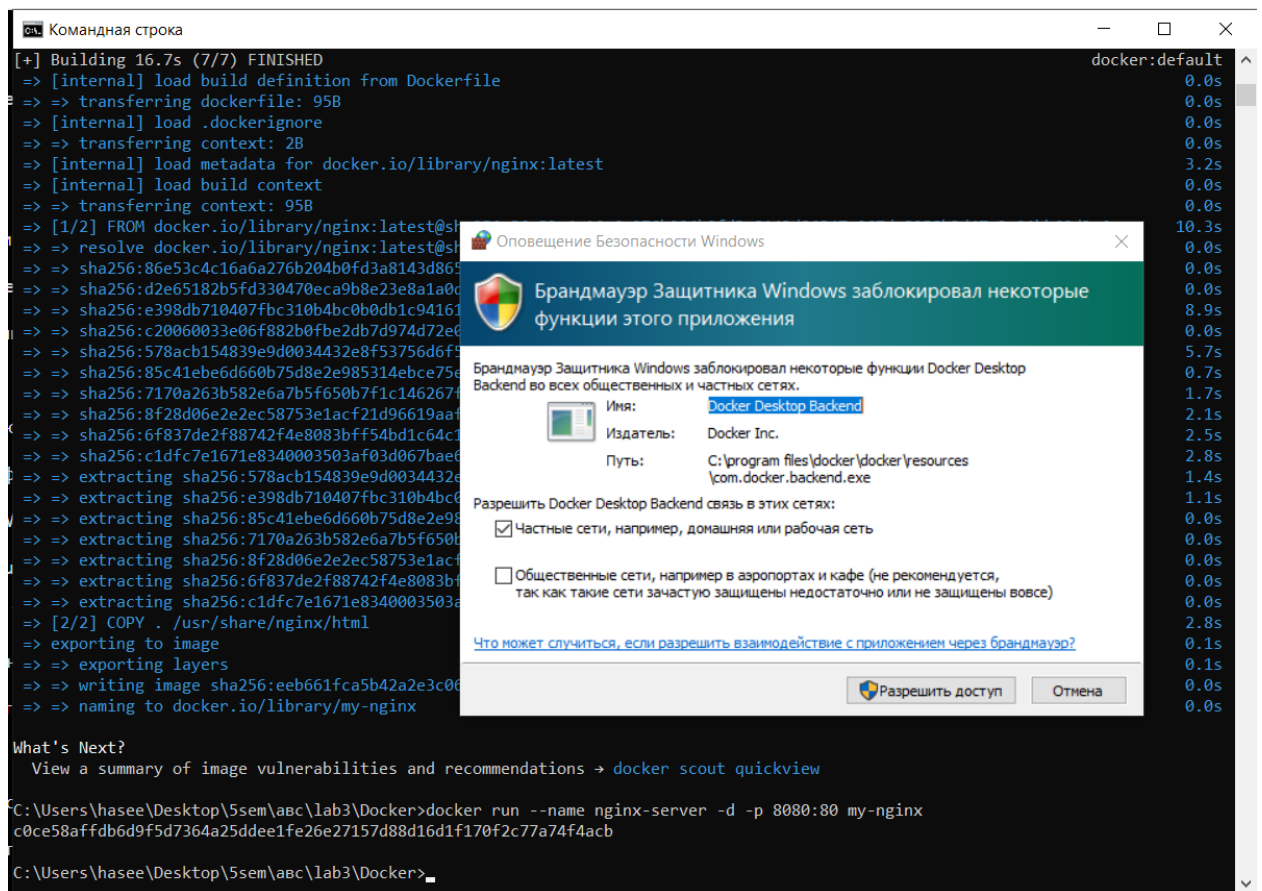
После чего запускаем контейнер из созданного образа следующей командой:

1	<code>docker run --name nginx-server -d -p 8080:80 my-nginx</code>
---	--

Рассмотрим, что делает эта команда:

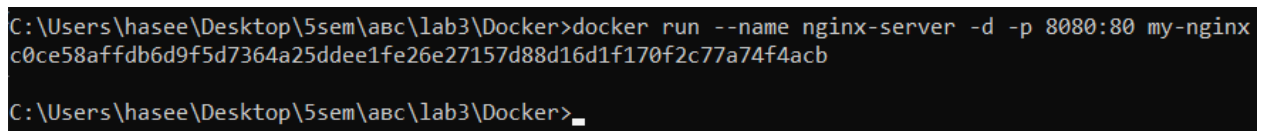
- 1) **docker** - командная утилита Docker, которая взаимодействует с Docker-демоном для управления контейнерами и образами.
- 2) **run** - подкоманда Docker, используемая для запуска контейнера из Docker-образа.
- 3) **--name nginx-server** - этот флаг позволяет вам задать имя (nginx-server) для контейнера.
- 4) **-d** - флаг, который указывает Docker запустить контейнер в фоновом режиме (detach mode), что означает, что контейнер будет работать в фоновом режиме, и управление консолью вернется нам.
- 5) **-p** - флаг, который используется для проброса портов между хостом и контейнером. В данном случае, порт 8080 на хосте будет привязан к порту 80 внутри контейнера. Таким образом, мы сможем обращаться к веб-серверу в контейнере через порт 8080 на нашем хосте.

6) **my-nginx** - имя Docker-образа, который мы хотим запустить в виде контейнера.



The screenshot shows a Windows command prompt window titled "Командная строка". The terminal output shows the process of building a Docker image named "my-nginx" from a Dockerfile. The build process includes steps like "Building 16.7s (7/7) FINISHED", "load build definition from Dockerfile", "transferring dockerfile: 95B", "load .dockerignore", "transferring context: 2B", "load metadata for docker.io/library/nginx:latest", "load build context", "transferring context: 95B", "FROM docker.io/library/nginx:latest@sha256:...", "extracting sha256:...", "exporting to image", "writing image sha256:...", and "naming to docker.io/library/my-nginx". A Windows Security warning dialog is overlaid on the terminal, titled "Оповещение Безопасности Windows". The dialog text says: "Брандмауэр Защитника Windows заблокировал некоторые функции этого приложения. Брандмауэр Защитника Windows заблокировал некоторые функции Docker Desktop Backend во всех общественных и частных сетях." It lists the application name as "Docker Desktop Backend", publisher as "Docker Inc.", and path as "C:\program files\docker\docker\resources\com.docker.backend.exe". There are checkboxes for "Частные сети, например, домашняя или рабочая сеть" (checked) and "Общественные сети, например в аэропортах и кафе (не рекомендуется, так как такие сети зачастую защищены недостаточно или не защищены вовсе)". At the bottom, there are buttons for "Разрешить доступ" and "Отмена". Below the dialog, the terminal shows "What's Next? View a summary of image vulnerabilities and recommendations → docker scout quickview" and the command "C:\Users\hasee\Desktop\5sem\abc\lab3\docker>docker run --name nginx-server -d -p 8080:80 my-nginx c0ce58affdb6d9f5d7364a25ddee1fe26e27157d88d16d1f170f2c77a74f4acb".

Разрешаем использовать функции Docker Desktop и получаем:

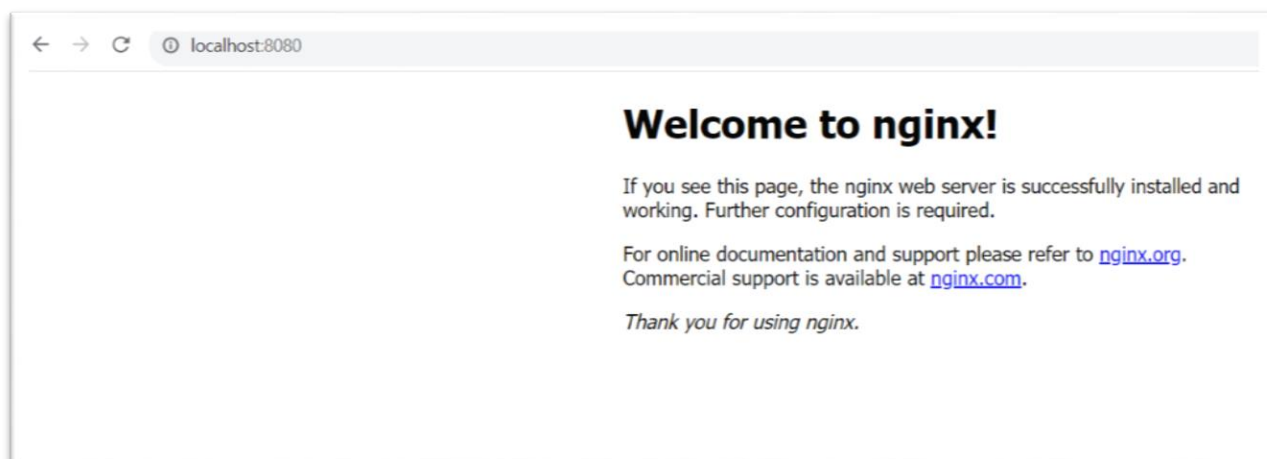


The screenshot shows a Windows command prompt window titled "Командная строка". The terminal output shows the command "C:\Users\hasee\Desktop\5sem\abc\lab3\docker>docker run --name nginx-server -d -p 8080:80 my-nginx c0ce58affdb6d9f5d7364a25ddee1fe26e27157d88d16d1f170f2c77a74f4acb" being executed successfully. The prompt then shows "C:\Users\hasee\Desktop\5sem\abc\lab3\docker>".

Теперь, чтобы проверить работу веб-сервера, в браузере переходим по следующему адресу:

<http://localhost:8080>

Как результат страница приветствия прекрасно отображается, а это лишь означает, что контейнер запущен и работает корректно:



Запуск веб-сервера

До этого мы использовали проброс портов, фактически выполнив пункт «б», поэтому запускаем контейнер без проброса портов следующей командой:

```
1 docker run --name my-nginx -d nginx
```

```
C:\Users\hasee\Desktop\5sem\abc\lab3\Docke>docker run --name my-nginx -d nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
578acb154839: Already exists
e398db710407: Already exists
85c41ebe6d66: Already exists
7170a263b582: Already exists
8f28d06e2e2e: Already exists
6f837de2f887: Already exists
c1dfc7e1671e: Already exists
Digest: sha256:86e53c4c16a6a276b204b0fd3a8143d86547c967dc8258b3d47c3a21bb68d3c6
Status: Downloaded newer image for nginx:latest
6e73877de12a993665dd98e49ca55dd548fcae387e990ebdd2ac3756612b2409
```

Проверяем, запущен ли контейнер:

```
1 docker ps
```

```
C:\Users\hasee\Desktop\5sem\abc\lab3\Docke>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                NAMES
6e73877de12a   nginx    "/docker-entrypoint...." 19 seconds ago Up 11 seconds  80/tcp              my-nginx
c0ce58affdb6   my-nginx "/docker-entrypoint...." 32 minutes ago Up 31 minutes   0.0.0.0:8080->80/tcp  nginx-server
```

Поскольку порты не проброшены, доступ к веб-серверу с хост-машины невозможен напрямую, поэтому мы будем взаимодействовать с веб-сервером внутри контейнера другим способом, для этого подключимся к самому контейнеру (открытие shell) следующей командой:

```
1 docker exec -it my-nginx /bin/bash
```

Давайте разберем эту команду:

- 1) **docker exec** - используется для выполнения команд внутри работающего контейнера.
- 2) **-it** – флаг, который обеспечивает интерактивный режим (с возможностью ввода с клавиатуры) и присоединение к стандартному потоку ввода/вывода (stdin, stdout) контейнера.
- 3) **my-nginx** - имя или идентификатор контейнера, внутри которого выполняется команда.

4) **/bin/bash** - команда, которую мы хотим выполнить внутри контейнера. В данном случае, мы запускаем интерактивную оболочку Bash внутри контейнера.

```
C:\Users\hasee\Desktop\5sem\авс\lab3\Docker>docker exec -it my-nginx /bin/bash
root@6e73877de12a:/#
```

Вот такой командой устанавливаем “Curl”:

1	<code>apt-get update && apt-get install curl</code>
---	---

```
C:\Users\hasee\Desktop\5sem\авс\lab3\Docker>docker exec -it my-nginx /bin/bash
root@6e73877de12a:/# apt-get update && apt-get install curl
Get:1 http://deb.debian.org/debian bookworm InRelease [151 kB]
Get:2 http://deb.debian.org/debian bookworm-updates InRelease [52.1 kB]
Get:3 http://deb.debian.org/debian-security bookworm-security InRelease [48.0 kB]
Get:4 http://deb.debian.org/debian bookworm/main amd64 Packages [8780 kB]
Get:5 http://deb.debian.org/debian bookworm-updates/main amd64 Packages [6668 B]
Get:6 http://deb.debian.org/debian-security bookworm-security/main amd64 Packages [96.6 kB]
Fetched 9135 kB in 5s (1971 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
curl is already the newest version (7.88.1-10+deb12u4).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@6e73877de12a:/#
```

После чего выполняем локальный запрос к веб-серверу:

1	<code>curl http://localhost:80</code>
---	---------------------------------------

```
root@6e73877de12a:/# curl http://localhost:80
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
root@6e73877de12a:/#
```

Таким образом мы получаем HTML-код стандартной страницы приветствия nginx, который точно такой же, как и в браузере:

```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    <h1>Welcome to nginx!</h1>
    <p>
      "If you see this page, the nginx web server is successfully installed and
      working. Further configuration is required."
    </p>
    <p>
      "For online documentation and support please refer to "
      <a href="http://nginx.org/">nginx.org</a>
      ". "
      <br>
      " Commercial support is available at "
      <a href="http://nginx.com/">nginx.com</a>
      ". "
    </p>
    ... <p> == $0
      <em>Thank you for using nginx.</em>
    </p>
  </body>
</html>
```

Нагрузочное тестирование веб-сервера

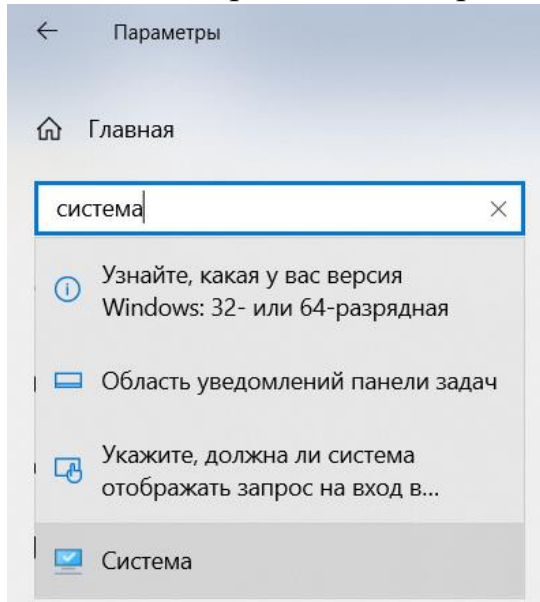
Для проведения нагрузочного тестирования веб-сервера будем использовать инструмент Apache Benchmark (ab), который является частью пакета Apache HTTP server.

Поэтому скачиваем с этого сайта Apache Benchmark:

<https://www.apachelounge.com/download/>

После чего извлекаем папку Apache64 по нужному нам пути (по сути по любому).

После чего переходим в «параметры», в них в поиске пишем «система»:



Жмякаем ее, далее в правой части приложения находим дополнительные настройки и переходим в них:

Сопутствующие параметры

[Параметры BitLocker](#)

[Диспетчер устройств](#)

[Удаленный рабочий стол](#)

[Защита системы](#)


Дополнительные параметры
системы

[Переименовать этот ПК \(для
опытных пользователей\)](#)

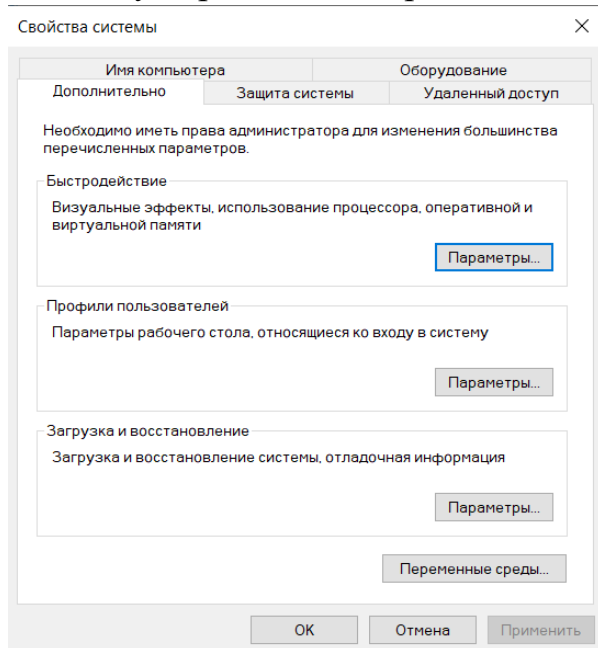
Справка в Интернете

[Проверка поддержки нескольких
языков](#)

 [Получить помощь](#)

 [Отправить отзыв](#)

По итогу перед нами открывается вот такое окно:



В котором переходим в «переменные среды...»:

Переменные среды



Переменные среды пользователя для hasee

Переменная	Значение
KMP_BLOCKTIME	0
OMP_WAIT_POLICY	PASSIVE
OneDrive	C:\Users\hasee\OneDrive
Path	C:\Users\hasee\AppData\Local\Microsoft\WindowsApps;C:\U...
QSYS_ROOTDIR	D:\intelFPGA_lite\21.1\quartus\sopc_builder\bin
TEMP	C:\Users\hasee\AppData\Local\Temp
TMP	C:\Users\hasee\AppData\Local\Temp

Создать...

Изменить...

Удалить

Системные переменные

Переменная	Значение
ComSpec	C:\Windows\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS	12
OS	Windows_NT
Path	C:\Program Files\Microsoft\jdk-11.0.16-hotspot\bin;C:\Wi...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTU...	AMD64
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 146 Stepping 4

Создать...

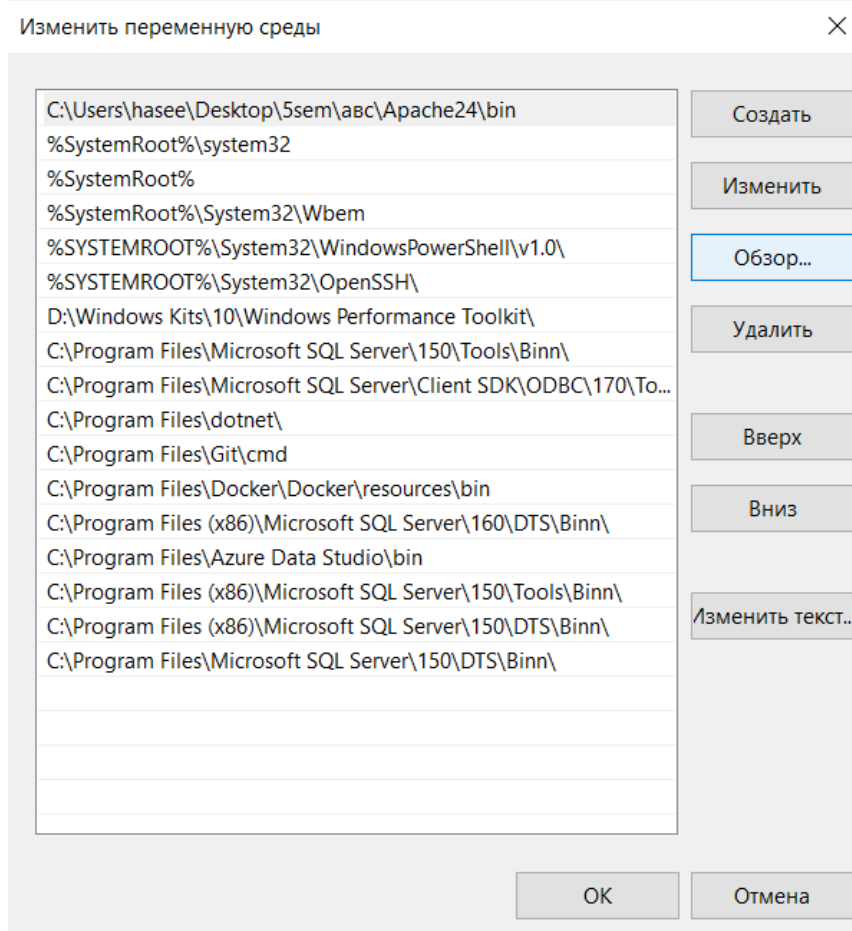
Изменить...

Удалить

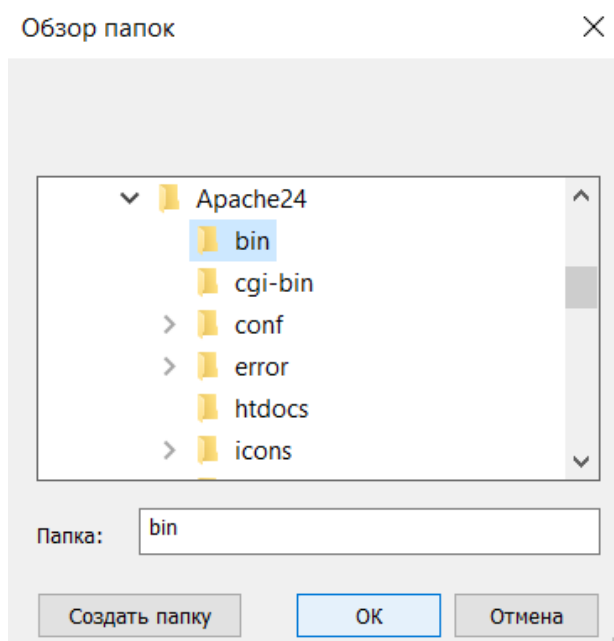
OK

Отмена

Далее нажимаем два раза на подчеркнутую красным строку Path в нижнем окне, после чего перед нами открывается еще одно окно:



Здесь мы нажимаем «обзор» и получаем:



В этом окне указываем папку bin из извлеченной нами папки Apache24 (можно и полностью извлечь папку, а не только Apache24), затем везде ждем «ok».

После чего запускаем командную строку вводим следующую команду и радуемся:

1	ab -V
---	-------

```
C:\Users\hasee>ab -V
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/

C:\Users\hasee>_
```

Теперь, имея установленный Apache Benchmark и рабочий веб-сервер, который доступен по порту 8080, в командной строке выполняем команду для нагрузочного тестирования:

1	ab -n 1000 -c 100 http://localhost:8080/
---	--

Сначала разберем команду:

- 1) **-n 1000** указывает, что будет отправлено 1000 запросов к серверу.
- 2) **-c 100** задает количество одновременных запросов (конкурентных пользователей).
- 3) **http://localhost:8080/** - URL веб-сервера.

Ну и само собой получаем следующий результат:

```
C:\Users\hasee>ab -n 1000 -c 100 http://localhost:8080/
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:      nginx/1.25.3
Server Hostname:      localhost
Server Port:          8080

Document Path:        /
Document Length:      615 bytes

Concurrency Level:    100
Time taken for tests:  0.369 seconds
Complete requests:    1000
Failed requests:      0
Total transferred:    848000 bytes
HTML transferred:     615000 bytes
Requests per second:  2711.95 [#/sec] (mean)
Time per request:     36.874 [ms] (mean)
Time per request:     0.369 [ms] (mean, across all concurrent requests)
Transfer rate:        2245.84 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median  max
Connect:      0    0   0.4      0     1
Processing:   7   35  12.5     32    79
Waiting:      3   23  14.3     20    70
Total:        7   35  12.4     33    79

Percentage of the requests served within a certain time (ms)
 50%    33
 66%    34
 75%    34
 80%    35
 90%    53
 95%    70
 98%    74
 99%    77
100%    79 (longest request)

C:\Users\hasee>_

```

Как можно заметить, был проведен сравнительный анализ локального хоста и выведена следующая информация:

Серверное программное обеспечение - nginx/1.25.3.

Имя хоста сервера – localhost

Порт сервера – 8080

Путь к документу - /

Длина документа – 615 байт

Уровень параллелизма – 100

Время, затраченное на тесты – 0,369

Полных запросов -1000

Неудачных запросов – 0

Всего передано – 848000 байт

Передано Html – 615000 байт

Запросов в секунду – 2711, 95

Время на запрос (среднее) – 36, 874 мс

Время на запрос (в среднем по всем одновременным запросам) – 0,369 мс

Скорость передачи – 2245, 84 кбайт/сек

Время подключения (мс)	мин	среднее	[+/-сд]	медиана	макс
Подключение	0	0	0,4	0	1
Обработка	7	35	12,5	32	79
Ожидание	3	23	14,3	20	70
Всего	7	35	12,4	33	79

Процент запросов, обработанных в пределах определенного времени (мс)

50% 33

66% 34

75% 34

80% 35

90% 53

95% 70

98% 74

99% 77

100% 79 (самый длительный запрос)

[+/-сд] - среднеквадратическое отклонение (стандартного отклонения) от среднего значения.

Ну и запустим тест на более больших числах:

```
C:\Users\hasee>
C:\Users\hasee>ab -n 10000 -c 1000 http://localhost:8080/
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 1000 requests
Completed 2000 requests
Completed 3000 requests
Completed 4000 requests
Completed 5000 requests
Completed 6000 requests
Completed 7000 requests
Completed 8000 requests
Completed 9000 requests
Completed 10000 requests
Finished 10000 requests


Server Software:      nginx/1.25.3
Server Hostname:      localhost
Server Port:          8080

Document Path:        /
Document Length:      615 bytes

Concurrency Level:    1000
Time taken for tests:  3.189 seconds
Complete requests:    10000
Failed requests:       0
Total transferred:    8480000 bytes
HTML transferred:     6150000 bytes
Requests per second:  3135.45 [#/sec] (mean)
Time per request:     318.933 [ms] (mean)
Time per request:     0.319 [ms] (mean, across all concurrent requests)
```

```
HTML transferred:      6150000 bytes
Requests per second:   3135.45 [#/sec] (mean)
Time per request:      318.933 [ms] (mean)
Time per request:      0.319 [ms] (mean, across all concurrent requests)
Transfer rate:         2596.54 [Kbytes/sec] received
```

Connection Times (ms)

	min	mean[+/-sd]	median	max
Connect:	0	0 0.4	0	1
Processing:	73	306 53.5	317	425
Waiting:	2	232 57.5	235	366
Total:	73	306 53.5	317	425

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

50%	317
66%	320
75%	323
80%	328
90%	336
95%	342
98%	405
99%	414
100%	425 (longest request)

C:\Users\hasee>

Найдем примерное количество ограничений одновременных запросов:

The screenshot shows the Docker Desktop interface. On the left is a sidebar with navigation options: Containers, Images, Volumes, Dev Environments (BETA), Docker Scout, and Learning center. The main panel displays the 'hopeful_buck' container, which is running the 'nginx:latest' image. The 'Exec' tab is selected, showing a terminal window. The terminal output shows the configuration of the nginx.conf file, specifically the 'events' block where 'worker_connections' is set to 1024. The container status is 'Running (2 minutes ago)'.

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

```
2023-11-11 11:11:11 2023/11/11 04:11:11 [notice] 1#1: nginx/1.25.3
2023-11-11 11:11:11 2023/11/11 04:11:11 [notice] 1#1: built by gcc 12.2.0 (Debian
2023-11-11 11:11:11 2023/11/11 04:11:11 [notice] 1#1: OS: Linux 5.15.90.1-microsof
2023-11-11 11:11:11 2023/11/11 04:11:11 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 16
2023-11-11 11:11:11 2023/11/11 04:11:11 [notice] 1#1: start worker processes
2023-11-11 11:11:11 2023/11/11 04:11:11 [notice] 1#1: start worker process 29
2023-11-11 11:11:11 2023/11/11 04:11:11 [notice] 1#1: start worker process 30
2023-11-11 11:11:11 2023/11/11 04:11:11 [notice] 1#1: start worker process 31
2023-11-11 11:11:11 2023/11/11 04:11:11 [notice] 1#1: start worker process 32
2023-11-11 11:11:11 2023/11/11 04:11:11 [notice] 1#1: start worker process 33
2023-11-11 11:11:11 2023/11/11 04:11:11 [notice] 1#1: start worker process 34
2023-11-11 11:11:11 2023/11/11 04:11:11 [notice] 1#1: start worker process 35
2023-11-11 11:11:11 2023/11/11 04:11:11 [notice] 1#1: start worker process 36
2023-11-11 11:11:11 2023/11/11 04:11:11 [notice] 1#1: start worker process 37
2023-11-11 11:11:11 2023/11/11 04:11:11 [notice] 1#1: start worker process 38
2023-11-11 11:11:11 2023/11/11 04:11:11 [notice] 1#1: start worker process 39
2023-11-11 11:11:11 2023/11/11 04:11:11 [notice] 1#1: start worker process 40
```

Всего 12 процессов на 1024 количества ограничений получаем примерно 12288 одновременных запросов.