|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Информационных технологий

Кафедра Инструментального и прикладного программного обеспечения

**ОТЧЕТ ПО ПРАКТИЧЕСКИМ РАБОТАМ №1-10**

**по дисциплине**

«Настройка и администрирование сервисного программного обеспечения»

|  |  |
| --- | --- |
| Выполнил студент группы ИКБО-20-22 | Шумахер М.Е. |
| Принял ассистент | Зарипов Е.А. |

|  |  |  |
| --- | --- | --- |
| Практиечские работы выполнены | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись руководителя)* |

Москва 2024 г.

**ОГЛАВЛЕНИЕ**

[ПРАКТИЧЕСКАЯ РАБОТА №1 3](#_Toc164993666)

[ПРАКТИЧЕСКАЯ РАБОТА №2 10](#_Toc164993667)

[ПРАКТИЧЕСКАЯ РАБОТА №3 26](#_Toc164993668)

[ПРАКТИЧЕСКАЯ РАБОТА №4 36](#_Toc164993669)

[ПРАКТИЧЕСКАЯ РАБОТА №5 45](#_Toc164993670)

[ПРАКТИЧЕСКАЯ РАБОТА №6 50](#_Toc164993671)

[ПРАКТИЧЕСКАЯ РАБОТА №7 55](#_Toc164993672)

[ПРАКТИЧЕСКАЯ РАБОТА №8 59](#_Toc164993673)

[ПРАКТИЧЕСКАЯ РАБОТА №9 66](#_Toc164993674)

[ПРАКТИЧЕСКАЯ РАБОТА №10 69](#_Toc164993675)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 74](#_Toc164993676)

ПРАКТИЧЕСКАЯ РАБОТА №1

**Цель работы**: получить навыки по развертыванию приложения Docker.

**Задание на практическую работу:**

Установить Docker. Запустить Docker Dashboard. Разобраться с функционалом Docker Dashboard, ознакомившись с https://docs.docker.com/desktop/dashboard/ Для установки и запуска контейнеров Docker в ОС Windows запустить cmd (command line) и выполнить команду docker run имя\_контейнера. Например, docker run redis. Таким образом, в данном случае будет установлена СУБД redis. Далее перейти в Docker Dashboard, найти установленный и запущенный контейнер (в нашем случае redis) выбрать его и найти кнопку CLI (command line interface – командная строка для данного контейнера. Пиктограмма выглядит следующим образом: >\_ )

Ознакомиться с операциями, производимыми в консоли Docker и выполнить следующие действия:

1. Обратиться по адресу https://hub.docker.com/ и ознакомиться с приложениями доступными для работы с Docker.
2. Установить контейнер СУБД redis. Используя CLI контейнера, запустить клиент redis-cli.
3. Ознакомиться с документацией по командам redis. Документация на СУБД <https://redis.io/commands>.
4. В redis выполнить команды SET https://redis.io/commands/set и GET https://redis.io/commands/get Создать 5 ключей со значениями с помощью SET и прочитать ключи со значениями с помощью GET.
5. Получение значения по ключу и его замена на новое.
6. Добавление строки к уже существующему значению.
7. Добавление числа и изменение его значения.
8. Создание ключа со значением типа хеш-таблица
9. Работа со множествами. Задействовать команды SADD, SDIFF, SMOVE, SPOP, SUNION, SREM.
10. Работа с упорядоченными наборами. Задействовать команды ZADD, ZCOUNT, ZDIFF, ZPOPMAX, ZPOPMIN, ZUNION, ZMSCORE, ZLEXCOUNT.
11. Из документации выбрать любые не использовавшиеся ранее 5 команд и задействовать их в работе.

**Выполнение заданий**

Обратимся по адресу https://hub.docker.com и ознакомимся с приложениями доступными для работы с Docker, Рисунок 1.1.

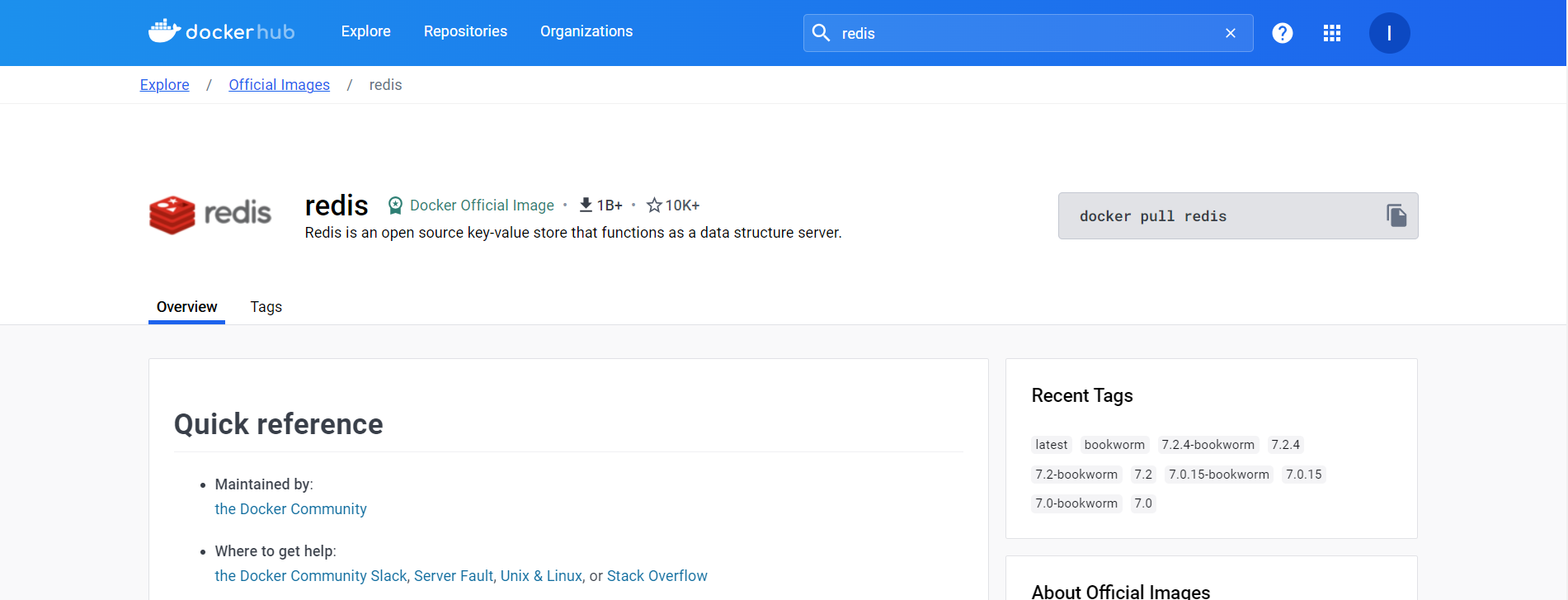


Рисунок 1.1 – Ознакомление с образом redis на Docker Hub

Установим контейнер СУБД redis, скачав при установке образ redis. Используя CLI контейнера, запустим клиент redis-cli, Рисунок 1.2.

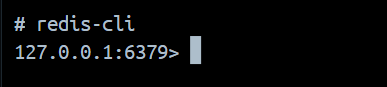


Рисунок 1.2 – Запуск redis-cli

Ознакомимся с документацией по командам redis из документации на https://redis.io/commands, Рисунок 1.3.



Рисунок 1.3 – Внесение изменений

В redis-cli выполним команды SET и GET. Создаем 5 ключей со значениями с помощью SET и прочитываем ключи со значениями с помощью GET., Рисунок 1.4.

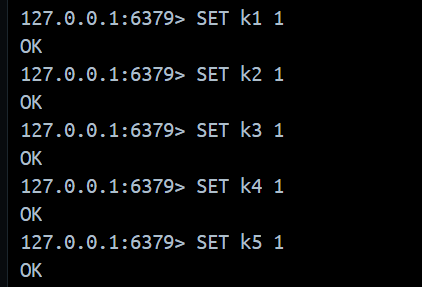


Рисунок 1.4 – Создание 5 ключей со значениями и их чтение   
с помощью команд SET и GET

Получим значения по ключу и заменим его на новое значение через команду GETSET, Рисунок 1.5.



Рисунок 1.5 – Получение значения и замена его на новое

Добавим строку к уже существующему значению через команду APPEND, Рисунок 1.6.

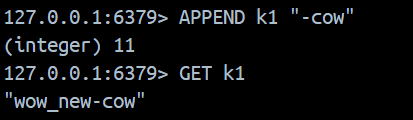


Рисунок 1.6 – Добавление строки к существующему ключу

Добавим число и изменим его значение на 10, Рисунок 1.7.

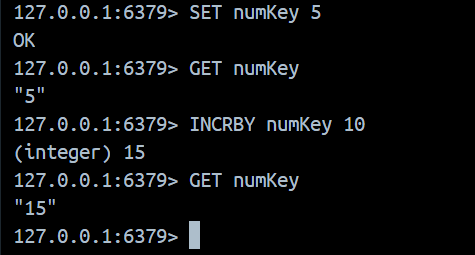


Рисунок 1.7 – Добавление числа и изменение его значения на 5

Создадим ключ со значением типа хеш-таблица, Рисунок 1.8.



Рисунок 1.8 – Создание ключа со значением типа хеш-таблица

Команды: SADD – добавление значения в множество, SDIFF – вывод разницы между множествами, SMOVE – перемещение элемента из одного множества в другой, SPOP – удаление и возвращение случайного элемента из множества, SUNION – возвращение объединённого множества, SREM – удаление элемента из множества. Задействуем данные команды для работы с множествами, Рисунок 1.9.

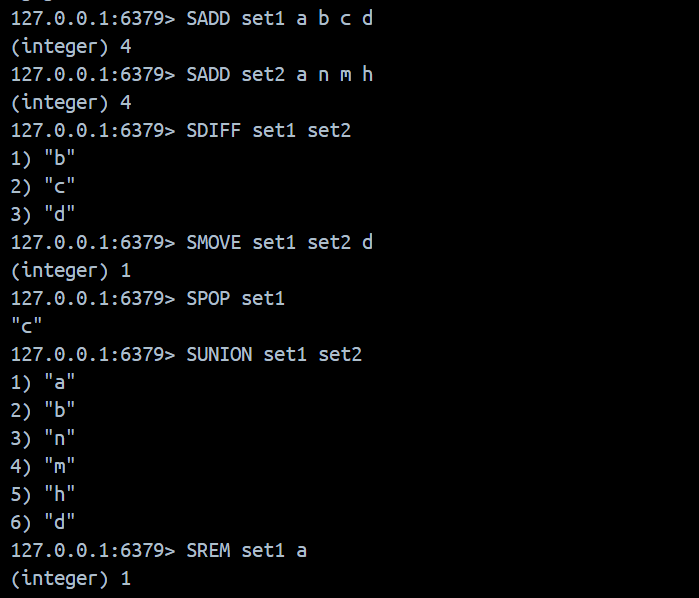
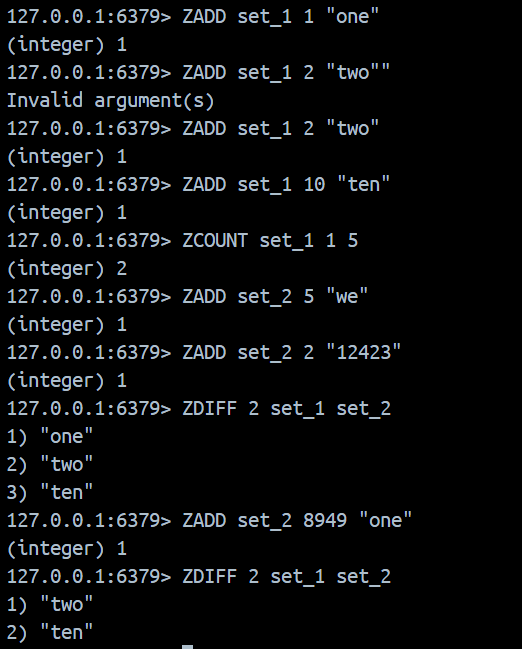


Рисунок 1.9 – Использование команд для работы с множествами

Команды: ZADD – добавление значения в набор, ZCOUNT – вывод кол-ва элементов на заданном диапазоне, ZDIFF – вывод разницы между двумя наборами, ZPOPMAX – удаление и возвращение элемента из набора с наивысшим приоритетом, ZPOPMIN – удаление и возвращение элемента из набора с низшим приоритетом, ZUNION – возвращает объединённый набор, ZMSCORE – возвращает приоритеты элементов в наборе, ZLEXCOUNT – возвращает количество элементов набора на заданном промежутке. Задействуем данные команды для работы с упорядоченными наборами, Рисунок 1.10.



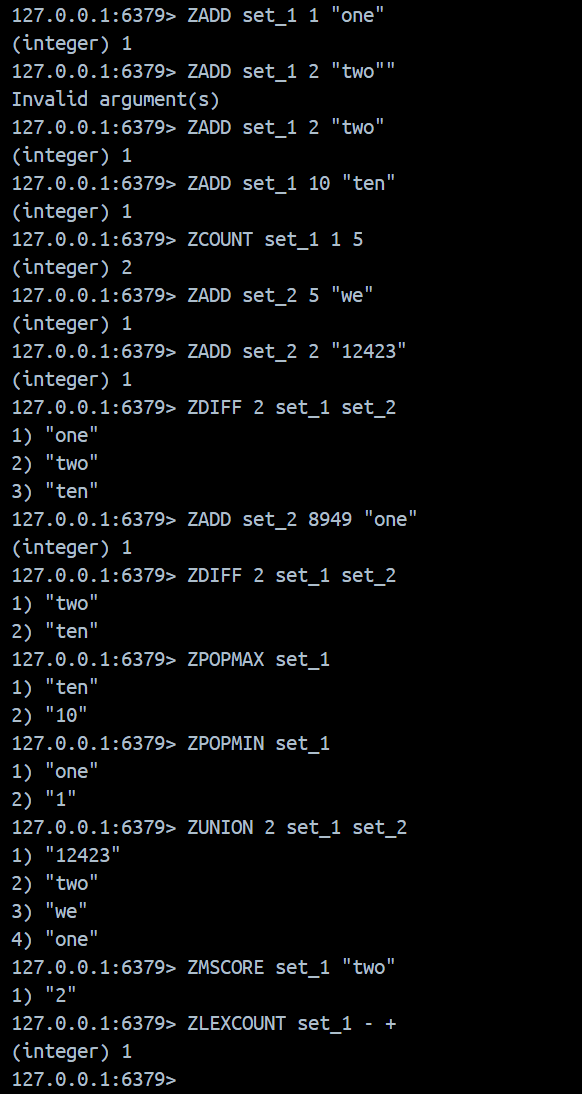


Рисунок 1.10 – Использование команд для работы с упорядоченными наборами

Выберем любые не использовавшиеся ранее 5 команд и задействуем их в работе, Рисунок 1.11.

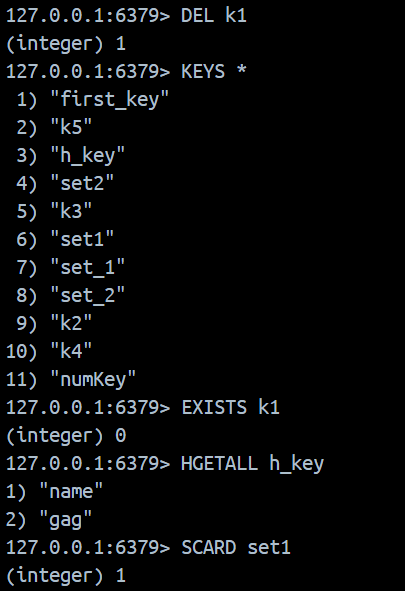


Рисунок 1.11 – Использование других 5-ти команд

**Вывод**

Результаты выполнения данной работы позволили успешно освоить навыки развертывания приложений с использованием Docker. После установки Docker и запуска Docker Dashboard был осуществлен анализ функционала панели управления, ознакомившись с документацией. Для практического применения была использована команда "docker run" в командной строке операционной системы Windows для запуска контейнера с базой данных redis. Далее, через Docker Dashboard, был произведен поиск и выбор запущенного контейнера redis, а также работа с командной строкой интерфейса для данного контейнера. Этот опыт позволил уверенно освоить процесс развертывания приложений в среде Docker и понять основные принципы управления контейнерами.

ПРАКТИЧЕСКАЯ РАБОТА №2

**Цель работы**: получить навыки по развертыванию операционных систем Unix на основе Windows Subsystem для Linux в Windows 10.

**Задание на практическую работу:**

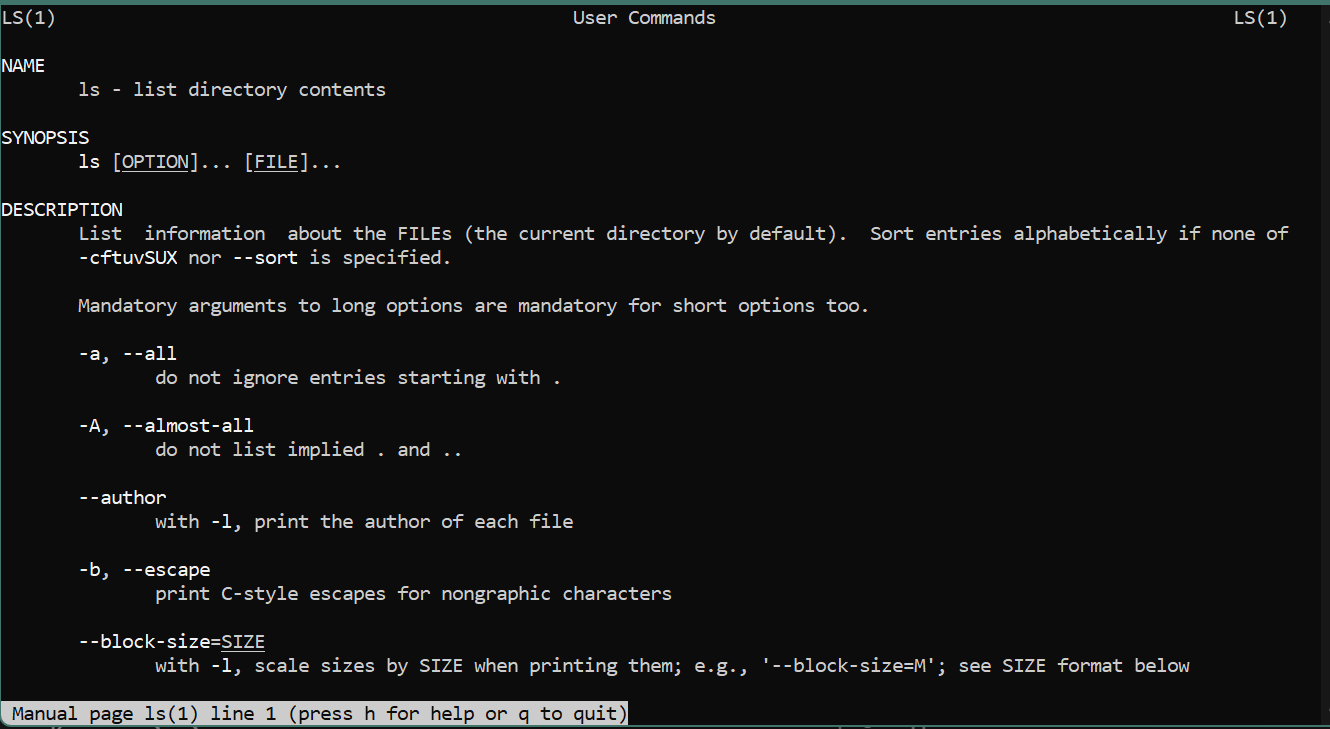
Установить Ubuntu.

Ознакомиться с операциями, производимыми в консоли установленной Linux системы и выполнить следующие команды:

1. Ознакомиться и выполнить команду man и опцию --help для любой команды
2. Ознакомиться и выполнить команду whatis
3. Ознакомиться и выполнить команду whereis Работа с текстом
4. Ознакомиться и выполнить команды more и less
5. Ознакомиться и выполнить команды head и tail
6. Ознакомиться и выполнить команду grep
7. Ознакомиться и выполнить команды sort
8. Ознакомиться и выполнить команды wc
9. Ознакомиться и выполнить команды diff
10. Отредактировать текстовый файл с помощью редакторов nano и vi Работа с процессами
11. Ознакомиться и выполнить команду kill
12. Ознакомиться и выполнить команду ps
13. Ознакомиться и выполнить команды top и htop. Посмотреть список запущенных процессов в операционной системе
14. Ознакомиться и выполнить команду time
15. Установить приложение используя команду sudo
16. Запустить приложение используя команду sudo
17. Ознакомиться и выполнить команду apt-get (с применением команды sudo)
18. Ознакомиться и выполнить команду date
19. Ознакомиться и выполнить команду uname
20. Ознакомиться и выполнить команду uptime
21. Ознакомиться и выполнить команду sleep
22. Ознакомиться и выполнить команды useradd, userdel и usermod
23. Ознакомиться и выполнить команду passwd
24. Ознакомиться и выполнить команду ip
25. Ознакомиться и выполнить команду ping
26. Ознакомиться и выполнить команду nethogs
27. Ознакомиться и выполнить команду traceroute
28. Ознакомиться и выполнить команду hostname
29. Ознакомиться и выполнить команду pwd
30. Ознакомиться и выполнить команду ls. Добавить параметры для этой команды -la. Выполнить эту же команду с параметрами -l и -a отдельно. Объяснить полученный результат
31. Ознакомиться и выполнить команду cd
32. Ознакомиться и выполнить команды mkdir и rmdir
33. Ознакомиться и выполнить команду rm
34. Ознакомиться и выполнить команду touch
35. Ознакомиться и выполнить команду cp
36. Ознакомиться и выполнить команду mv
37. Ознакомиться и выполнить команду locate
38. Ознакомиться и выполнить команду cat
39. Ознакомиться и выполнить команду df
40. Ознакомиться и выполнить команду du
41. Ознакомиться и выполнить команду tar
42. Ознакомиться и выполнить команды zip и unzip
43. Ознакомиться и выполнить команду chmod
44. Ознакомиться и выполнить команду chown
45. Ознакомиться и выполнить команду file
46. Ознакомиться и выполнить команду find
47. Ознакомиться и выполнить команду clear
48. Ознакомиться и выполнить команду halt
49. Ознакомиться и выполнить команду reboot
50. Ознакомиться и выполнить команды mount и umount

**Выполнение заданий**

Работа с документацией. Выполним команду man (предоставляет доступ к руководствам по использованию команд и программ) и опцию –help (отображает подсказку по командам) для любой команды, Рисунок 2.1.



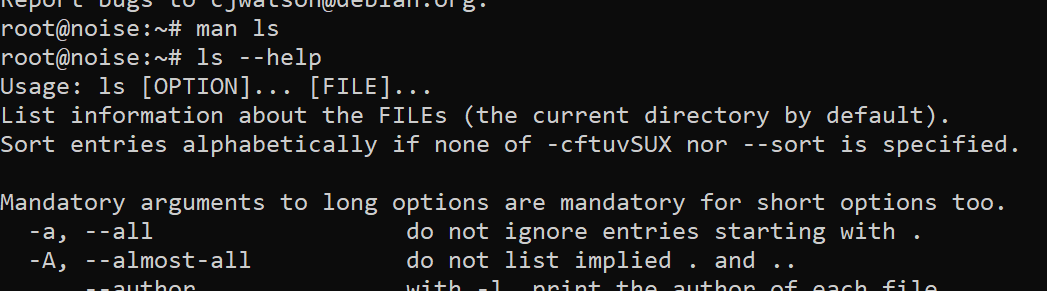


Рисунок 2.2 ­– Выполнение команды man и опции –help

Выполним команду whatis (выводит краткое описание команд), Рисунок 2.3.



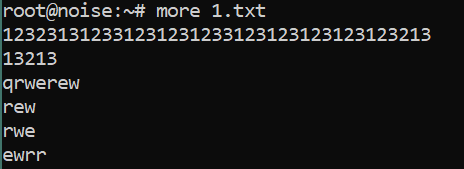
Рисунок 2.3 ­– Выполнение команды whatis

Выполним команду whereis (позволяет найти расположение двоичных файлов, файлов исходного кода и файлов справочной страницы для команды), Рисунок 2.4.



Рисунок 2.4 – Выполнение команды whereis

Работа с текстом. Выполним команды more (позволяет просматривать содержимое текстовых файлов постранично) и less (усовершенствованная версия команды more), Рисунок 2.5.



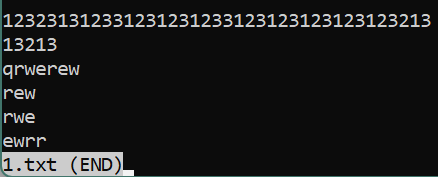


Рисунок 2.5­– Выполнение команд more, less

Выполним команды head (вывод первых строк из файлов) и tail (вывод последних строк из файлов), Рисунок 2.6.

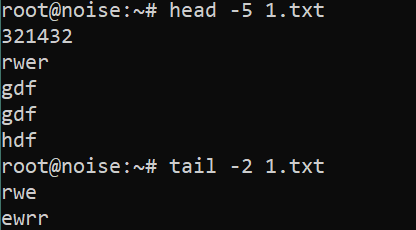


Рисунок 2.6 ­– Выполнение команды

Выполним команду grep (находит на вводе строки, отвечающие заданному регулярному выражению, и выводит их), Рисунок 2.7.

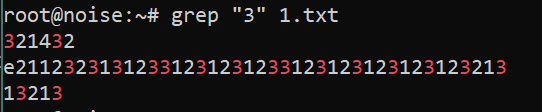


Рисунок 2.7 ­– Выполнение команды grep

Выполним команду sort (сортирует строки в файле и выводит их на консоль), Рисунок 2.8.

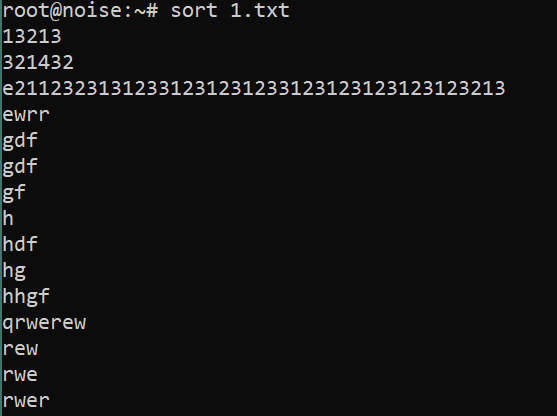


Рисунок 2.8 ­– Выполнение команды sort

Выполним команду wc (подсчитывает число строк, слов и байт в файлах, указанных в параметре), Рисунок 2.9.



Рисунок 2.9 ­– Выполнение команды wc

Выполним команду diff (выводит разницу в файлах), Рисунок 2.10.

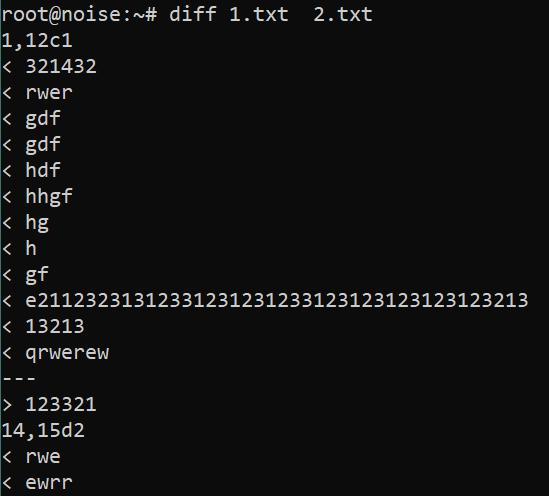
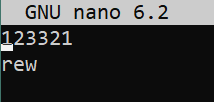


Рисунок 2.10 ­– Выполнение команды diff

Отредактируем текстовый файл с помощью редакторов nano и vi, Рисунок 2.11.



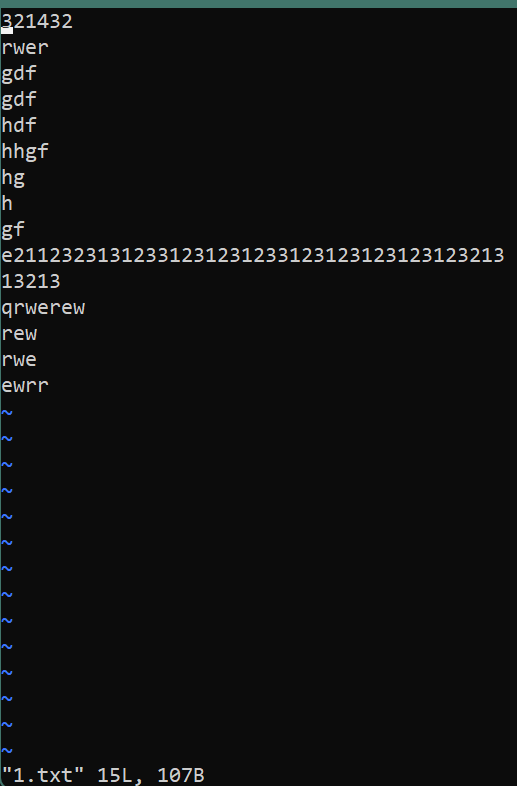


Рисунок 2.11 ­– Выполнение команд nano, vi

Работа с процессами. Выполним команду kill (завершение работы процесса), Рисунок 2.12.



Рисунок 2.12 ­– Выполнение команды kill

Выполним команду ps (вывод всех процессов системы), Рисунок 2.13.

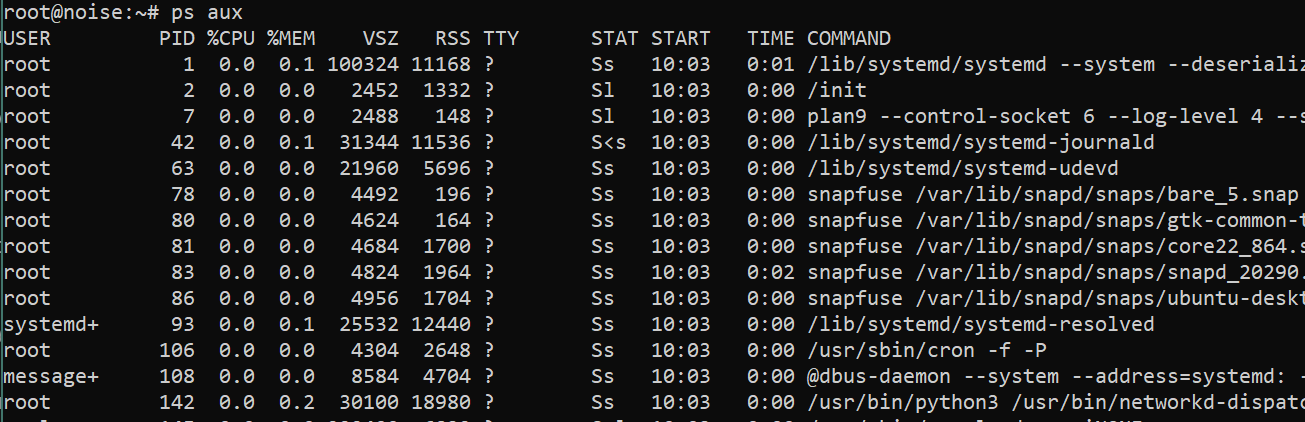


Рисунок 2.13­– Выполнение команды ps

Выполним команды top (позволяет пользователям отслеживать процессы и использование системных ресурсов) и htop и просмотрим список запущенных процессов в операционной системе, Рисунок 2.14.

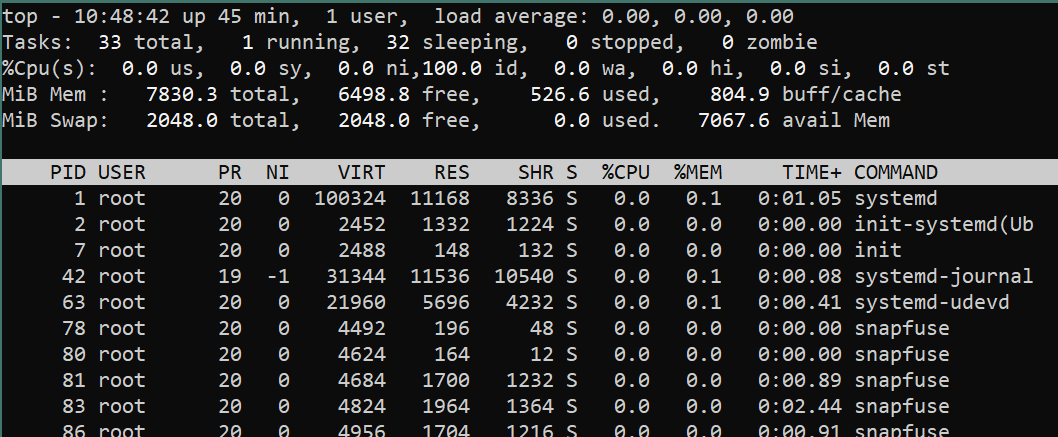
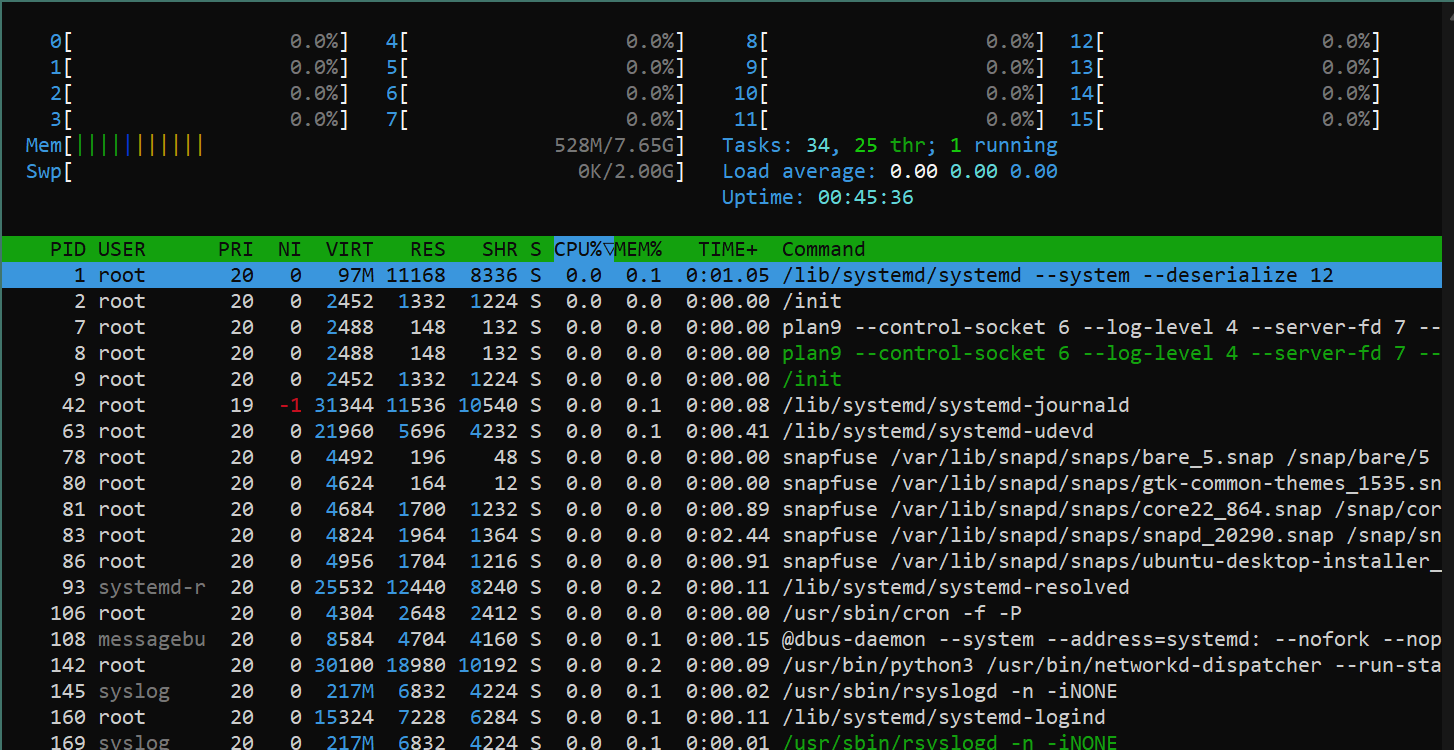
 

Рисунок 2.14­– Выполнение команды top, htop

Выполним команду time (запускает заданную пользователем команду и после этого выводит информацию о времени ее выполнения), Рисунок 2.15.

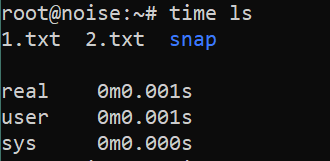


Рисунок 2.15 ­– Выполнение команды time

Команды пользователя. Установим приложение используя команду sudo, Рисунок 2.16.

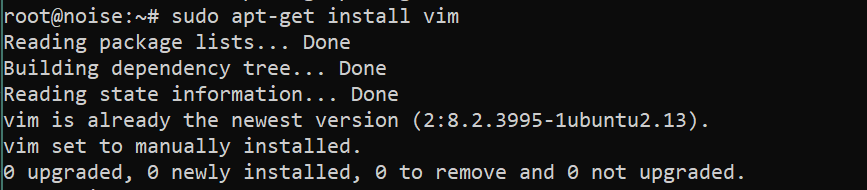


Рисунок 2.16 ­– Выполнение команды sudo apt-get

Запустим приложение используя команду sudo, Риуснок 2.17.

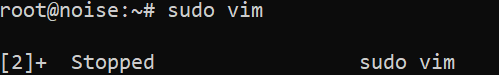


Рисунок 2.17 ­– Выполнение команды sudo vim

Выполним команду apt-get, Рисунок 2.18.



Рисунок 2.18 ­– Выполнение команды apt-get

Выполним команду date, Рисунок 2.19.



Рисунок 2.19 ­– Выполнение команды date

Выполним команду uname, Рисунок 2.20.



Рисунок 2.20 ­– Выполнение команды uname

Выполним команду uptime, Рисунок 2.21.



Рисунок 2.21 ­– Выполнение команды uptime

Выполним команду sleep, Рисунок 2.22.



Рисунок 2.22 ­– Выполнение команды sleep

Управление пользователями. Выполним команды useradd, userdel и usermod, passwd, Рисунок 2.23.

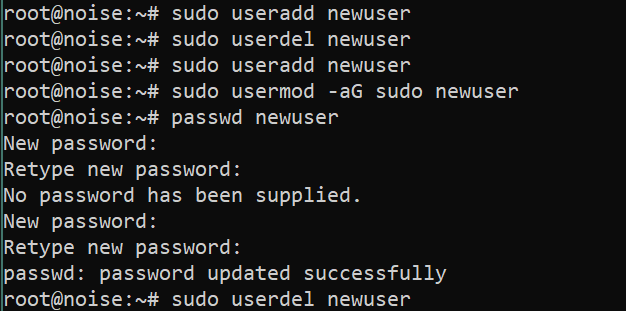


Рисунок 2.23 ­– Выполнение команд управления пользователями

Выполним команду ip, Рисунок 2.24.

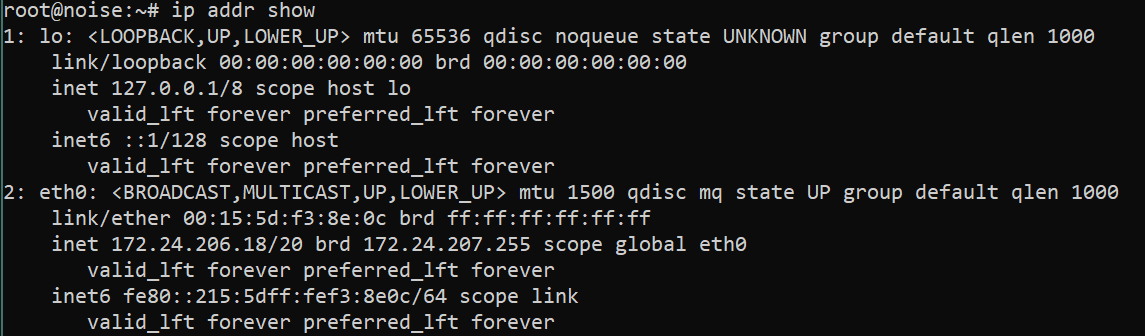


Рисунок 2.24 ­– Выполнение команды ip

Выполним команду ping, Рисунок 2.25.

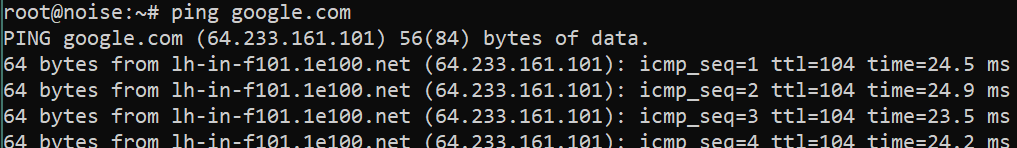


Рисунок 2.25 ­– Выполнение команды ping

Выполним команду nethogs (просмотр состояния сетей), Рисунок 2.26.

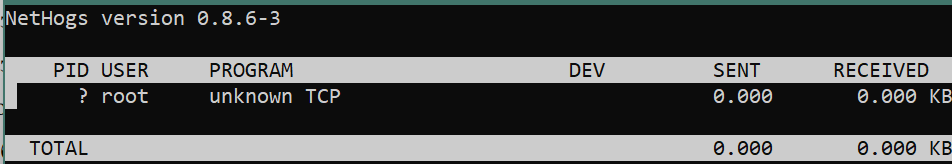


Рисунок 2.26 ­– Выполнение команды nethogs

Выполним команду traceroute (отправка пакетов на url), Рисунок 2.27.

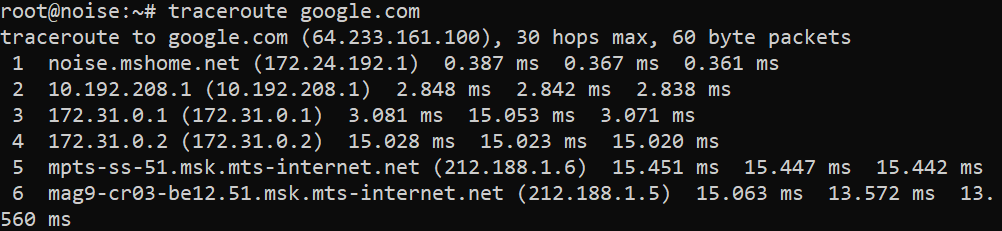


Рисунок 2.27 ­– Выполнение команды traceroute

Выполним команду hostname, Рисунок 2.28.



Рисунок 2.28 ­– Выполнение команды hostname

Выполним команду pwd, Рисунок 2.29.



Рисунок 2.29 ­– Выполнение команды pwd

Выполним команду ls. Добавить параметры для этой команды -la. Выполнить эту же команду с параметрами -l и -a отдельно, Рисунок 2.30. Объяснить полученный результат:

-la выводит длинный лист файлов (также начинающихся на .)

-l выводит длинный лист файлов (все, кроме начинающихся на .)

-a выводит названия файлов (также начинающихся на ..

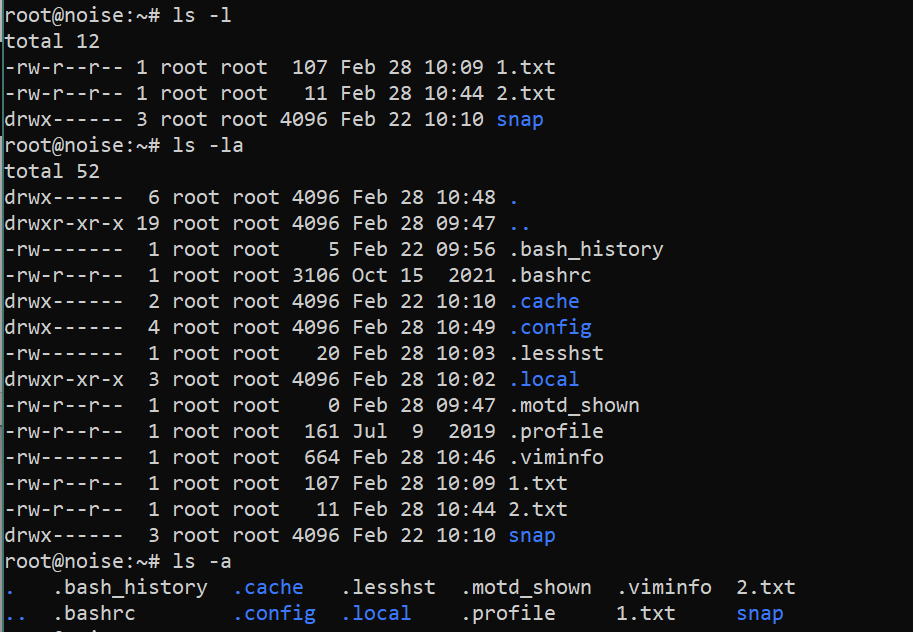


Рисунок 2.30 ­– Выполнение команд ls

Выполним команду cd, Рисунок 2.31.

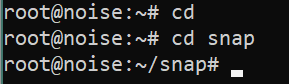


Рисунок 2.31 ­– Выполнение команды cd

Выполним команды mkdir и rmdir, rm, cp, mv, touch, Рисунок 2.32.

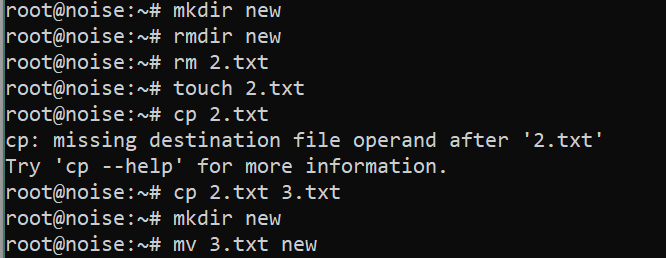


Рисунок 2.32 ­– Выполнение команд mkdir и rmdir, rm, cp, mv, touch

Выполним команду locate, Рисунок 2.33.

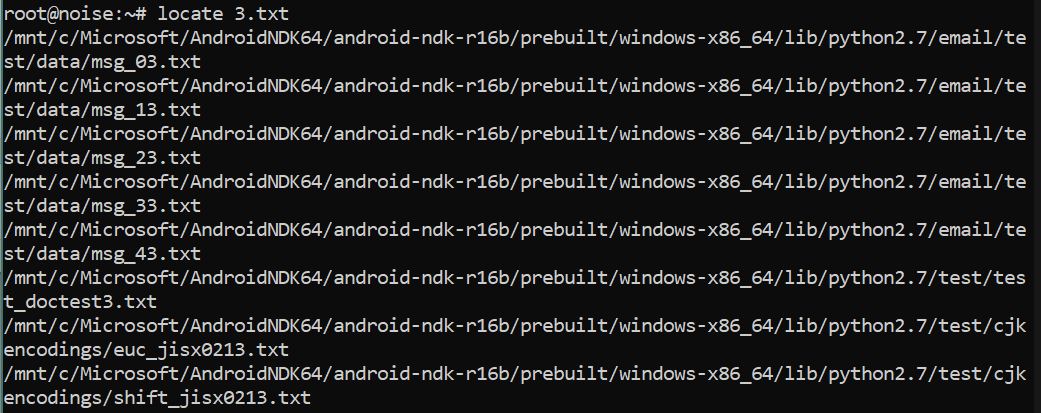


Рисунок 2.33 ­– Выполнение команды locate

Выполним команду cat, df, Рисунок 2.34.

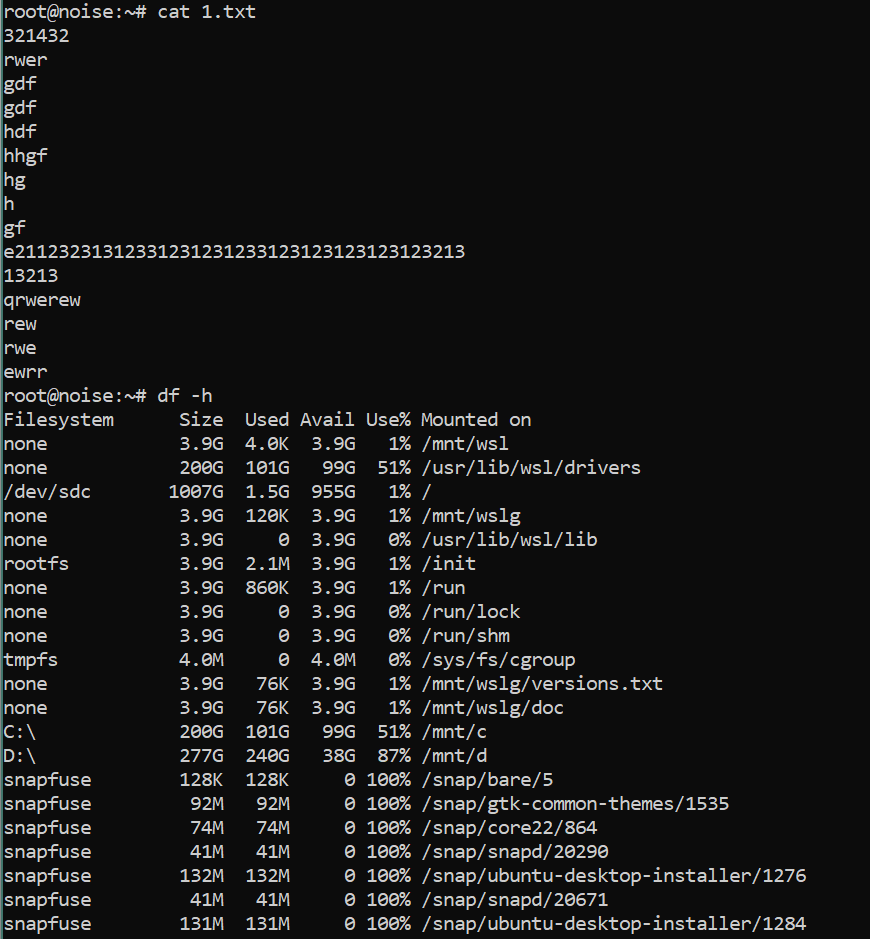


Рисунок 2.34 ­– Выполнение команд cat, df

Выполним команду du, Рисунок 2.35.

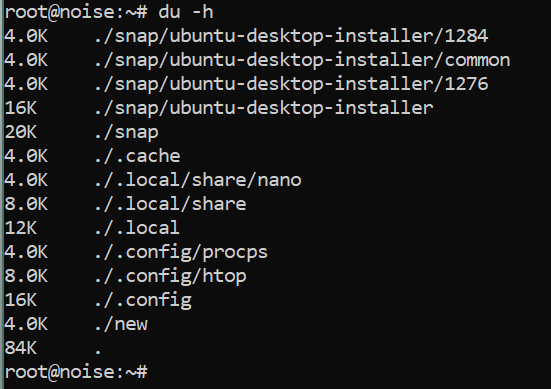


Рисунок 2.35 ­– Выполнение команды du

Выполним команду tar, Рисунок 2.36.



Рисунок 2.36 ­– Выполнение команды tar

Выполним команды zip и unzip, chmod, chown, Рисунок 2.37.

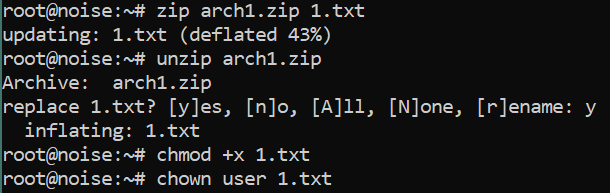
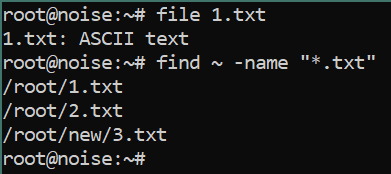


Рисунок 2.37 ­– Выполнение команд zip и unzip, chmod, chown

Выполним команды file, find, Рисунок 2.38.



Рисуно 2.38 – Выполнение команд file, find

Выполним команду clear, выполним команду halt (принудительное завершение работы процессора), Рисунок 2.39.

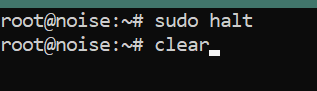


Рисунок 2.39 ­– Выполнение команды halt, clear

Выполним команду reboot, Рисунок 2.40.



Рисунок 2.40 ­– Выполнение команды reboot

Выполним команды mount и umount, Рисунок 2.41.



Рисунок 2.41 ­– Выполнение команд mount и umount

**Вывод**

Выполнение данной работы позволило освоить навыки развертывания операционных систем Unix на базе Windows Subsystem для Linux в Windows 10. В ходе практики были изучены и применены различные команды консоли, включая работу с текстом, процессами, управление пользователями, а также базовые операции с файлами и системой.

ПРАКТИЧЕСКАЯ РАБОТА №3

**Цель работы:** получить навыки по запуску Linux-приложений в Windows 10, а также запуску Windows-приложений из ОС Linux.

**Задание на практическую работу:**

В Windows 10 появились три классические консольные UNIX утилиты curl.exe, tar.exe, ssh.exe.

curl --version

ssh -V

tar --version

ver – вывести версию Windows

Ознакомиться и выполнить команды в командной строке Windows 10. Для выполнения работы понадобится командный интерпретатор cmd. После запуска cmd с помощью команды cd переместиться в удобный каталог для работы (там, где удобнее хранить временные файлы, например, c:\temp, например, cd c:\temp. Все выполняемые команды проверить на работоспособность и привести в отчете. В cmd работает поиск по комбинации CTRL-F.

1. Проверить версию команды curl. Например, curl –version
2. Открыть 10 URL различных ресурсов и ознакомиться с полученными данными. Например, curl mail.ru curl rambler.ru curl google.com
3. Осуществить загрузку файла с помощью команды curl с оригинальным именем, опция -O. Скачать одной командой 3 файла с опцией -О.
4. Осуществить загрузку с 3 ftp-серверов (например, производителей компьютерного оборудования HP, Asus, MSI, Lenovo, Dell) файлы (например, драйверы. Например, curl -O ftp://ftp.hp.com:21/pub/softpaq/sp40001/sp40001.cva Возможно, для ftp потребуется указывать логин, пароль Например, curl -u anonymous:anonymous -O ftp://yourftpserver/yourfile.tar.gz anonymous – стандартный логин и пароль для анонимного входа на ftpсервер
5. Осуществить загрузку файла с помощью команды curl и сохраните файл под другим именем, опция -o. Скачайте три файла с опцией -o для сохранения файлов с определенным именем
6. Сохранить какое-либо изображение используя опцию -OL . Сохраните 3 изображения используя опцию -OL (можно сохранить каждое изображение по отдельности)
7. Посмотреть для 5 URL какие файлы cookie загружаются на компьютер. Например, curl --cookie-jar cnncookies.txt https://www.cnn.com/index.html -O. Просмотреть сформированные cookie файлы можно с помощью команды type. Например, type cnncookies.txt
8. Выполнить команду curl с опцией -L. Сравнить вывод без этой опции. Объяснить полученный результат вывода. Например, curl -L google.com
9. Получить заголовки сайтов 10 URL командой curl, опция -I
10. Составить (а при наличии тестового сервера и отправить) POST-запросы используя команду curl, в виде списка имя=значение, и в виде JSON, для 3 URL, опция -d Например, curl -d "param1=test1&param2=test2" http://test.com (ключ=значение) curl -d '{"param1":"test1","param2":"test2"}' \http://www.test.com (JSON)
11. Использовать авторизацию для 3 URL, опция -u. Например, curl -u <https://testurl.com/entry>
12. Перенаправить запрос с localhost на URL (эквивалентно прописыванию в /etc/hosts), сделать для 3 URL. Для каждого URL свое перенаправление отдельной командой., опция –resolve. Например, curl --resolve www.test.com:80:localhost <http://www.test.com/>
13. Вывести время работы команды для конкретного URL. Повторить команду 3 раза на одном URL. Выполнить такую же операцию для 5 различных URL, опция -w Например, curl -w "%{time\_total}\n" -o /dev/null -s [www.mail.ru](http://www.mail.ru)
14. Для 3 URL осуществить имитацию определенного браузера, опция -A. Может понадобиться в случае, если удаленный сервер может быть настроен так, чтобы блокировать пользовательский агент (браузер) или возвращать различное содержимое в зависимости от устройства посетителя и браузера Например, curl -A "Mozilla/5.0 (X11; Linux x86\_64; rv:60.0) Gecko/20100101 Firefox/60.0" <https://getfedora.org/>
15. Отправить файлы cookie при скачивании дистрибутива. По умолчанию при запросе ресурса с помощью curl файлы cookie не отправляются и не сохраняются. Чтобы отправить файлы cookie на сервер, необходимо использовать опцию -b за которой следует имя файла, содержащего файлы cookie, или строку. Например, чтобы загрузить rpm-файл Oracle Java JDK jdk-10.0.2\_linux-x64\_bin.rpm необходимо передать файл cookie с именем oraclelicense со значением a curl -L -b "oraclelicense=a" -O <http://download.oracle.com/otnpub/java/jdk/10.0.2+13/19aef61b38124481863b1413dce1855f/jdk10.0.2_linux-x64_bin.rpm>
16. Выполнить команду по загрузке rpm-файла Oracle Java JDK jdk10.0.2\_linux-x64\_bin.rpm без отправки cookie. Сравнить скачанные файлы и объяснить полученный результат
17. Самостоятельно выбрать ранее не использовавшиеся 5 опций команды curl. Выполнить команды и объяснить полученный результат.

**Выполнение задания**

Проверим версию команды curl, Рисунок 3.1.

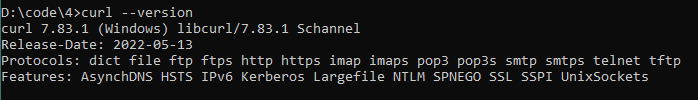


Рисунок 3.1 – Проверка версии curl

Откроем 10 URL различных ресурсов и ознакомимся с полученными данными, Рисунок 3.2.

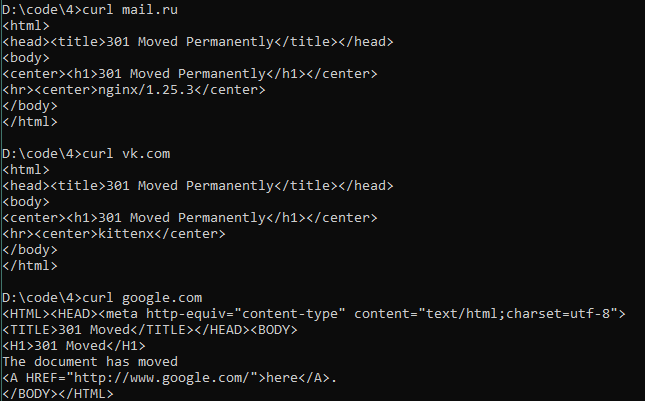


Рисунок 3.2 – Открытие 10 URL

Данное действие было повторено еще на 7 URL. Осуществим загрузку файла с помощью команды curl с оригинальным именем, опция -O. Скачаем одной командой 3 файла с опцией -О, Рисунок 3.3.

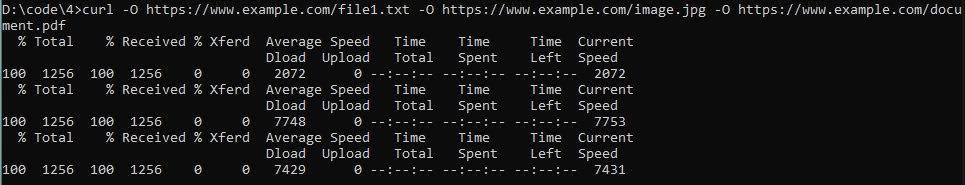


Рисунок 3.3 – Осуществление загрузки файла с помощью команды curl

Осуществим загрузку файлов с 3 разных ftp-серверов, Рисунок 3.4.

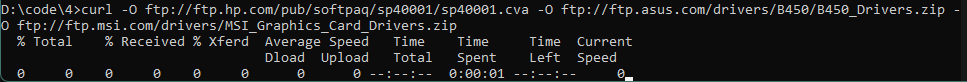


Рисунок 3.4 – Осуществление загрузки файлов с ftp-серверов

Осуществим загрузку файла с помощью команды curl и сохраним файл под другим именем, опция -o. Скачаем три файла с опцией -o для сохранения файлов с определенным именем, Рисунок 3.5.

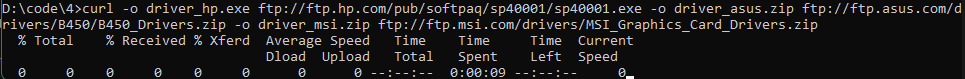


Рисунок 3.5 – Осуществление загрузки файлов с опцией -o

Сохраним какое-либо изображение используя опцию -OL. Сохраним 3 изображения используя опцию -OL, Рисунок 3.6.

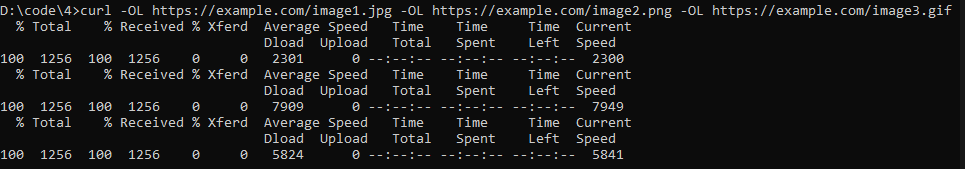


Рисунок 3.6 – Осуществление загрузки изображений с опцией -OL

Посмотрим для 5 URL какие файлы cookie загружаются на компьютер, Рисунок 3.7.

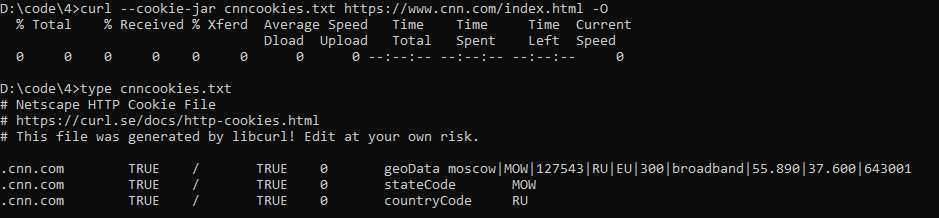


Рисунок 3.7 – Просмотр cookie файлов на компьютере

Данное действие было сделано еще на 4 URL. Выполним команду curl с опцией -L. Сравним вывод без этой опции, Рисунок 3.8-3.9. Объяснение: Опция -L позволяет следовать перенаправлениями, если они имеются. Без неё команда выводит только первоначальный ответ, а с ней – ответ после всех перенаправлений.

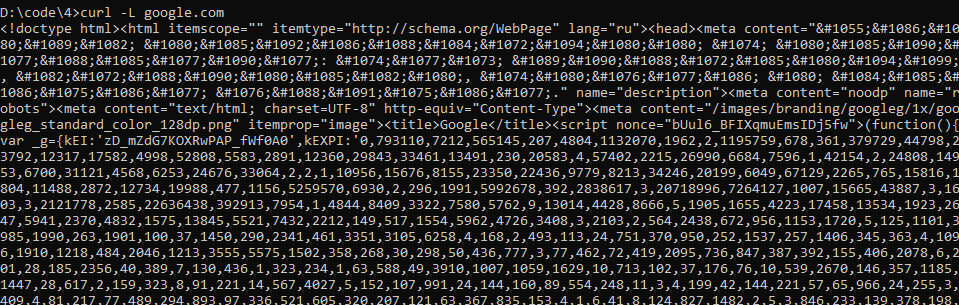


Рисунок 3.8 – Выполнение команды curl с опцией -L

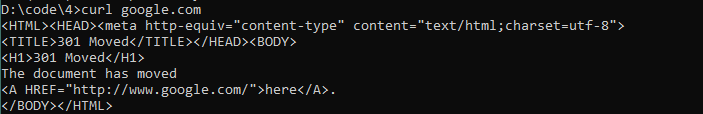


Рисунок 3.9 – Выполнение команды curl без опции -L

Получим заголовки сайтов 10 URL командой curl, опция -I, Рисунок 3.10.

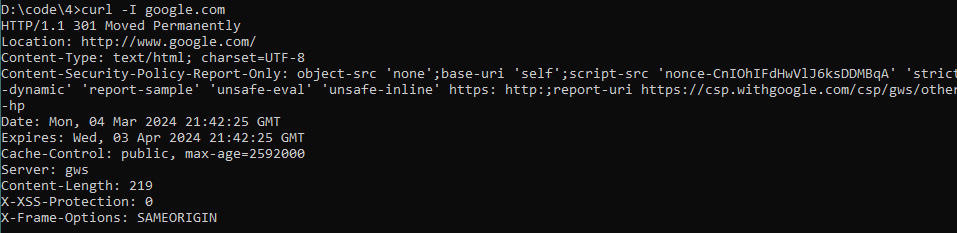


Рисунок 3.10 – Выполнение команды curl с опцией -I

Данное действие повторим еще 9 раз с другими URL. Составим POST-запросы используя команду curl, в виде списка имя=значение, и в виде JSON, для 3 URL, опция -d, Рисунок 3.11.

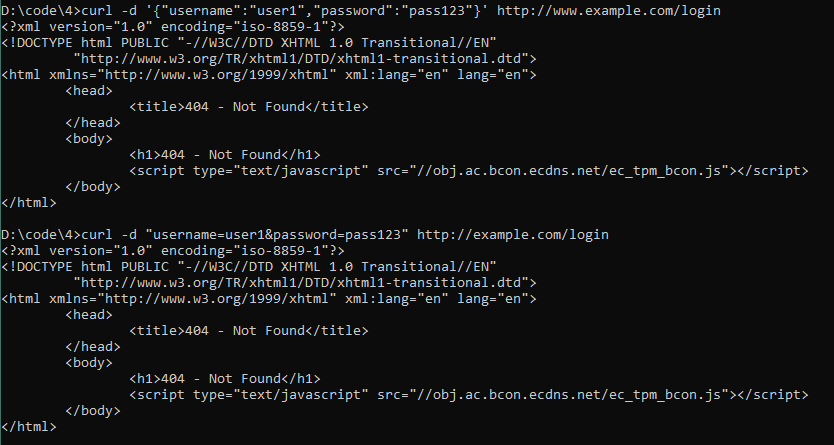


Рисунок 3.11 – Выполнение команды curl с POST запросом

Проделаем эти действия еще с 2 URL. Используем авторизацию для 3 URL, опция -u, Рисунок 3.12.



Рисунок 3.12 – Выполнение команды curl с опцией -u

Проделаем это действие еще с 2 URL. Выведем время работы команды для конкретного URL. Повторим команду 3 раза на одном URL, Рисунок 3.13. Выполним такую же операцию для 5 различных URL, опция -w, Рисунок 3.14.

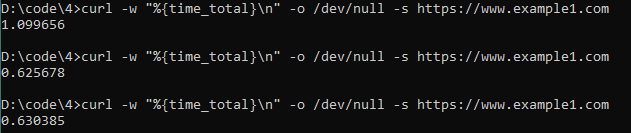


Рисунок 3.13 – Выполнение команды на вывод времени работы, часть 1

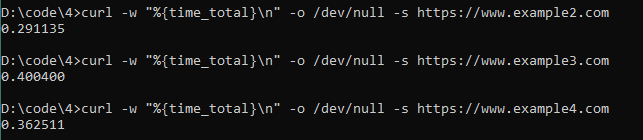


Рисунок 3.14 – Выполнение команды на вывод времени работы, часть 2

Команда была проверена еще на 2 URL. Для 3 URL осуществить имитацию определенного браузера, опция -A, Рисунок 3.15.

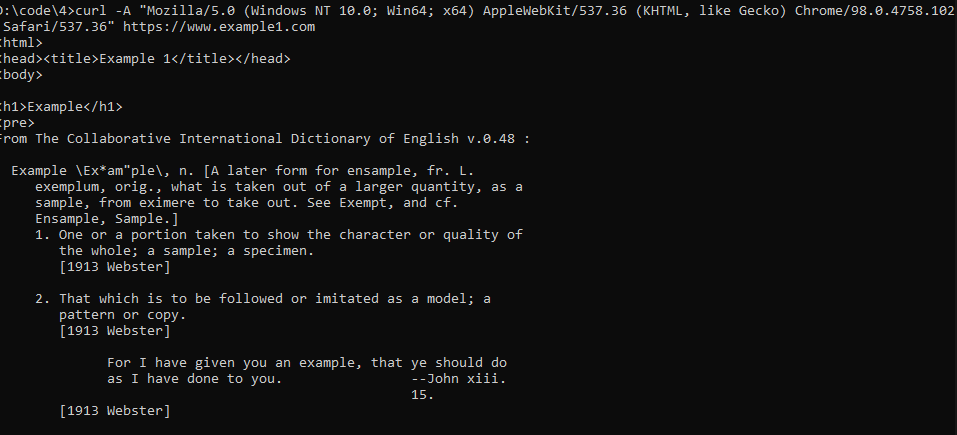


Рисунок 3.15 – Выполнение команды на имитацию определенного браузера

Проделаем эту команду еще на 2 URL. Отправим файлы cookie при скачивании дистрибутива, Рисунок 3.16. По умолчанию при запросе ресурса с помощью curl файлы cookie не отправляются и не сохраняются. Чтобы отправить файлы cookie на сервер, необходимо использовать опцию -b за которой следует имя файла, содержащего файлы cookie, или строку.

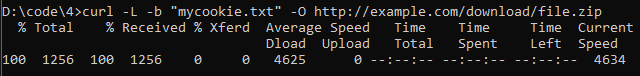


Рисунок 3.16 – Выполнение команды на отправку файлов cookie

Выполним команду по загрузке rpm-файла Oracle Java JDK jdk10.0.2\_linux-x64\_bin.rpm без отправки cookie, Рисунок 3.17.



Рисунок 3.17 – Загрузка rpm-файла без отправки cookie

Теперь, поскольку вы загрузили тот же файл без отправки файла cookie, он, вероятно, будет идентичен по содержанию и размеру. Однако, если сервер Oracle использует механизмы аутентификации на основе файла cookie, отсутствие файла cookie может привести к различному поведению сервера и, возможно, к различиям в содержании загруженного файла.

Самостоятельно выберем ранее не использовавшиеся 5 опций команды curl, Рисунок 3.18-3.22.

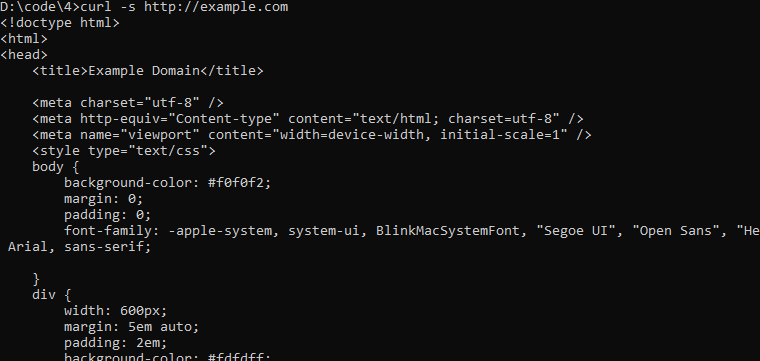


Рисунок 3.18 – Выполнение команды curl -s

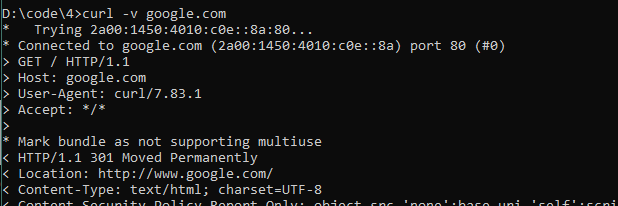


Рисунок 3.19 – Выполнение команды curl -v

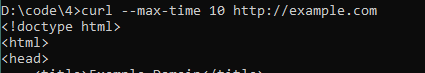


Рисунок 3.20 – Выполнение команды curl –max-time



Рисунок 3.21 – Выполнение команды curl –upload-file

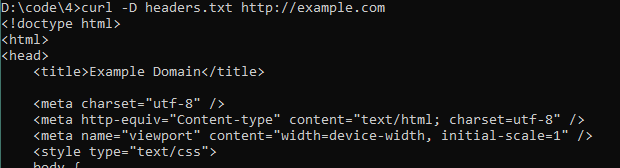


Рисунок 3.22 – Выполнение команды curl -D

**Вывод**

Выполнение работы позволило приобрести навыки запуска Linux-приложений в Windows 10 и запуска Windows-приложений из ОС Linux. В процессе были выполнены команды curl для работы с URL-ресурсами, загрузки файлов, работы с FTP-серверами, управления файлами cookie, имитации браузера, отправки POST-запросов и другие. Работа с утилитой curl позволила эффективно взаимодействовать с веб-ресурсами и изучить основы взаимодействия между операционными системами.

ПРАКТИЧЕСКАЯ РАБОТА №4

**Цель работы:** получить навыки по запуску Linux-приложений в Windows 10, а также запуску Windows-приложений из ОС Linux.

**Задание на практическую работу:**

Ознакомиться и выполнить команды в командной строке Windows 10, а также повторить действия в WSL2 (в Linux. Для выполнения работы понадобится командный интерпретатор cmd. После запуска cmd с помощью команды cd переместиться в удобный каталог для работы (там, где удобнее хранить временные файлы, например, c:\temp, например, cd c:\temp. Все выполняемые команды проверить на работоспособность и привести в отчете. В cmd работает поиск по комбинации CTRL-F. Команда SSH (Windows и Linux)

Возможна такая ситуация, что ключи команды SSH в Windows и Linux могут отличаться.

Выполняя работу, предполагается, что работа ведется с файлами и директорией, содержащей готовый к загрузке на удаленный сервер созданный сайт.

Рекомендуется результат выполнения каждого задания сохранять в отдельную директорию (в заданиях, где копируются файлы и директории), например z1, z2 и т.д.

1. Проверить версию команды ssh

2. Подключиться к localhost (в среде Windows 10. Таким образом, находясь физически на этой машине (Windows 10) можно осуществить имитацию работы с удаленной машиной (localhost), которая также будет этой же машиной. Подключиться, можно указав localhost или как вариант ip-адрес машины 127.0.0.1 или адрес локальной машины, который можно получить, выполнив команду ipconfig и посмотрев IPv4-адрес (будет такого вида 192.168.0.45 или 192.168.145 или 10.0.1.45 или 10.1.1.45.

3. Используя команду tar –version проверьте версию архиватора tar. Используя команду tar /? посмотрите краткую справку. С помощью команды tar --help посмотрите подробную справку. Заархивируйте несколько файлов в архив. Разархивируйте полученный архив. Заархивируйте директорию. Разархивируйте полученный архив.

4. С помощью команды scp осуществить копирование трех файлов (каждый файл отдельно) через ssh c Windows 10 машины на localhost. Этой же командой и в этой же среде скопировать директорию. Скопируйте архив, а затем разархивируйте архив на новом месте.

5. Используя WSL2 (среду Linux), подключиться посредством SSH к машине Windows 10. Если SSH (клиент или сервер) в Linux не установлен, следует осуществить установку.

6. С помощью команды scp осуществить копирование трех файлов (каждый файл отдельно) через ssh из среды Linux на Windows 10 машину. Этой же командой и в этой же среде скопировать директорию. Скопируйте архив, а затем разархивируйте архив на новом месте.

7. С помощью команды sftp осуществить подключение к localhost (в нашем случае это будет remote host по отношению к нашей локальной машине. Ознакомиться с командами sftp, используя команду help. Вывести версию sftp командой version. С помощью команды get загрузить с remote host на localhost три файла. С помощью команды put отправить с локальной машины на remote host три файла. Убедиться по всем командам, что копирование прошло корректно. В sftp доступна команда ls.

8. Осуществить подключение из WSL2 (Linux), используя команду sftp к Windows 10. Выполнить sftp-команды put и get (переслать три файла) по направлению Windows 10.

9. Осуществить подключение из Windows 10 используя команду sftp к WSL2 (Linux. Выполнить sftp-команды put и get (переслать три файла) по направлению WSL2 (Linux)

10. Самостоятельно выбрать ранее не использовавшиеся 2 опции каждой из команд scp, sftp, tar. Выполнить команды в среде Linux и Windows 10 с этими опциями и объяснить полученный результат.

11. Попробуйте скопировать файл со следующего ресурса (или любого другого удаленного доступного для работы ресурса) командами scp, sftp. ftp://ftp.hp.com:21/pub/softpaq/sp40001/sp40001.cva Объясните полученный результат по отношению к команде curl. Попробуйте скопировать этот же файл командой ftp (в среде Linux и Windows 10)

**Выполнение задания**

Проверим версию команды ssh, Рисунок 4.1.



Рисунок 4.1 – Проверка команды ssh

Подключимся к localhost (в среде Windows 10), Рисунок 4.2.



Рисунок 4.2 – Подключение к localhost

Используя команду tar –version проверим версию архиватора tar, через команду tar /? посмотрим краткую справку. С помощью команды tar --help посмотрим подробную справку. Далее заархивируем несколько файлов в архив и разархивируем полученный архив, заархивируем директорию и разархивируем полученный архив, Рисунок 4.3-4.4.

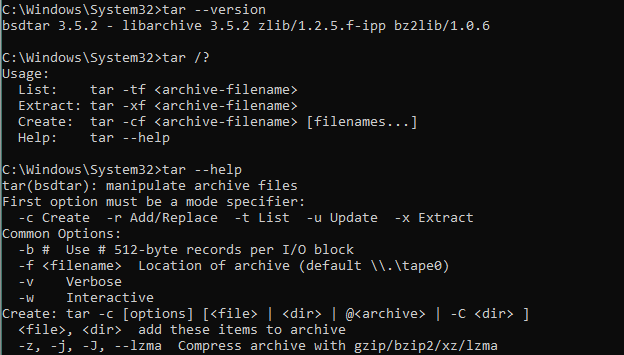


Рисунок 4.3 – Выполнение команды tar, часть 1

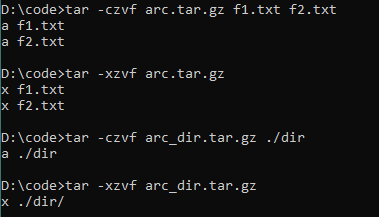


Рисунок 4.4 – Выполнение команды tar, часть 2

С помощью команды scp осуществим копирование трех файлов (каждый файл отдельно) через ssh c localhost на Windows 10 машину. Этой же командой и в этой же среде скопируем директорию. Скопируем архив, а затем разархивируем его на новом месте, Рисунок 4.5-4.6.

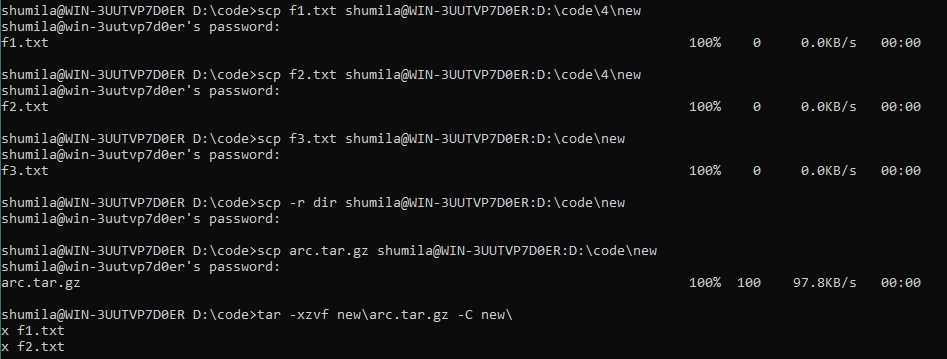


Рисунок 4.5 – Выполнение команды scp

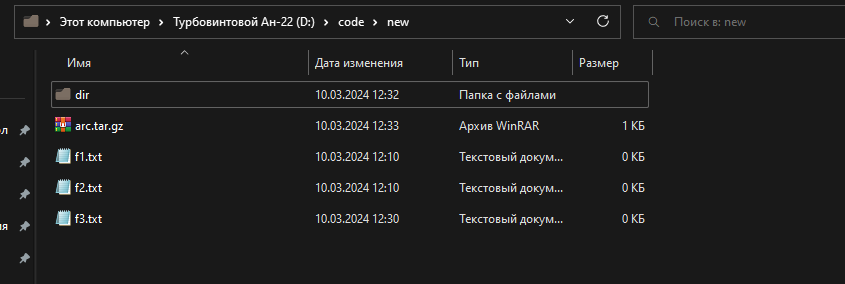


Рисунок 4.6 – Проверка выполнения команды scp

Используя WSL2 (среду Linux), подключимся посредством SSH к машине Windows 10, Рисунок 4.7.



Рисунок 4.7 – Подключение по ssh в WSL к Win10

С помощью команды scp осуществим копирование трех файлов (каждый файл отдельно) через ssh из среды Linux на Windows 10 машину. Этой же командой и в этой же среде скопируем директорию. Скопируем архив, а затем разархивируем архив на новом месте, Рисунок 4.8-4.10.

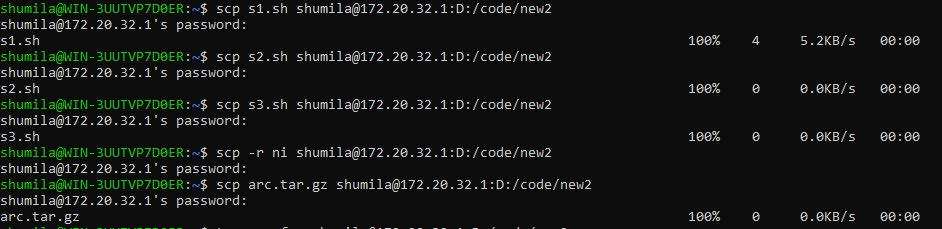


Рисунок 4.8 – Выполнение scp на WSL в Win10, часть 1



Рисунок 4.9 – Выполнение scp на WSL в Win10, часть 2

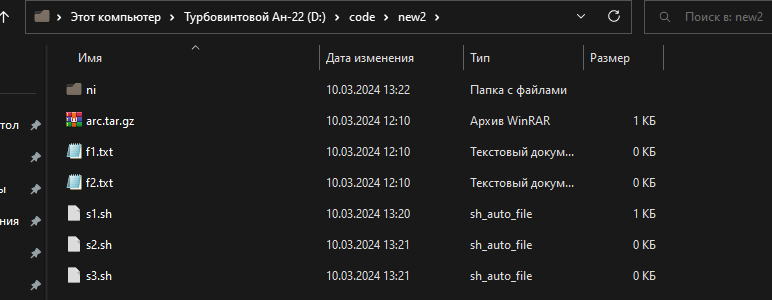


Рисунок 4.10 – Проверка выполнения scp на WSL в Win10, часть 3

С помощью команды sftp осуществим подключение к localhost (в нашем случае это будет remote host по отношению к нашей локальной машине. Ознакомимся с командами sftp, используя команду help. Выведем версию sftp командой version. С помощью команды get загрузим с remote host на localhost три файла. С помощью команды put отправим с локальной машины на remote host три файла, Рисунок 4.11-4.12.

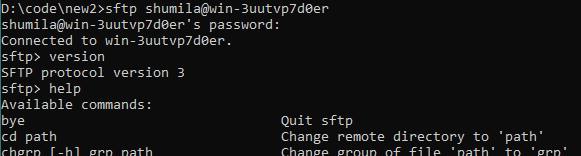


Рисунок 4.12 – Выполнение команды sftp, часть 1

Рисунок 4.13 – Выполнение команды sftp, часть 2

Осуществим подключение из WSL2 (Linux), используя команду sftp к Windows 10. Выполним sftp-команды put и get (переслать три файла) по направлению Windows 10, Рисунок 4.14-4.15.

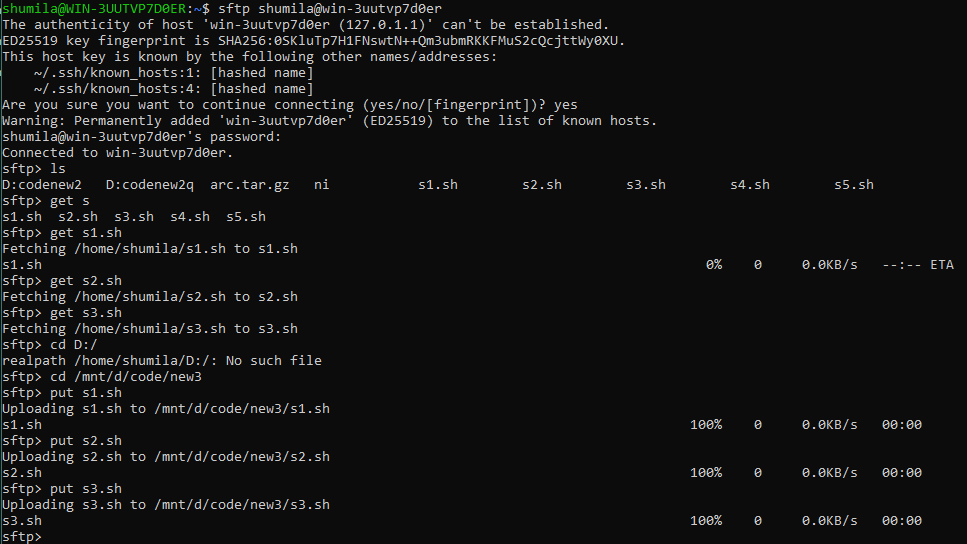


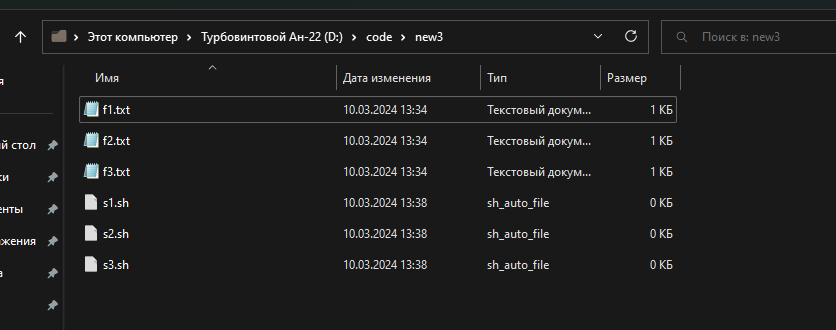
Рисунок 4.14 – Выполнение команд get и put через sftp, часть 1

Рисунок 4.15 – Проверка выполнения команды get

Осуществим подключение из Windows 10 используя команду sftp к WSL2 (Linux. Выполним sftp-команды put и get (переслать три файла) по направлению WSL2 (Linux), Рисунок 4.16.



Рисунок 4.16 – Выполнение команд get и put через sftp, часть 2

Выберем ранее не использовавшиеся 2 опции каждой из команд scp, sftp, tar. Выполним команды в среде Linux и Windows 10 с этими опциями, Рисунок 4.17-4.19.

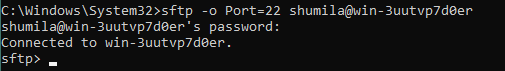


Рисунок 4.17 – Выполнение команд, часть 1

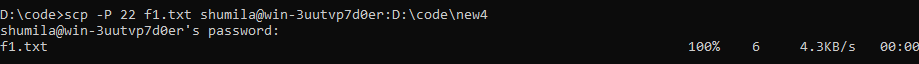


Рисунок 4.18 – Выполнение команды, часть 2



Рисунок 4.19 – Выполнение команды, часть 3

Попробуем скопировать файл со следующего ресурса (ftp://ftp.hp.com:21/pub/softpaq/sp40001/sp40001.cva) командами scp, sftp. Объясните полученный результат по отношению к команде curl. Попробуйте скопировать этот же файл командой ftp (в среде Linux и Windows 10), Рисунок 4.20.

Команды scp и sftp предназначены для копирования файлов по протоколу SSH (Secure Shell), который обычно работает на порте 22. Однако, файл находится на FTP-сервере, который использует протокол FTP (File Transfer Protocol), работающий на порту 21. Попытка использовать scp или sftp для копирования файла с FTP-сервера не удастся, так как эти команды предназначены для работы с SSH, а не с FTP.



Рисунок 4.20 – Получение файла с ресурса

**Вывод**

В результате выполнения работы были получены навыки по запуску Linux-приложений в Windows 10, а также запуску Windows-приложений из ОС Linux. Разница между командами:

Curl - команда используется для передачи данных по сети с использованием различных протоколов, таких как HTTP, HTTPS, FTP и многих других.

Scp - команда используется для копирования файлов между локальной машиной и удаленным сервером через защищенное соединение SSH.

Sftp - команда также используется для передачи файлов между локальной машиной и удаленным сервером, но она работает через протокол SFTP.

Ftp - команда используется для передачи файлов между локальной машиной и удаленным сервером через протокол FTP. В отличие от SCP и SFTP, FTP не обеспечивает шифрование данных, поэтому он менее безопасен для передачи конфиденциальной информации.

ПРАКТИЧЕСКАЯ РАБОТА №5

**Цель работы:** получить навыки работы с Docker-реестрами.

**Задание на практическую работу:**

Произвести следующую работу с Docker-реестрами:

1. Создайте учетную запись в сервисе https://hub.docker.com/ и изучите основные функции и возможности.

2. Используя Docker CLI выведите список 3 образов, имеющих более 5 звезд с названием nginx в консоль.

3. Получите образ с именем nginx, внесите и зафиксируйте изменения в нем.

4. Войдите в реестр Docker Hub.

5. Задайте тэг образу и загрузите образ в Docker Hub.

6. Удалите образ на локальной машине.

7. Произведите получение образа из Docker Hub.

8. Выйдите из реестра Docker Hub.

9. Разверните локально собственный приватный реестр Docker из официального образа registry с GUI интерфейсом из образа konradkleine/docker-registry-frontend:v2

10. Войдите в собственный приватный реестр.

11. Задайте тэг образу и загрузите образ в собственный приватный реестр.

12. Удалите образ на локальной машине.

13. Произведите получение образа из собственного приватного реестра.

14. Выйдите из приватного реестра.

15. С помощью команды curl получите список всех доступных образов в приватном реестре. Выведите список всех тэгов вашего образа.

16. Зайдите в веб-интерфейс приватного реестра, произведите поиск образа и исследуйте его.

**Выполнение задания**

Используя Docker CLI выведем список 3 образов, имеющих более 5 звезд с названием nginx в консоль, Рисунок 5.1.

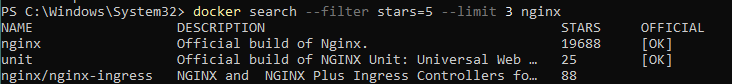


Рисунок 5.1 – Вывод 3 образов, имеющих более 5 звезд с названием nginx

Получим образ с именем nginx, внесем и зафиксируем изменения в нем, Рисунок 5.2-5.3.

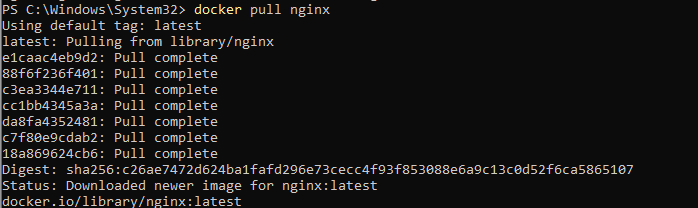


Рисунок 5.2 – Получение образа nginx

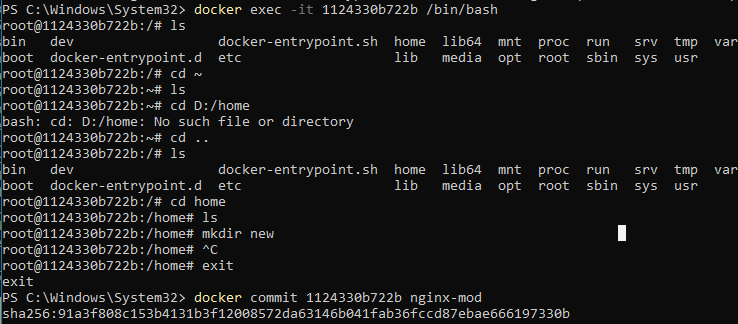


Рисунок 5.3 – Внесение и фиксация изменений в образ

Войдем в реестр Docker Hub, зададим тэг образу и загрузим образ в Docker Hub, Рисунок 5.4-5.5.

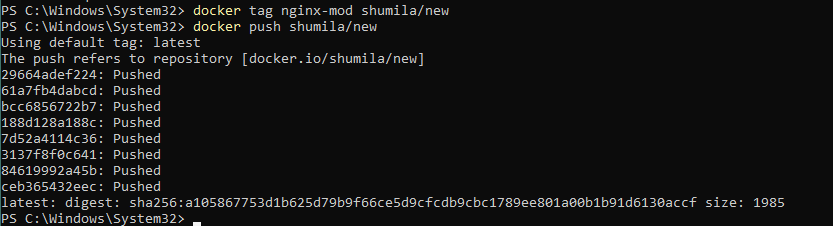


Рисунок 5.4 – Загрузка образа на Docker Hub

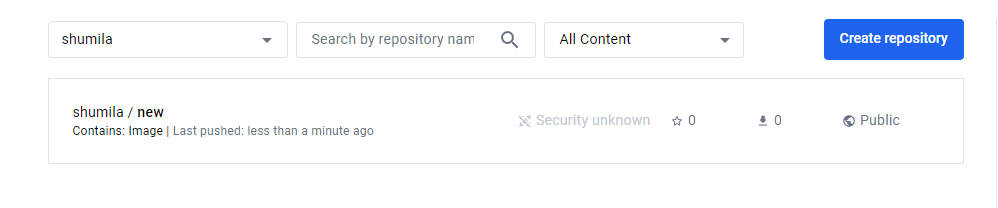


Рисунок 5.5 – Проверка образа на Docker Hub

Удалим образ на локальной машине, Рисунок 5.6.

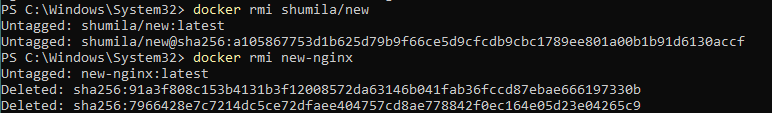


Рисунок 5.6 – Удаление образа на локальной машине

Получим образ и осуществим выход из Docker Hub, Рисунок 5.7.

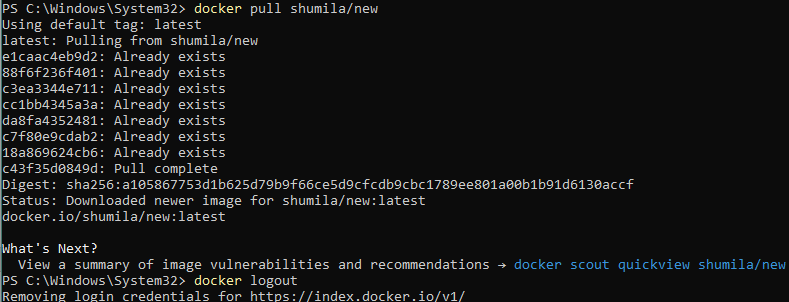


Рисунок 5.7 – Получение образа из Docker Hub

Развернем на 8080 порту локально собственный приватный реестр Docker из официального образа registry (на 5000 порту) с GUI интерфейсом из образа konradkleine/docker-registry-frontend:v2, Рисунок 5.8.

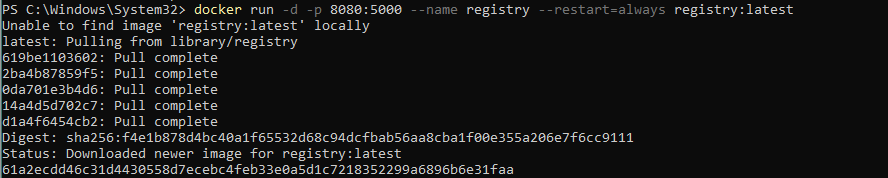


Рисунок 5.8 – Развертывание docker-registry-frontend

Войдем в собственный приватный реестр по адресу localhost:8080, Рисунок 5.9.

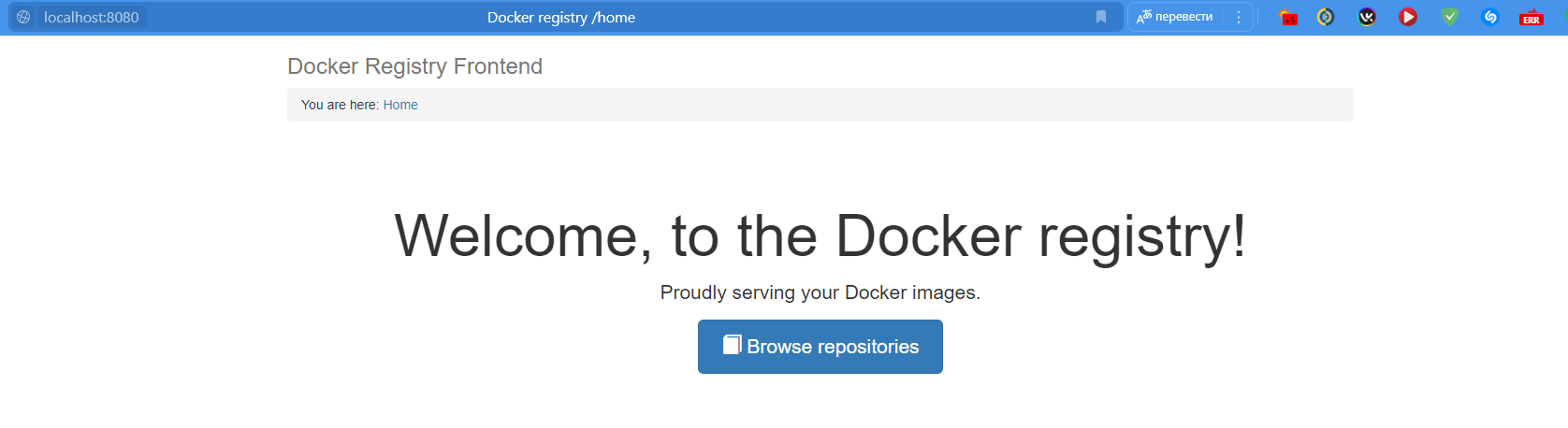


Рисунок 5.9 – Вход в собственный реестр

Зададим тэг образу и загрузим его в собственный приватный реестр, Рисунок 5.10-5.11.

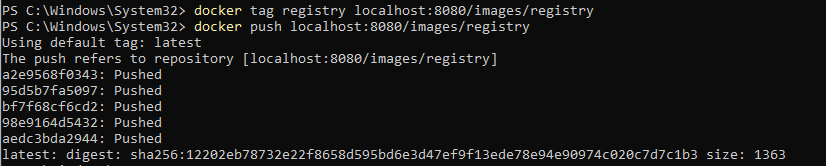


Рисунок 5.10 – Выгрузка образа в приватный реестр

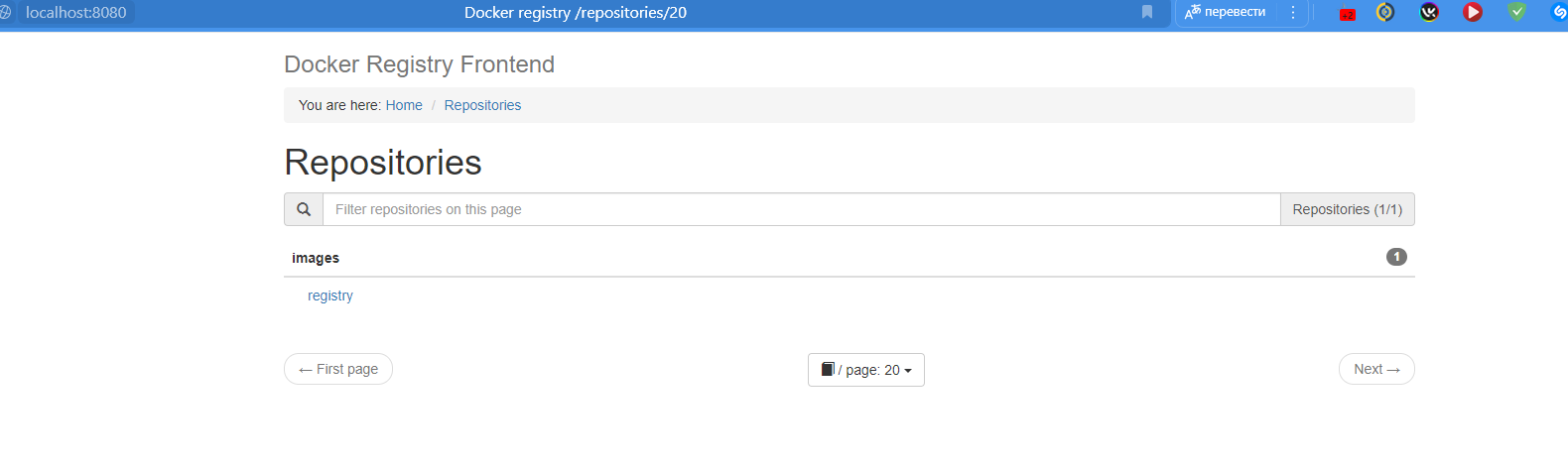


Рисунок 5.11 – Проверка приватного реестра

Удалим образ на локальной машине и получим его из собственного приватного реестра, Рисунок 5.12.

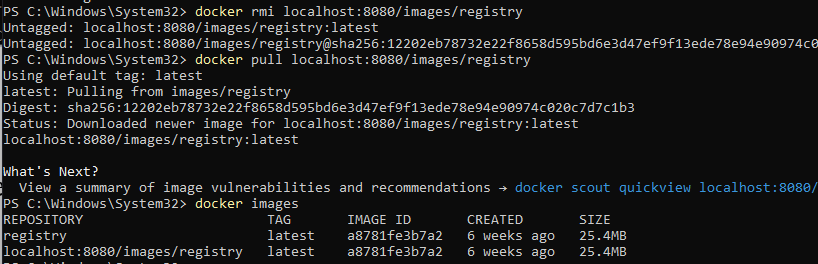


Рисунок 5.12 – Удаление и получение образа из приватного реестра

С помощью команды curl получим список всех доступных образов в приватном реестре. Выведем список всех тэгов вашего образа., Рисунок 5.13.

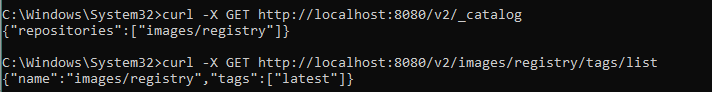


Рисунок 5.13 – Получение образов и тегов образа через curl

Зайдем в веб-интерфейс приватного реестра, найдем образ и исследуем его. Однако при попытке перехода в образ ничего не происходит, Рисунок 5.14.

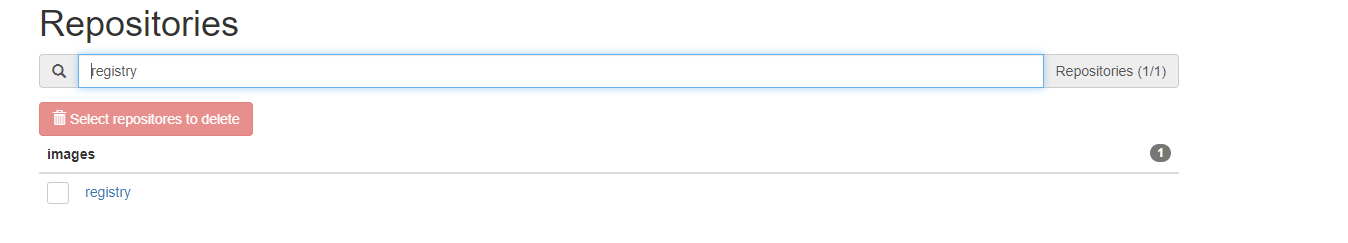


Рисунок 5.14 – Проверка образа в приватном репозитории

**Вывод**

В ходе работы приобретены навыки создания учетной записи и использования функций Docker Hub, освоил работу с Docker CLI для управления образами, включая внесение изменений и загрузку образов в Docker Hub, а также удаление их с локальной машины. Также осуществлено развертывание и администрирование собственного приватного реестра Docker, в том числе загрузку и получение образов из этого реестра и изучение их через веб-интерфейс.

ПРАКТИЧЕСКАЯ РАБОТА №6

**Цель работы:** получить навыки сборки Docker-контейнера с использованием Docker-compose.

**Задание на практическую работу:**

Используя Docker-compose создать контейнеры для работы:

1. Веб-сервера apache, так чтобы можно было запускать самостоятельно созданный сайт. Продумайте компоновку контейнера на свое усмотрение.
2. СУБД mysql, при обращении к которой из самостоятельно созданного приложения можно было помещать и читать данные в базу данных. Продумайте компоновку контейнера на свое усмотрение.
3. Поместить в контейнер собственное приложение. Продумайте компоновку контейнера на свое усмотрение.
4. Самостоятельно придумать содержимое контейнера, которое принесет практическую пользу.

**Выполнение задания**

Создадим контейнер для веб-сервера apache, так чтобы можно было запускать самостоятельно созданный сайт, Рисунок 6.1 – 6.4. (рис. 6.1).

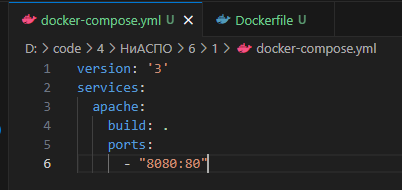


Рисунок 6.1 – Файл docker-compose.ym

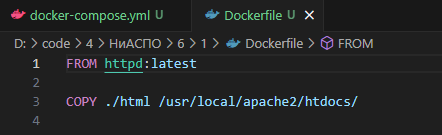


Рисунок 6.2 – Файл dockerfile

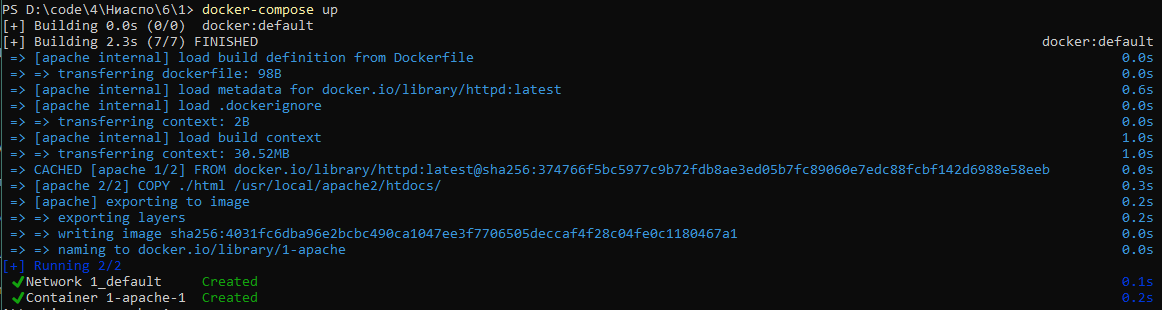


Рисунок 6.3 – Запуск контейнера

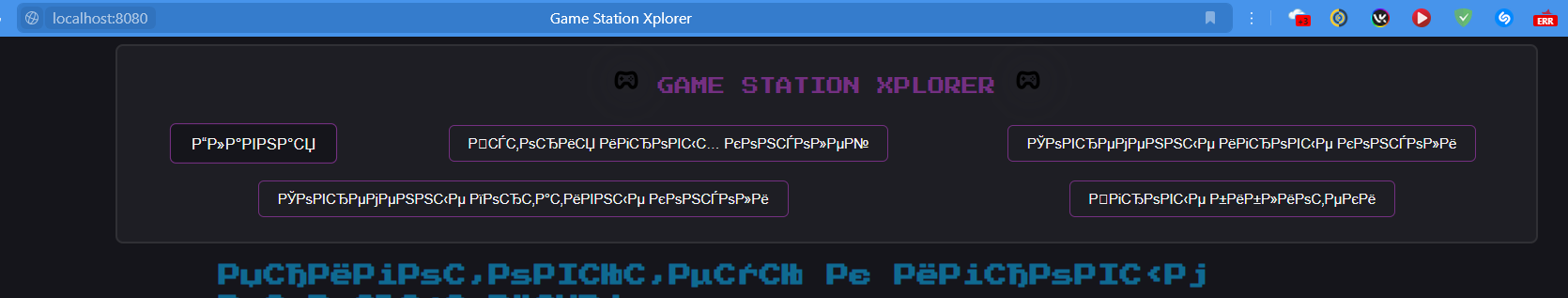


Рисунок 6.4 – Результат работы контейнера

Создадим контейнер для СУБД, Рисунок 6.5 – 6.8.

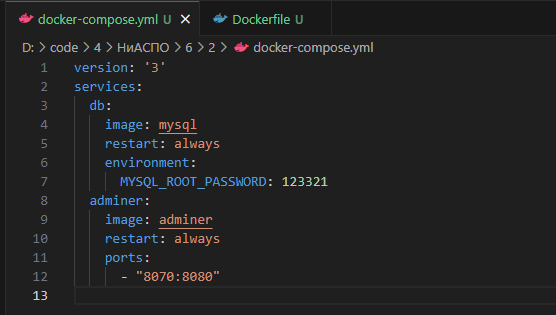


Рисунок 6.5 – Образ проекта на DockerHub

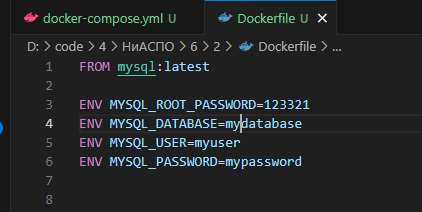


Рисунок 6.6– Файл docker-compose.yml

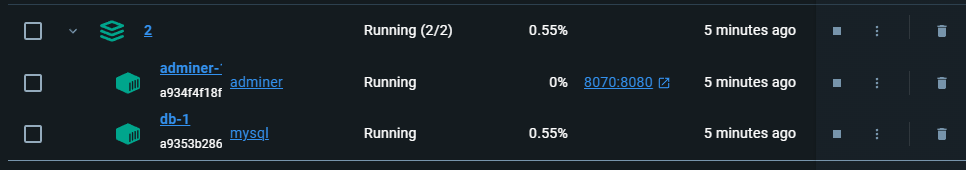


Рисунок 6.7 – Запущенный контейнер

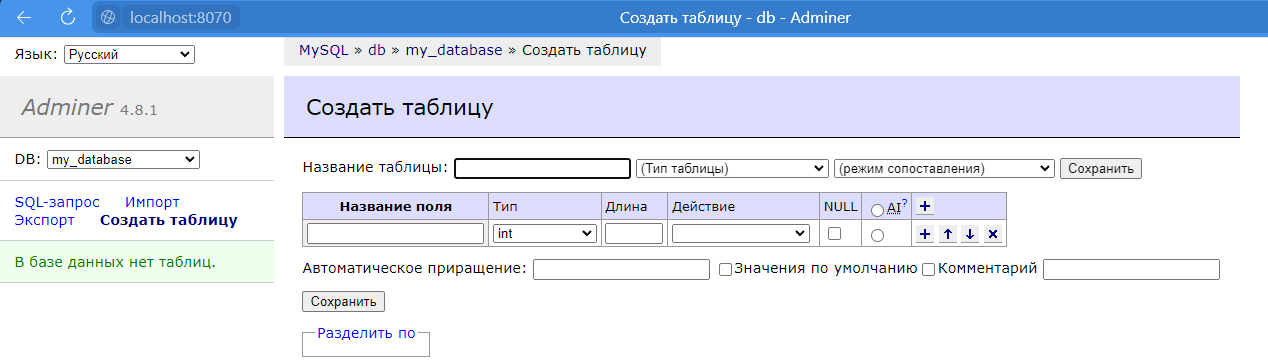


Рисунок 6.8 – Результат работы контейнера

Поместим в контейнер собственное приложение, Рисунок 6.9 – 6.14.

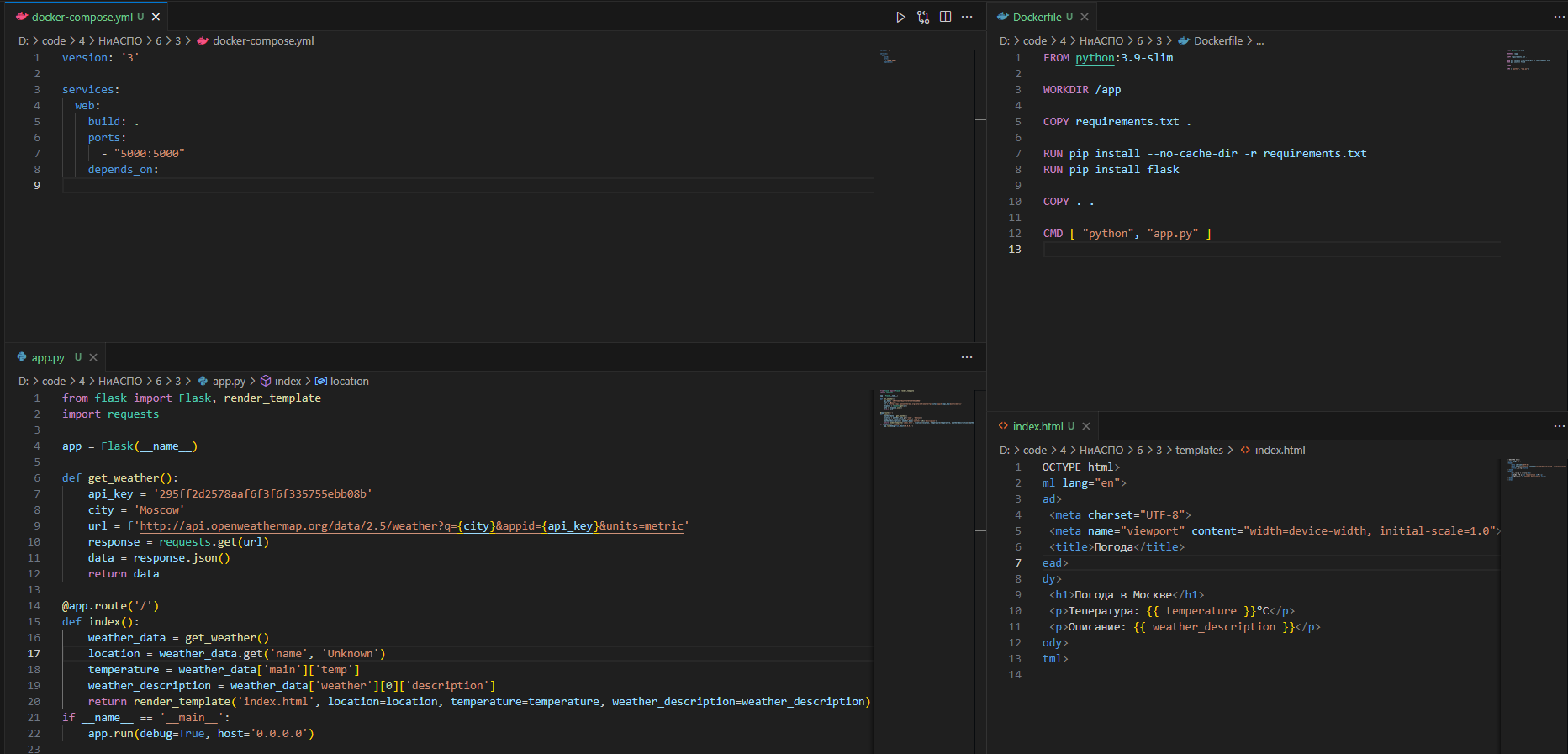


Рисунок 6.5 – Файлы для запуска приложения



Рисунок 6.6– Запуск приложения

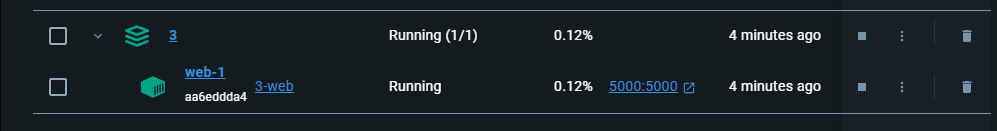


Рисунок 6.7 – Запущенный контейнер

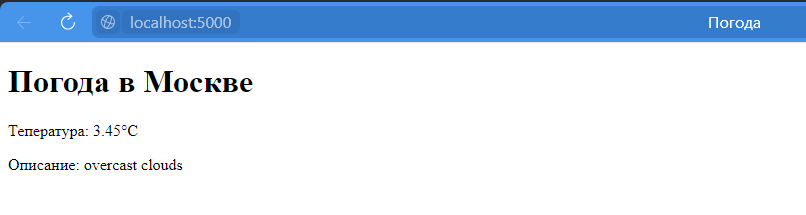


Рисунок 6.8 – Результат работы контейнера

**Вывод**

В результате выполнения работы были успешно приобретены навыки сборки Docker-контейнера с использованием Docker-compose. Были созданы контейнеры для работы веб-сервера Apache, СУБД MySQL и собственного приложения. Каждый контейнер был сконфигурирован в соответствии с потребностями задачи, обеспечивая возможность запуска и взаимодействия с созданным сайтом и базой данных. Также было предложено содержимое контейнера, способствующее практической пользе. Полученные навыки позволят эффективно управлять процессом разработки и развертывания приложений с использованием Docker.

ПРАКТИЧЕСКАЯ РАБОТА №7

**Цель работы:** получить навыки сборки Docker-контейнера с использованием Docker-compose.

**Задание на практическую работу:**

Используя инструкцию, представленную в общих сведениях Docker-compose создать следующие сервисы. Работу каждого сервиса продемонстрировать преподавателю.

1. Повторить пример, который был рассмотрен в разделе «Общие сведения». Проект назвать docker\_py. Обязательно продемонстрировать сервис при обсуждении с преподавателем отчёта практической работы.
2. Создать аналогичный сервис как в примере, рассмотренном в разделе «Общие сведения», но вместо языка Python использовать язык JavaScript. Проект назвать docker\_js. Обязательно продемонстрировать сервис при обсуждении с преподавателем отчёта практической работы.
3. Создать сервис как в примере, рассмотренном в разделе «Общие сведения», но вместо сервера на Python использовать веб сервер Apache. Проект назвать docker\_apache. Обязательно продемонстрировать сервис при обсуждении с преподавателем отчёта практической работы.
4. Создать сервис как в примере, рассмотренном в разделе «Общие сведения», но вместо сервера на Python использовать веб сервер Nginx. Проект назвать docker\_nginx. Обязательно продемонстрировать сервис при обсуждении с преподавателем отчёта практической работы.
5. Создать сервис в состав которого входят следующие компоненты apache, mysql, phpMyAdmin. Проект назвать docker\_phpmyadmin. Обязательно продемонстрировать сервис при обсуждении с преподавателем отчёта практической работы.

**Выполнение задания**

Для начала запустим тестовый пример, как показано на Рисунке 7.1.



Рисунок 7.1 – Запуск тестового примера

Теперь поменяем python на javascript, перепишем Dockerfile, docker-compose.yml и запустим, как показано на Рисунке 7.2.

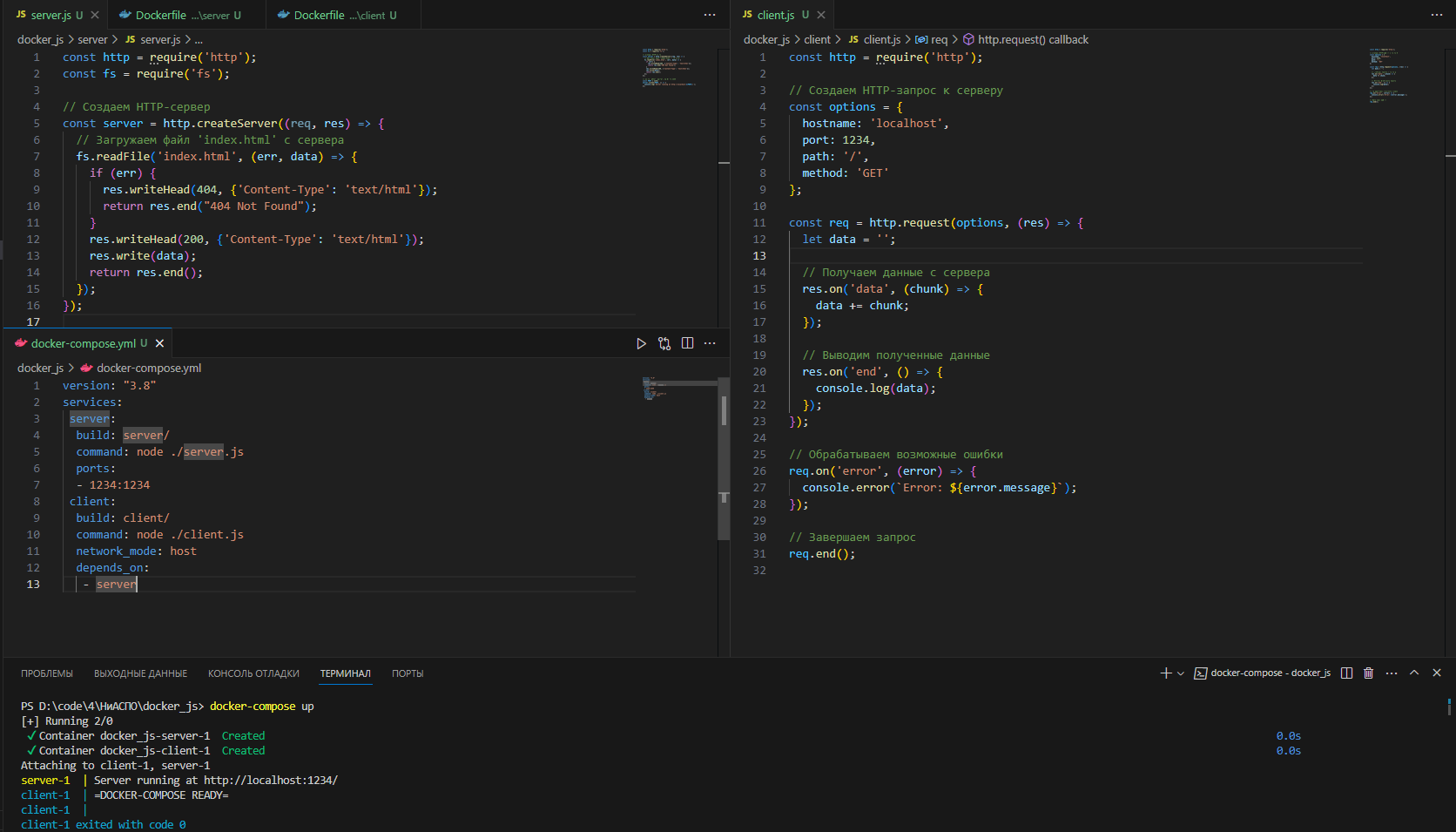


Рисунок 7.2 – Сервер и клиент на javascript

Вернем код, написанный на python, но теперь будем использовать вместо сервера на python сервер на apache, как показано на Рисунке 7.3.

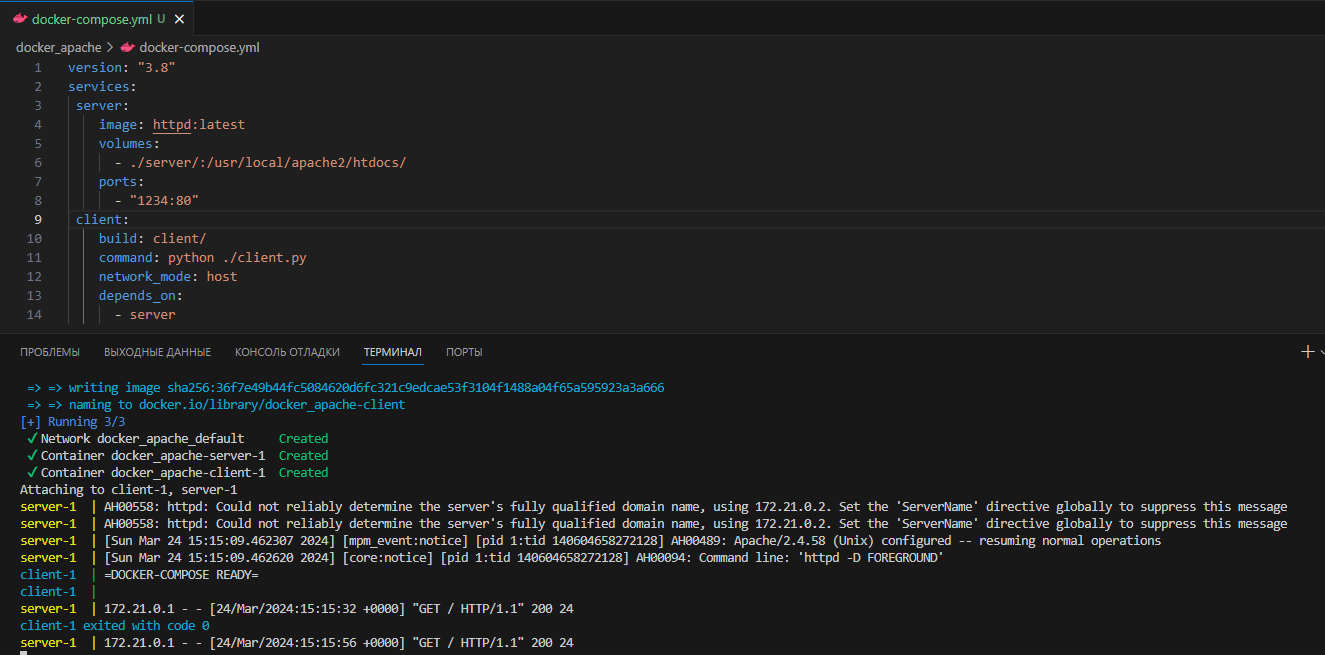


Рисунок 7.3 – Сервер на apache

Теперь вместо сервера apache будем использовать nginx, как показано на Рисунке 7.4.

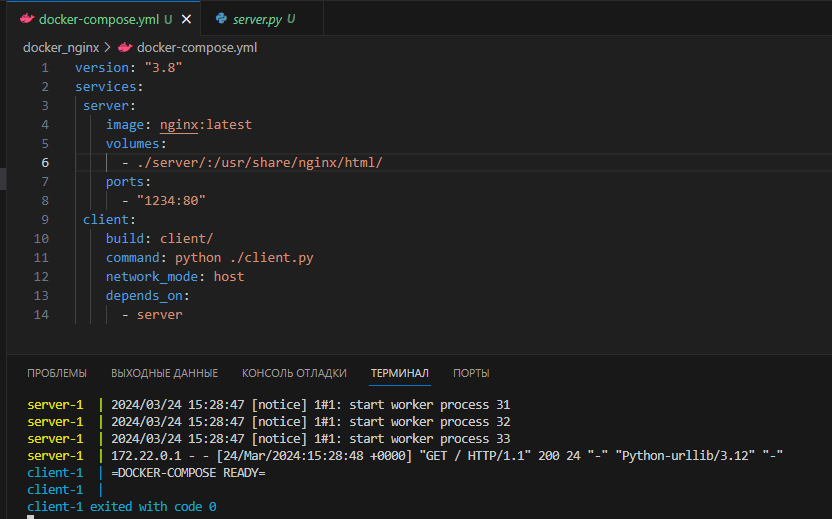


Рисунок 7.4 – Сервер на nginx

Соберем сервер с помощью apache, mysql и phpmyadmin, как показано на Рисунке 7.5.



Рисунок 7.5 – Новый docker-compose.yml

На Рисунке 7.6 показан успешный запуск phpmyadmin.

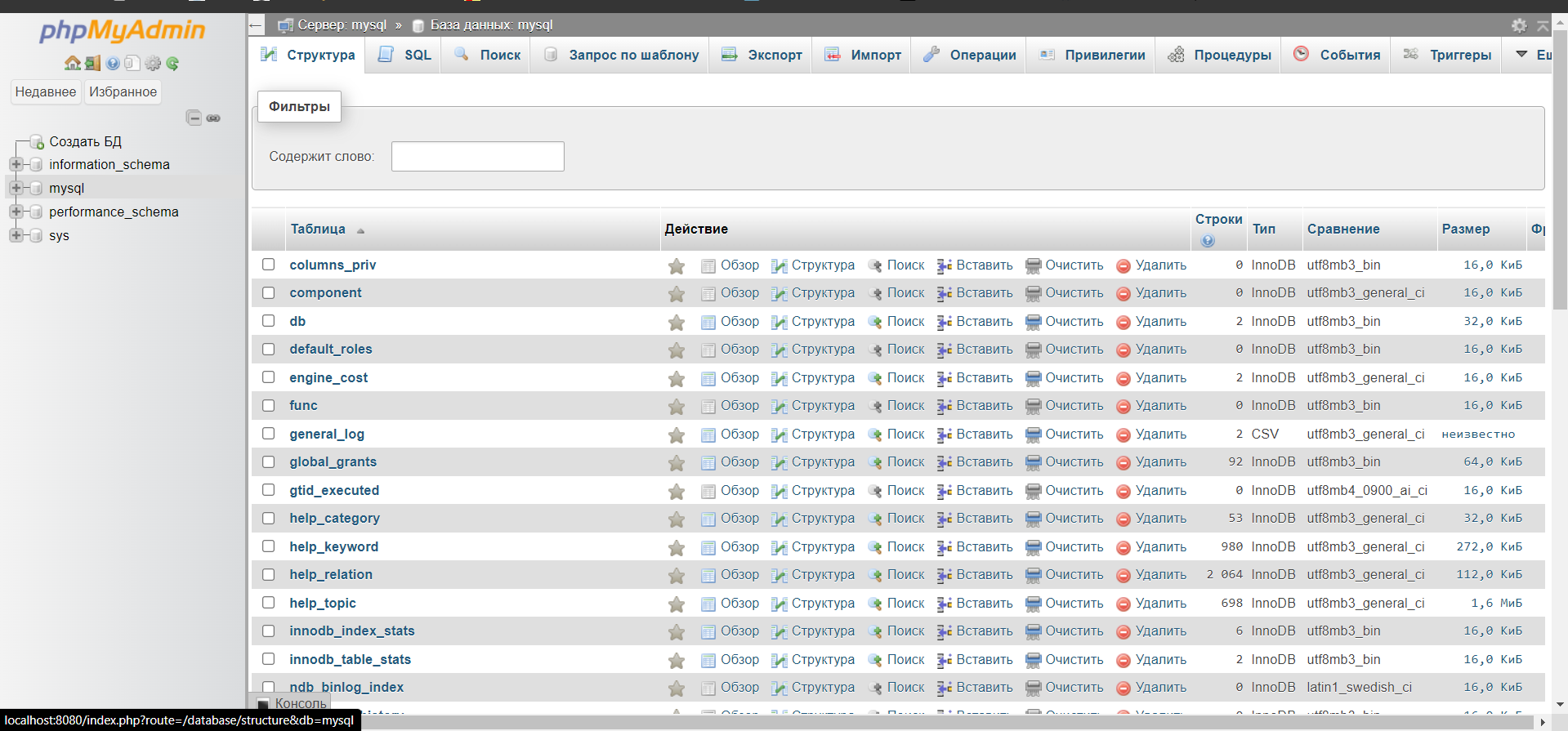


Рисунок 7.6 – Запуск phpmyadmin в браузере

**Вывод**

В ходе выполнения работы были успешно достигнуты поставленные цели, а именно приобретение навыков сборки Docker-контейнера с использованием Docker-compose. Были созданы сервисы согласно предложенным заданиям, включая повторение примера с использованием языка Python, а также создание аналогичных сервисов с использованием языка JavaScript, веб-серверов Apache и Nginx, а также составного сервиса с apache, mysql и phpMyAdmin. Полученные навыки позволят эффективно управлять сборкой и управлением Docker-контейнеров для различных приложений и сервисов.

ПРАКТИЧЕСКАЯ РАБОТА №8

**Цель работы:** получить навыки сборки Docker-контейнера с использованием Docker-compose.

**Задание на практическую работу:**

Развернуть контейнеры на хостинге. Развернутую архитектуру на хостинге объяснить и продемонстрировать преподавателю. Предполагается, что следующие несколько практических работ будут ориентированы на работу с cloud хостингом. На cloud хостинге в своем аккаунте выполнить следующие задания:

1. Развернуть docker-контейнер с Jenkins на cloud-хостинге. Объяснить, для чего предназначен Jenkins. Продемонстрировать пример применения Jenkins-контейнера на cloud-хостинге. Проект назвать docker\_jenkins. Обязательно продемонстрировать на компьютере выполненную работу при обсуждении с преподавателем отчёта практической работы.

2. Развернуть docker-контейнер с Maven на cloud-хостинге. Объяснить, для чего предназначен Maven. Продемонстрировать пример применения Maven-контейнера на cloud-хостинге. Проект назвать docker\_maven. Обязательно продемонстрировать на компьютере выполненную работу при обсуждении с преподавателем отчёта практической работы.

3. Развернуть docker-контейнер с CMS Drupal на cloud-хостинге. Объяснить, для чего предназначен Drupal. Продемонстрировать пример применения Drupal-контейнера на cloud-хостинге. Проект назвать docker\_drupal. Обязательно продемонстрировать на компьютере выполненную работу при обсуждении с преподавателем отчёта практической работы.

4. Развернуть docker-контейнер с СУБД Couchbase на cloud хостинге. Объяснить, для чего предназначен Couchbase. Продемонстрировать пример применения Couchbaseконтейнера на cloud-хостинге. Проект назвать docker\_couchbase. Обязательно продемонстрировать на компьютере выполненную работу при обсуждении с преподавателем отчёта практической работы.

5. Развернуть docker-контейнер с Node.js на cloud-хостинге. Объяснить, для чего предназначен Node.js. Продемонстрировать пример применения Node.js-контейнера на cloud-хостинге. Проект назвать docker\_nodejs. Обязательно продемонстрировать на компьютере выполненную работу при обсуждении с преподавателем отчёта практической работы.

**Выполнение задания**

Jenkins – программная система с открытым исходным кодом, предназначенная для обеспечения CI/CD (непрерывная интеграция/непрерывная доставка) процесса. Для настройки pipeline пишется специальный конфигурационный файл Jenkinsfile на языке Groovy. Развернем docker-контейнер с Jenkins на cloud-хостинге и продемонстрируем пример применения Jenkins-контейнера на cloud-хостинге.

На облачном хостинге проверим деплой нашего контейнера, Рисунок 8.1. Также покажем демонстрацию работы Jenkins на облачном хостинге, Рисунок 8.2.

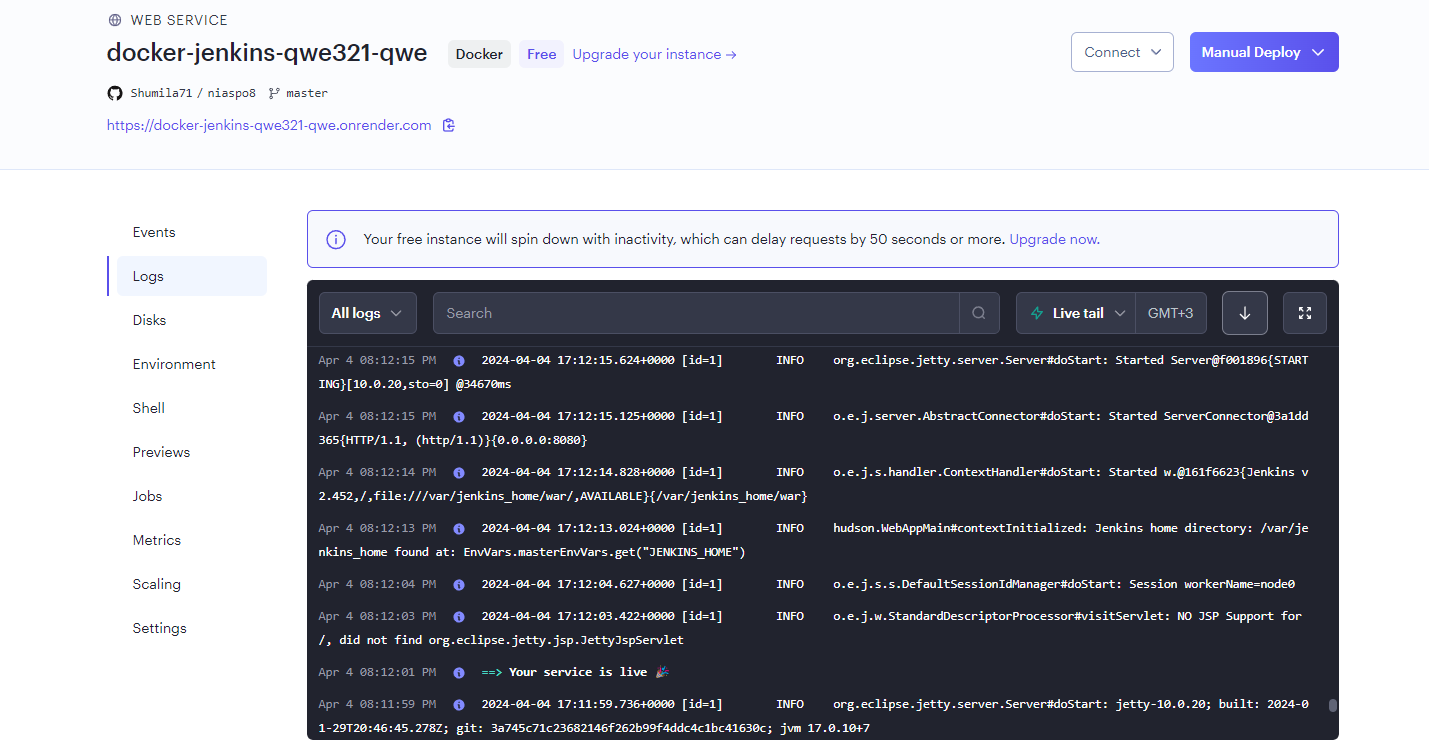


Рисунок 8.1 – Деплой контейнера с Jenkins

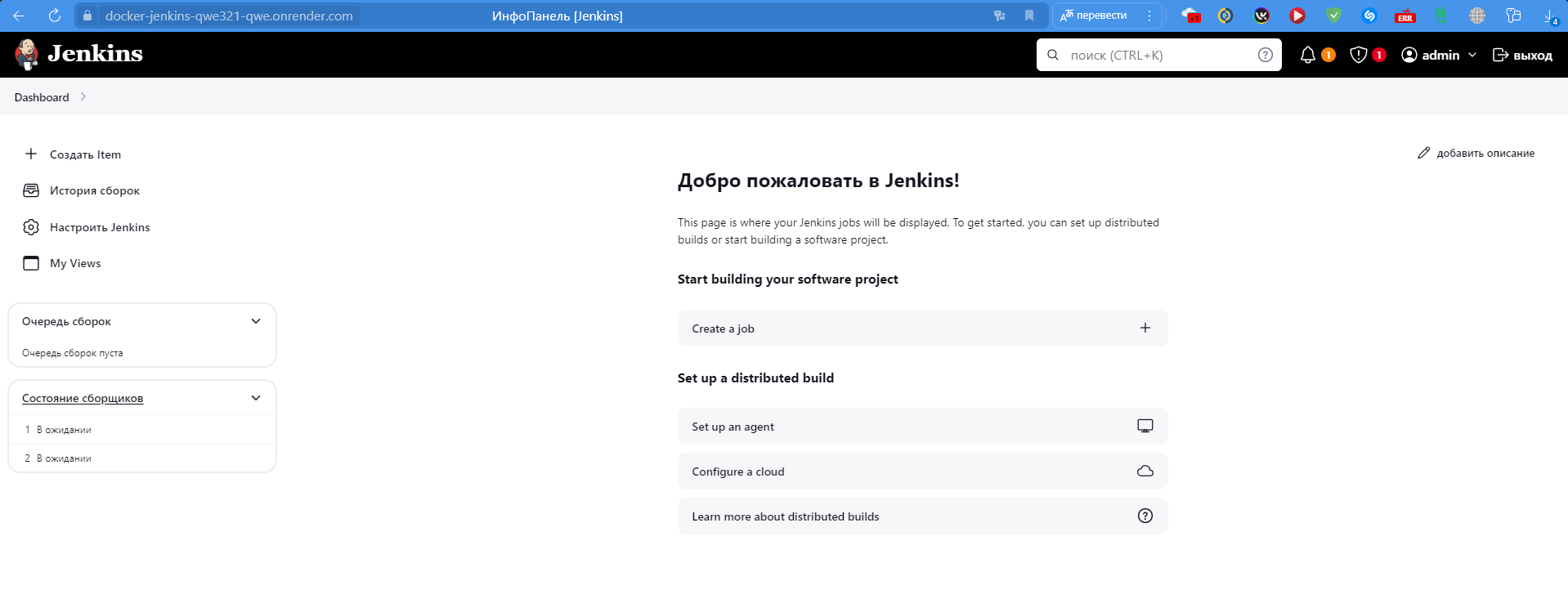


Рисунок 8.2 – Демонстрация работы Jenkins

Maven — это инструмент для автоматической сборки проектов на Java и других языках программирования. Он помогает разработчикам правильно подключить библиотеки и фреймворки, управлять их версиями, выстроить структуру проекта и составить к нему документацию. Развернем docker-контейнер с Maven на cloud-хостинге и продемонстрируем пример применения Maven-контейнера на cloud-хостинге.

Для этого напишем небольшой проект, выводящий текст в консоль. Dockerfile для запуска, Рисунок 8.4. На облачном хостинге проверим деплой нашего контейнера и покажем демонстрацию работы Maven на облачном хостинге, Рисунок 8.5.

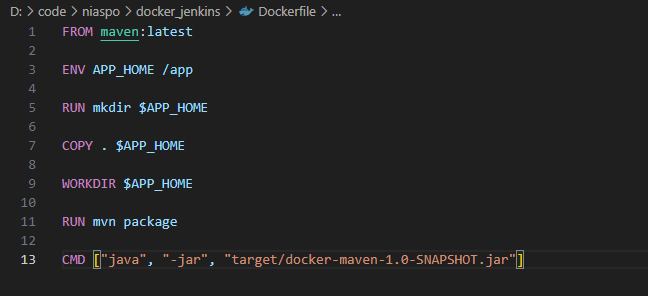


Рисунок 8.3 – Dockerfile для Maven



Рисунок 8.5 – Демонстрация работы Maven

CMS Drupal – система управления содержимым, используемая также как каркас для веб-приложений. Развернем docker-контейнер с Drupal на cloud-хостинге и продемонстрируем пример применения Drupal-контейнера на cloud-хостинге.

Для этого напишем Dockerfile для создания окружения внутри контейнера и установки необходимых пакетов, Рисунок 8.6. На облачном хостинге проверим деплой нашего контейнера, Рисунок 8.7. Также покажем демонстрацию работы CMS Drupal на облачном хостинге, Рисунок 8.8.

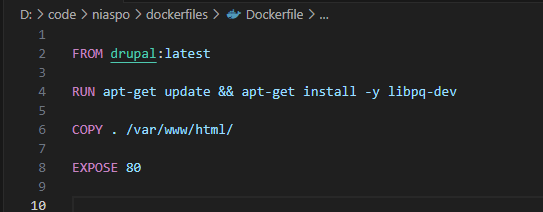


Рисунок 8.6 – Dockerfile для CMS Drupal

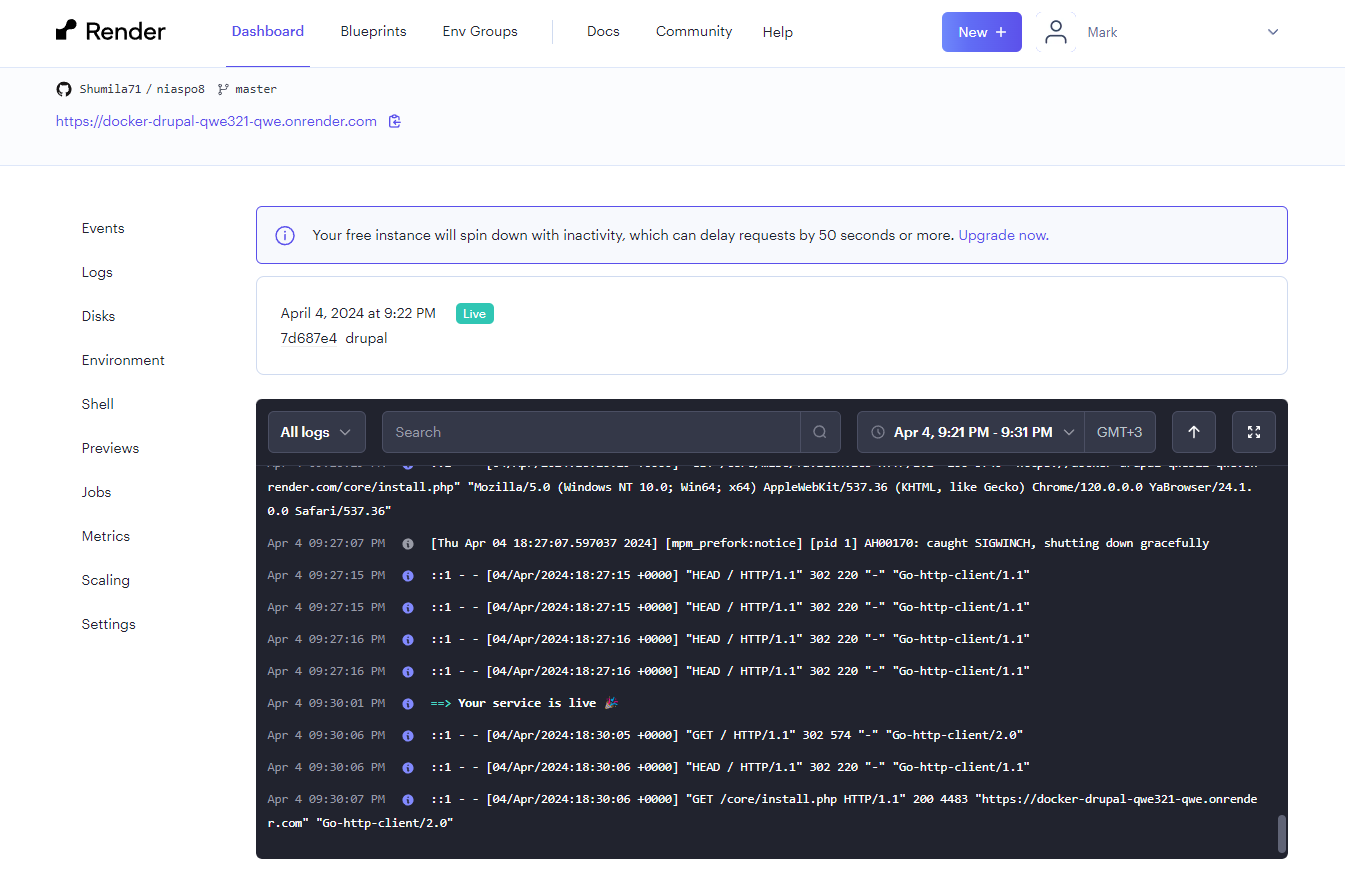


Рисунок 8.7 – Деплой контейнера с CMS Drupal

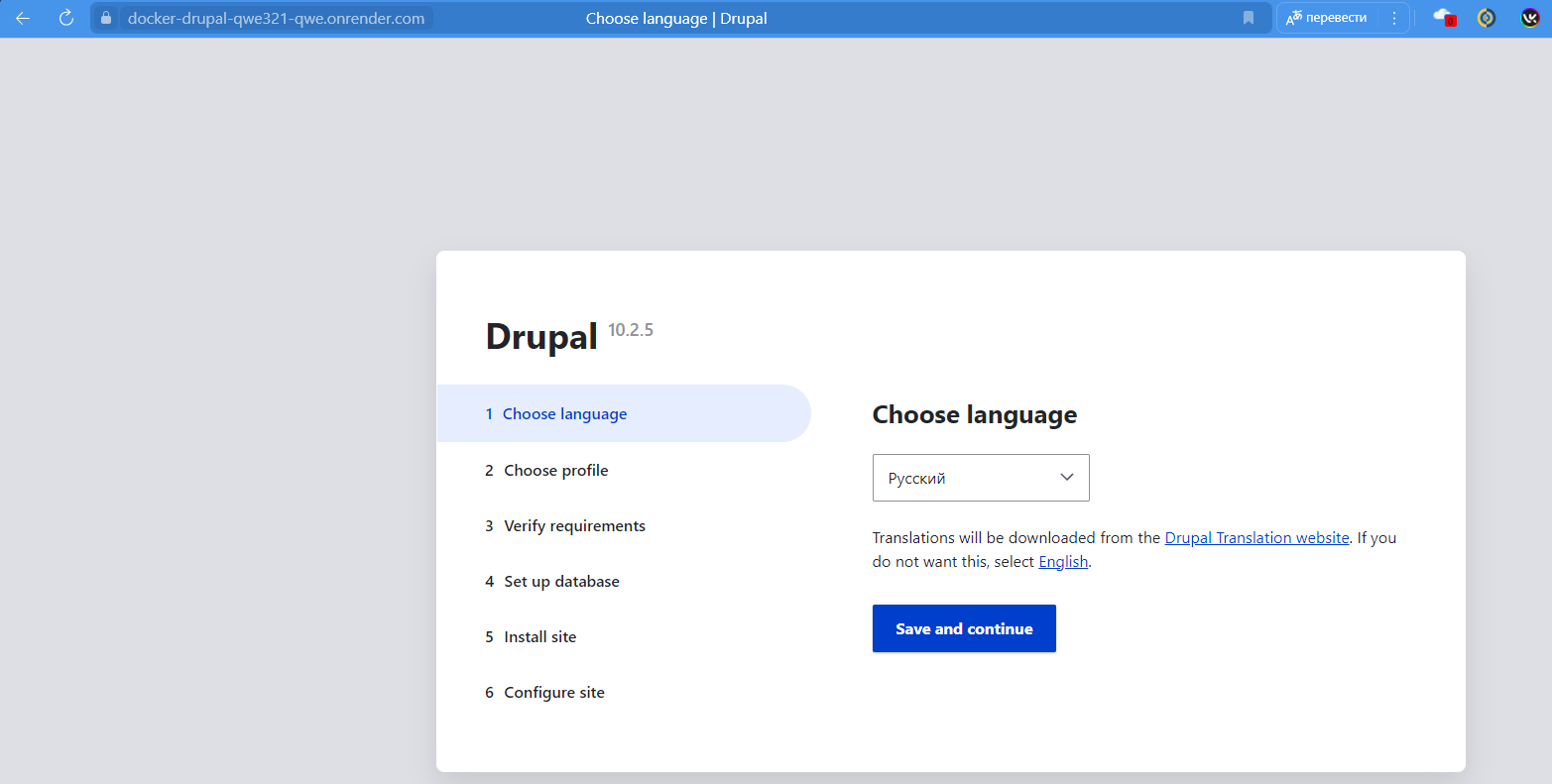


Рисунок 8.8 – Демонстрация работы CMS Drupal

Couchbase – СУБД типа NoSQL. Развернем docker-контейнер с Couchbase на cloud-хостинге и продемонстрируем пример применения Couchbase-контейнера на cloud-хостинге.

Для этого напишем Dockerfile для создания окружения внутри контейнера и установки необходимых пакетов, Рисунок 8.9. На облачном хостинге проверим деплой нашего контейнера, Рисунок 8.10.

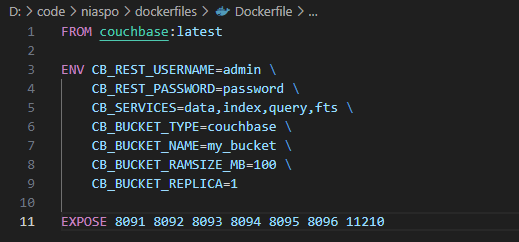


Рисунок 8.9 – Dockerfile для Couchbase

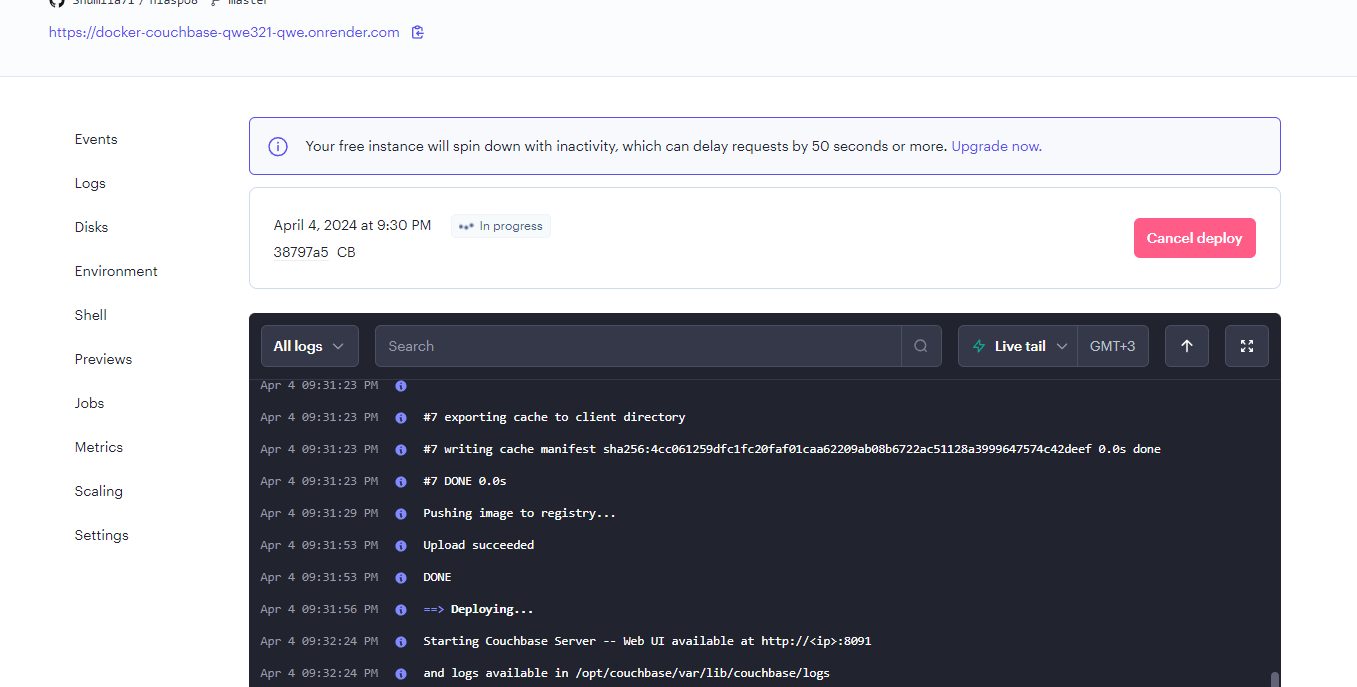


Рисунок 8.10 – Деплой контейнера с Couchbase

Node.js – программная платформа, основанная на движке V8, превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Развернем docker-контейнер с Node.js на cloud-хостинге и продемонстрируем пример применения Node.js-контейнера на cloud-хостинге.

Для этого напишем небольшой проект на JavaScript и Dockerfile для создания окружения внутри контейнера и установки необходимых пакетов, Рисунок 8.11. На облачном хостинге проверим деплой нашего контейнера, Рисунок 8.12.

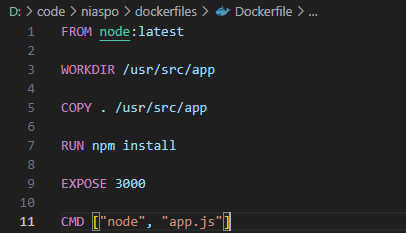


Рисунок 8.11 – Dockerfile для Node.js

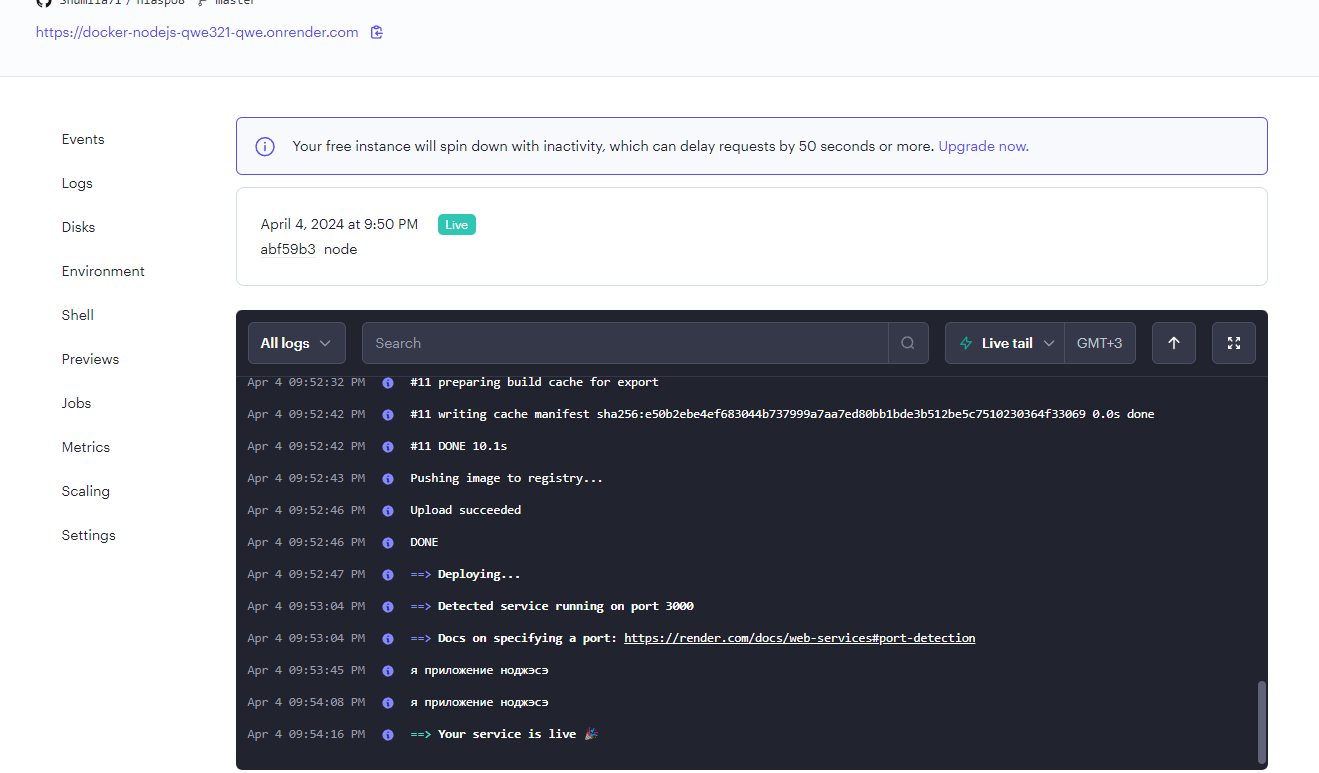


Рисунок 8.12 – Деплой контейнера с Node.js

**Вывод**

В результате выполнения работы освоены навыки развертывания Docker-контейнеров на облачном хостинге. Изучено, как использовать Docker-compose для создания и управления контейнерами, а также развертка различных сервисов, таких как Jenkins, Maven, Drupal, Couchbase и Node.js, на облачной инфраструктуре. В результате практики улучшено понимание работы с Docker в облачной среде, что позволяет эффективнее управлять и развивать инфраструктуру при работе с проектами.

ПРАКТИЧЕСКАЯ РАБОТА №9

**Цель работы:** получить навыки создания сетевого соединения и передачи данных между docker-контейнерами посредством localhost.

**Задание на практическую работу:**

Создать два докер-контейнера, которые будут одновременно "слушать" localhost на разных портах. Пользователь делает запрос к контейнеру №1. Далее контейнер №1 делает запрос к контейнеру №2. Контейнер №2 обработает запрос, передаст результат контейнеру №1. Контейнер №1 передаст результат пользователю. API может быть реализовано на любой технологии. В итоге необходимо настроить систему из нескольких контейнеров, связанных друг с другом.

**Выполнение задания**

Напишем приложение на питоне, будем обращаться к localhost, где будет находиться контейнер, делающий запрос на второй контейнер, и покажет пользователю возвращенный результат запроса. Напишем для этого docker-compose и 2 Dockerfile, и саму API, Рисунок 9.1-9.4. Все взаимодействие будет внутри сети niaspo9.

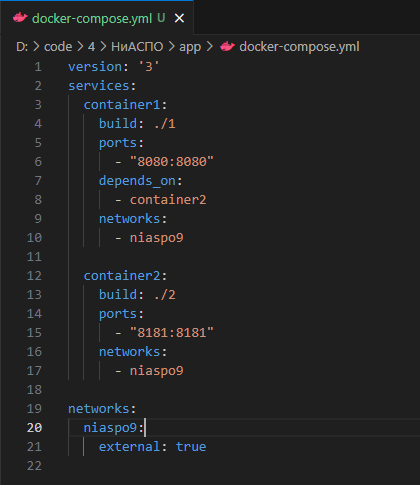


Рисунок 9.1 – Файл docker-compose

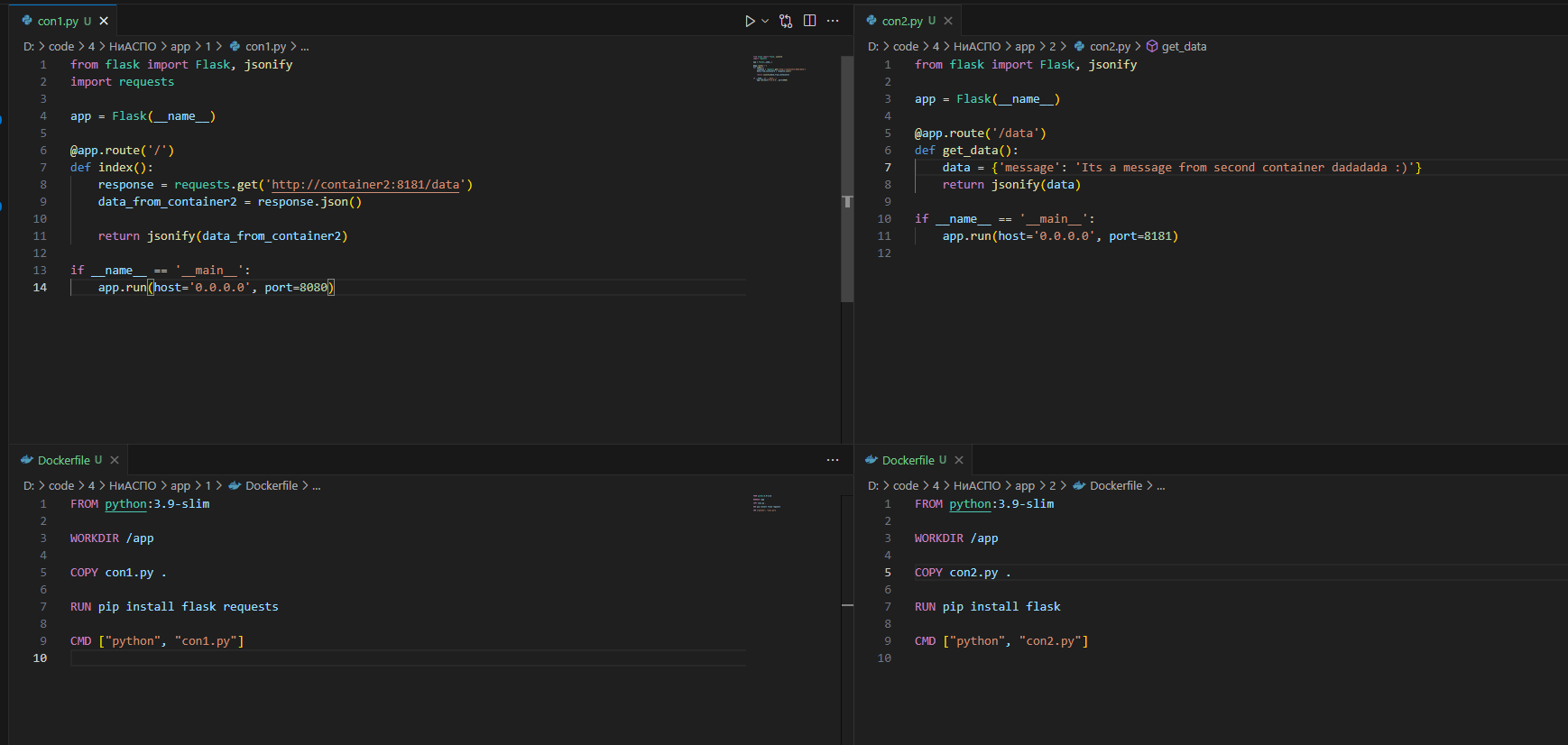


Рисунок 9.2 – API и Dockerfile

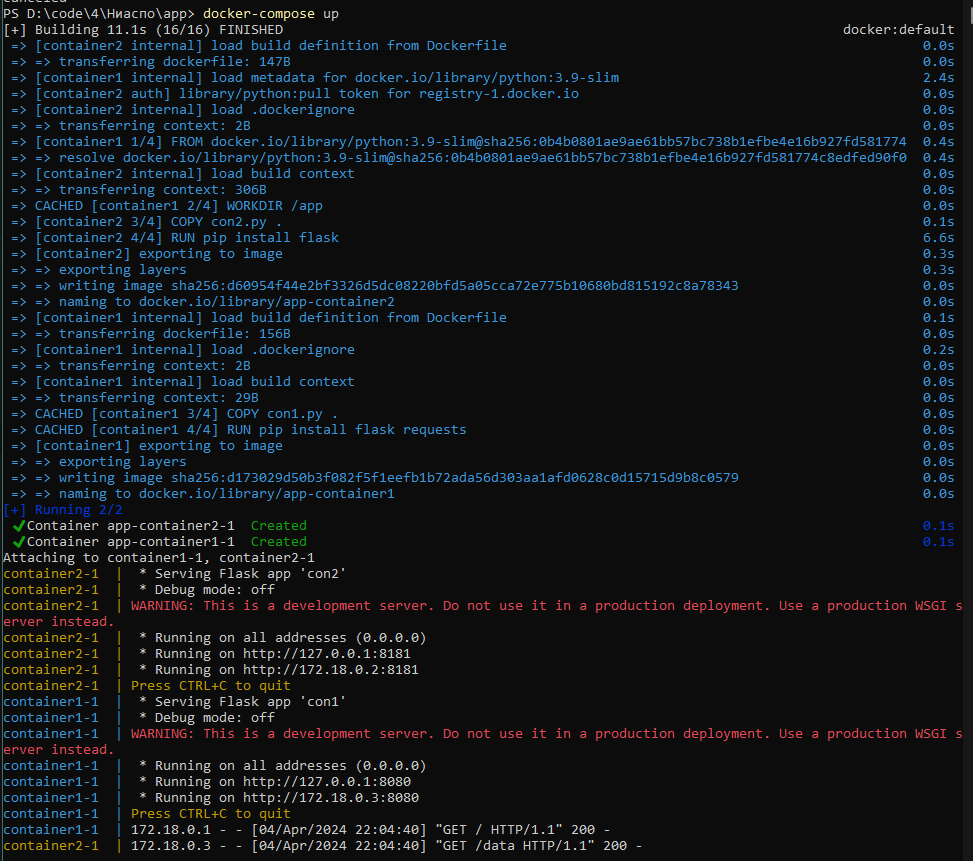


Рисунок 9.3 – Запуск docker-compose

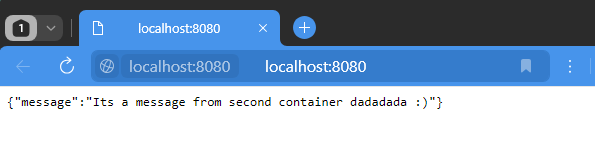


Рисунок 9.4 – Результат работы

**Вывод**

В ходе выполнения данной практической работы были освоены навыки создания сетевого соединения и передачи данных между Docker-контейнерами через localhost. Этот опыт помог понять принципы настройки взаимодействия между контейнерами, включая установку "слушателей" на различных портах и обработку запросов между ними. Полученные навыки представляют ценность для работы с микросервисной архитектурой и организации сложных систем, состоящих из взаимосвязанных контейнеров.

ПРАКТИЧЕСКАЯ РАБОТА №10

**Цель работы:** научиться настраивать кластер K8s и настроить оркестрацию докер контейнеров с использованием K8s.

**Задание на практическую работу**:

Развернуть кластер Kubernetes. Создать несколько докер контейнеров. Можно взять, например, созданные контейнеры в предыдущих работах (например, где контейнеры передавали друг-другу данные) или любые другие контейнеры из других работ. Можно создать и новые контейнеры. Запустить докер-контейнеры внутри кластера Kubernetes.

**Выполнение задания**

Для разворачивания кластера Kubernetes, необходимо установить kubectl, API клиент Kubernetes, Рисунок 10.1, и minikube, инструмент для запуска одноузлового кластера Kubernetes на локальной машине, Рисунок 10.2.



Рисунок 10.1 – Проверка установленного kubectl

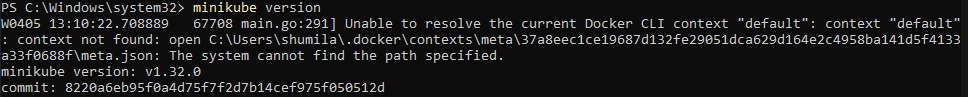


Рисунок 10.2 – Проверка установленного minikube

Создадим под с nginx и проверим поды, Рисунок 10.3. Видно, что создался под с nginx. Подключимся к созданному поду по ssh и через curl, предварительно узнать ip-адрес пода, прописав команду kubectl describe nginx, выведем домашнюю страницу nginx внутри виртуальной машины, на которой запущен minikube, Рисунок 10.5. Развернем и проверим работу пода с registry, Рисунок 10.5-10.6.

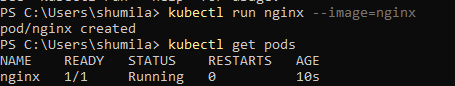


Рисунок 10.3 – Создание пода nginx

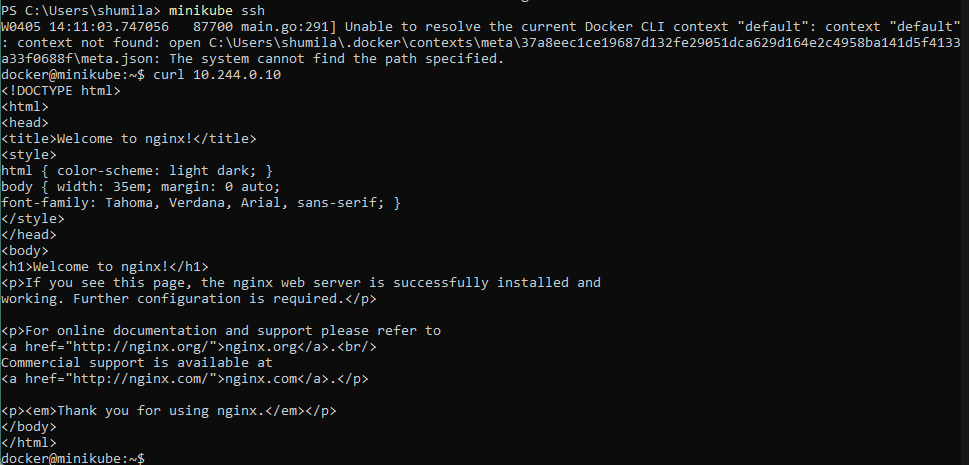


Рисунок 10.4 – Проверка работы созданного пода nginx

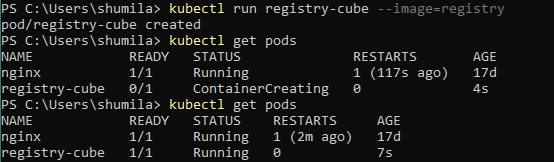


Рисунок 10.5 – Создание пода registry



Рисунок 10.6 – Проверка работы созданного пода registry

Создадим на Flask приложение, которое будет выводить нам “Hello, World!” и Dockerfile для него, Рисунок 10.7. Затем развернем данный под и протестируем его работу, Рисунок 10.8-10.9.

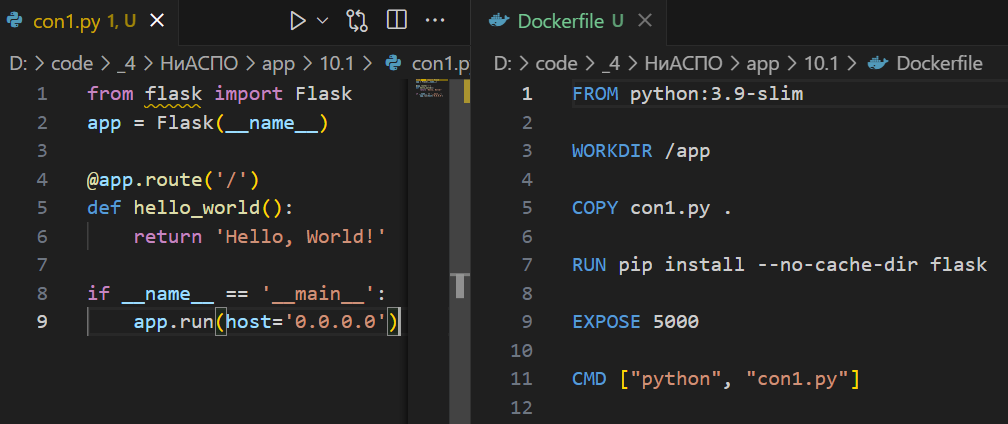


Рисунок 10.7 – Flask приложение и Dockerfile

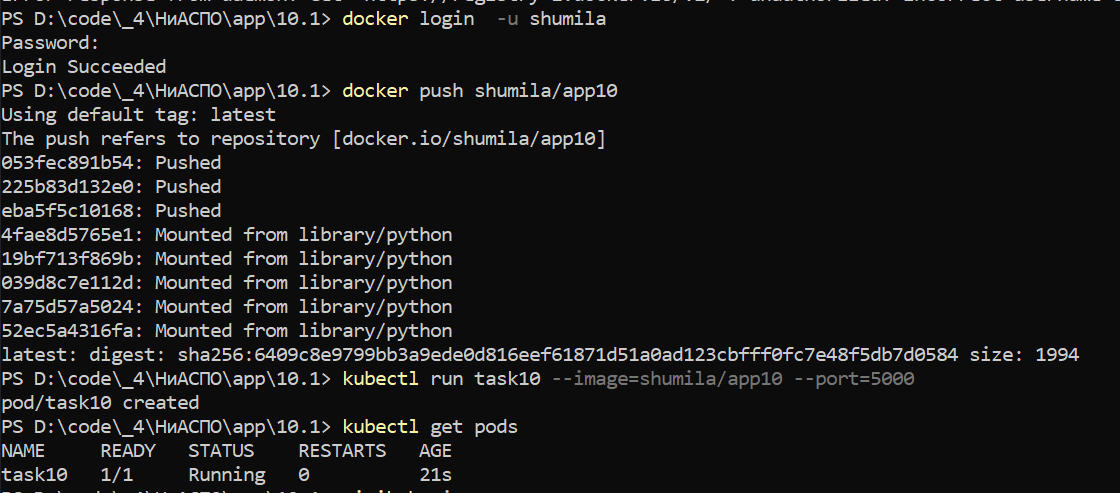


Рисунок 10.8 – Создание пода

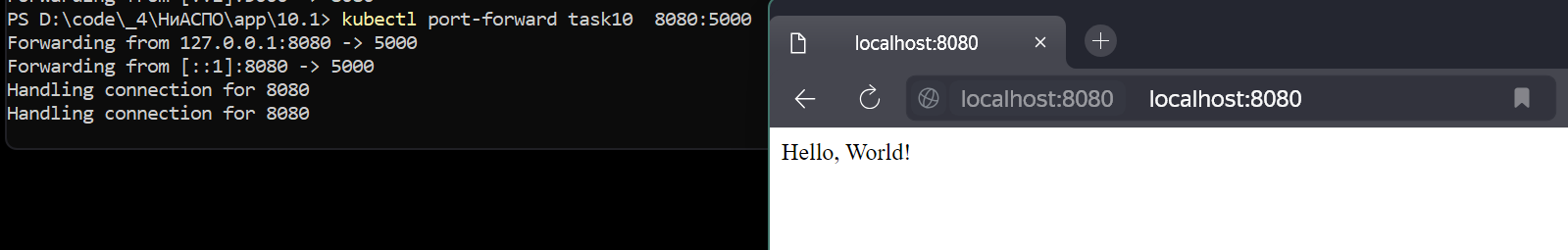


Рисунок 10.9 – Проверка работы созданного пода task10

Создадим 3 контейнера на Flask, которые будут общаться между собой, напишем для них Dockerfile и YAML файл для создания подов, затем протестируем их работу, Рисунок 10.10-10.14. Первый контейнер содержит сообщение, второй его запрашивает, и добавляет, затем третий запрашивает сообщение от второго и добавляет свое.

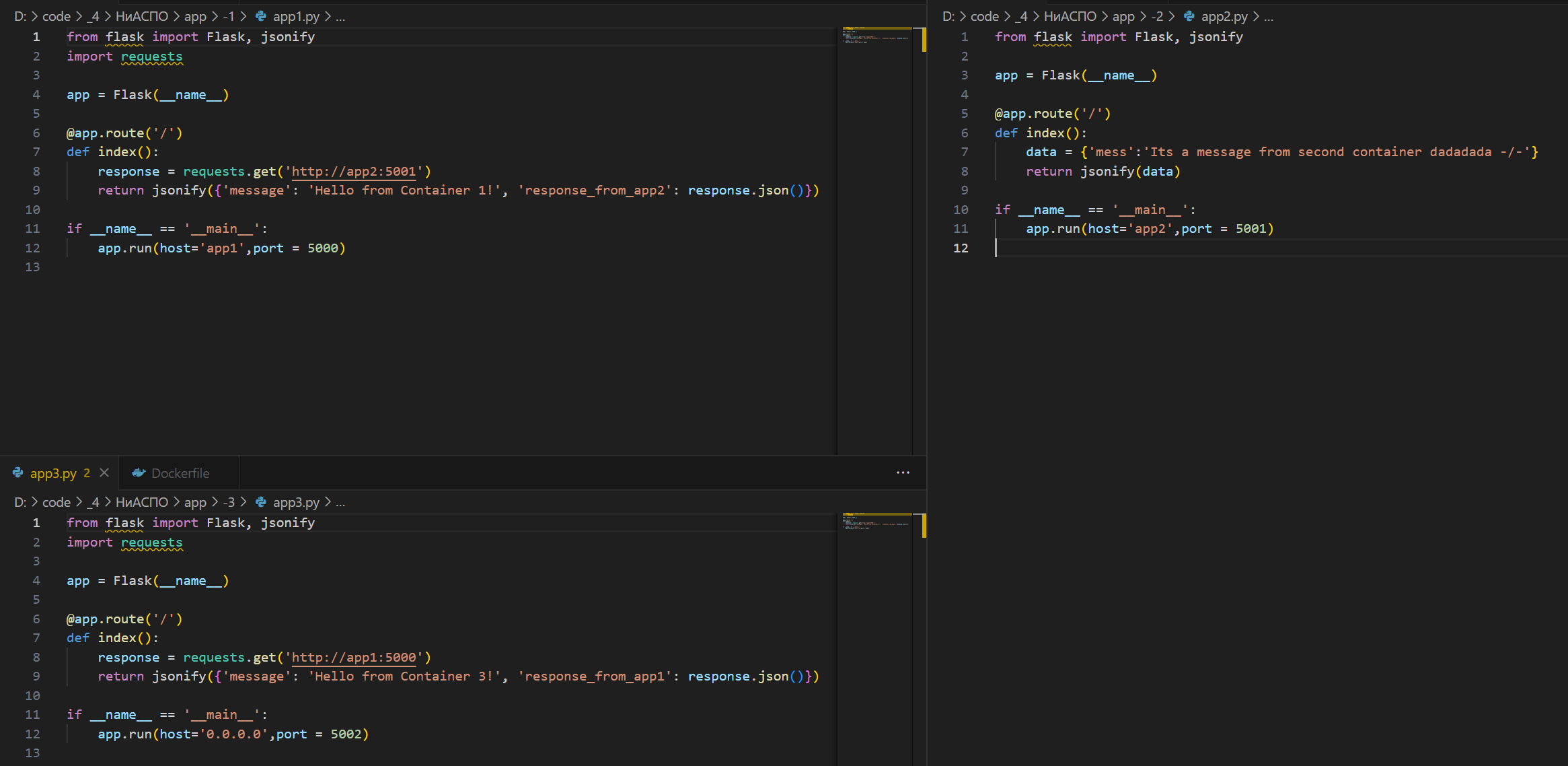


Рисунок 10.10 – Код 3 контейнеров на Flask

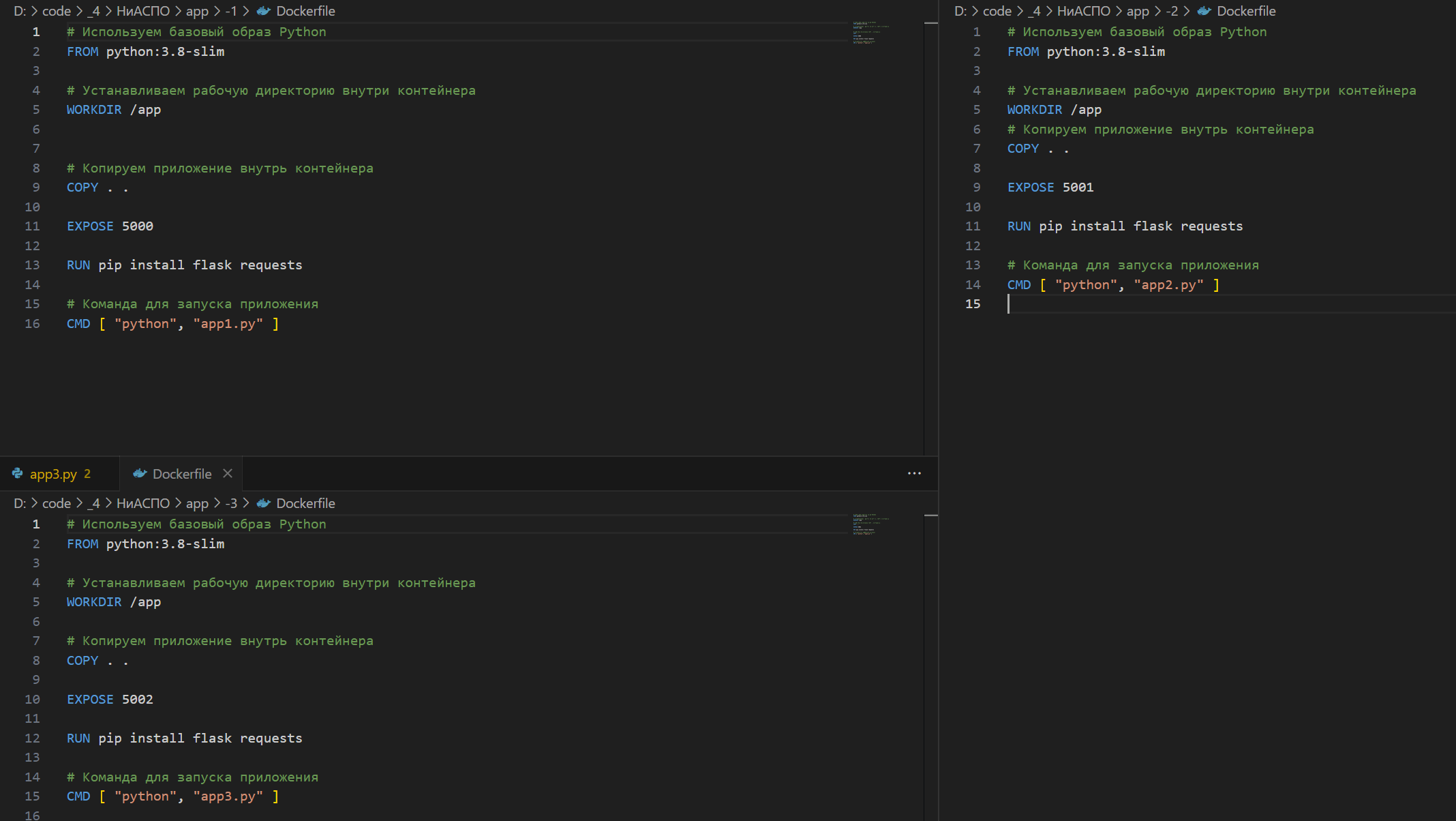


Рисунок 10.11 – Dockerfile для 3 контейнеров

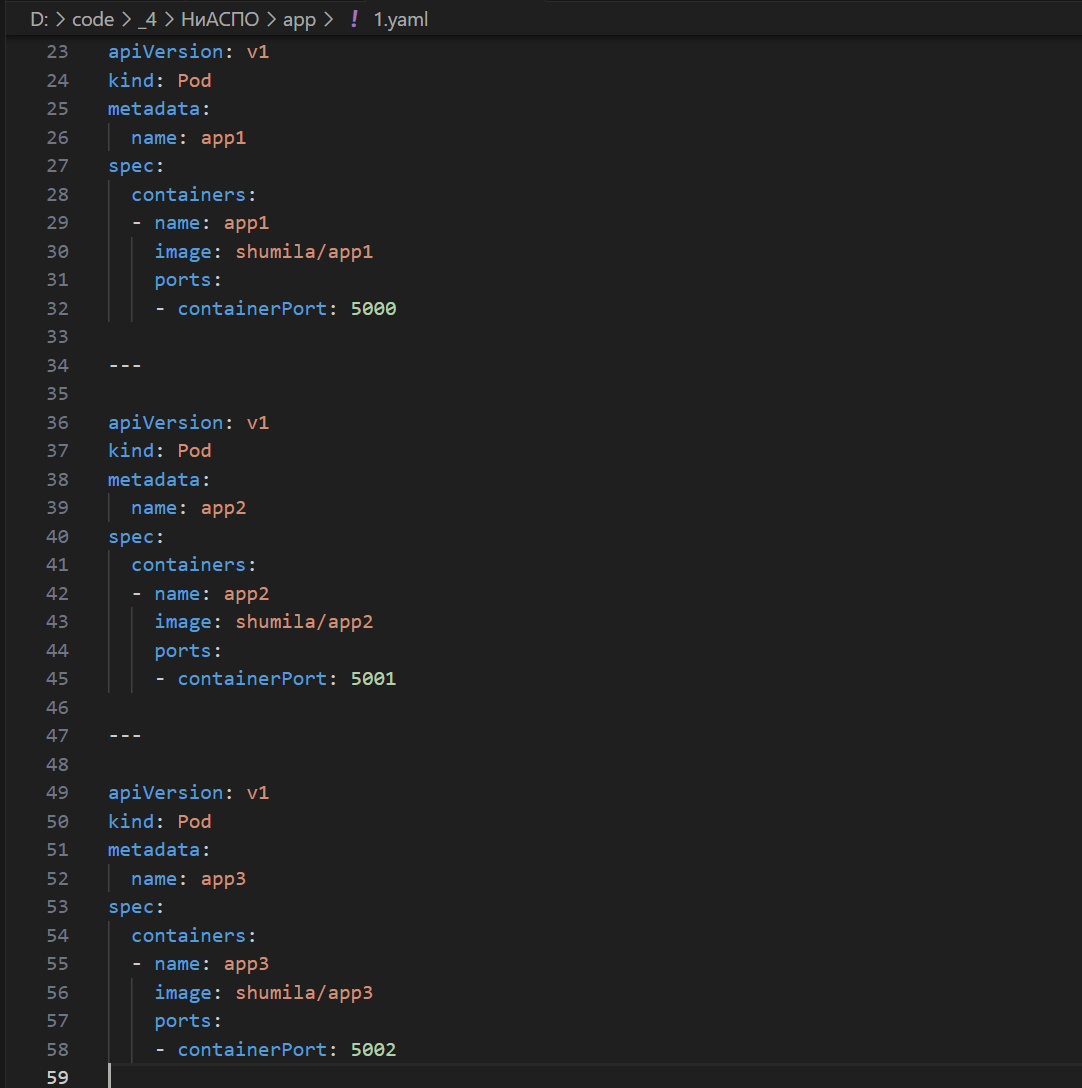


Рисунок 10.12 – Файл YAML для создания подов

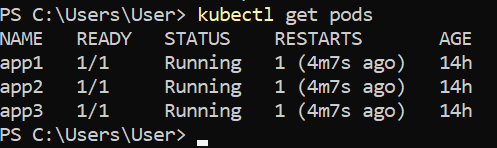


Рисунок 10.13 – Проверка работы созданных подов

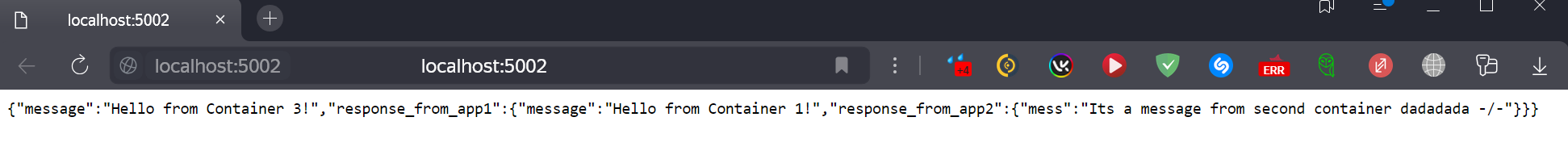


Рисунок 10.14 – Результат работы подов

**Вывод**

В результате работы по настройке кластера Kubernetes и оркестрации Docker контейнеров получены основные навыки управления распределенной инфраструктурой. Развертывание кластера и запуск контейнеров в его составе требует понимания принципов микросервисной архитектуры и инструментов для автоматизации их управления. Полученные навыки и опыт в работе с Kubernetes представляют ценность для дальнейшего развития в области DevOps и облачных технологий.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Управление Docker контейнерами с помощью Docker Compose / [Электронный ресурс] // IT is good: [сайт]. – URL: https://itisgood.ru/2019/10/10/upravlenie-docker-kontejnerami-s-pomoshhju-docker-compose/ (дата обращения: 12.03.2024)
2. Использование Docker Compose / [Электронный ресурс] // Microsoft learn: [сайт]. – URL: https://docs.microsoft.com/ruru/visualstudio/docker/tutorials/use-docker-compose (дата обращения: 05.03.2024)
3. Документация по Docker-composer / [Электронный ресурс] // composer - Official Image: [сайт]. – URL: https://hub.docker.com/\_/composer (дата обращения: 14.03.2024)
4. Официальная документация Dockerfile reference / [Электронный ресурс] // Dockerfile reference: [сайт]. – URL: https://docs.docker.com/engine/reference/builder/ (дата обращения: 11.03.2024)
5. Docker on Render (англ.) / [Электронный ресурс] // Render Docs: [сайт]. – URL: https://render.com/docs/docker (дата обращения: 15.03.2024).
6. Официальная документация Kubernetes / [Электронный ресурс] // Kubernetes: [сайт]. – URL: https://kubernetes.io/docs/home// (дата обращения: 21.03.2024)
7. Основы работы с Kubernetes / [Электронный ресурс] // 8HOST.COM: [сайт]. – URL: https://www.8host.com/blog/osnovy-raboty-s-kubernetes/ (дата обращения: 23.03.2024)
8. Основы Kubernetes: введение в технологию и базовая архитектура / [Электронный ресурс] // VK Cloud: [сайт]. – URL: https://mcs.mail.ru/blog/osnovy-kubernetes-vvedenie-v-tehnologiju (дата обращения: 23.03.2024)