МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт математики и информационных систем

Факультет автоматики и вычислительной техники

Кафедра систем автоматизации управления

**Виртуализация и контейнеризация**

Отчет по лабораторной работе №3  
по дисциплине

«Управление IT-проектами»

Выполнил студент гр. ИТб-4302-02-00 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Вершинин П.А./

(Подпись)

Руководитель ст. преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Родионов К.В./

(Подпись)

Работа защищена с оценкой «\_\_\_\_\_\_\_\_\_\_\_» «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2023 г.

Цель: получить практические навыки по использованию «Docker» для упрощения процесса разработки, тестирования и развертывания приложений с помощью контейнеризации.

В ходе работы необходимо выполнить следующее задание:

* создать тестовое nodejs приложение;
* выполнить сборку образа приложения;
* запустить приложение в Docker-контейнере;
* описать запуск приложения через «docker-compose»;
* развернуть приложение из git-репозитория или «docker hub».

Создадим новый проект, вызвав команду «npm init –y», создадим папку «src», а в ней файл «app.js» и поместим в него следующий код:

|  |
| --- |
| const http = require('http');  const port = 3000;  const server = http.createServer((req, res) => {  res.statusCode = 200;  res.setHeader('Content-Type', 'text/plain');  res.end('Hello World');  });  server.listen(port, () => {  console.log(`Server running at [http://localhost:${port}/`](http://localhost:$%7bport%7d/%60));  }); |

Теперь упакуем это приложение в контейнер, чтобы запустить через «Docker». Для этого нужно поместить следующей код в файл «Dockerfile»:

|  |
| --- |
| FROM node:alpine  WORKDIR /app  COPY package.json ./  RUN npm i  COPY src ./src  EXPOSE 3000  CMD ["node", "src/app.js"] |

После описания файла «Dockerfile» выполним команду «docker build» (рис. 1).

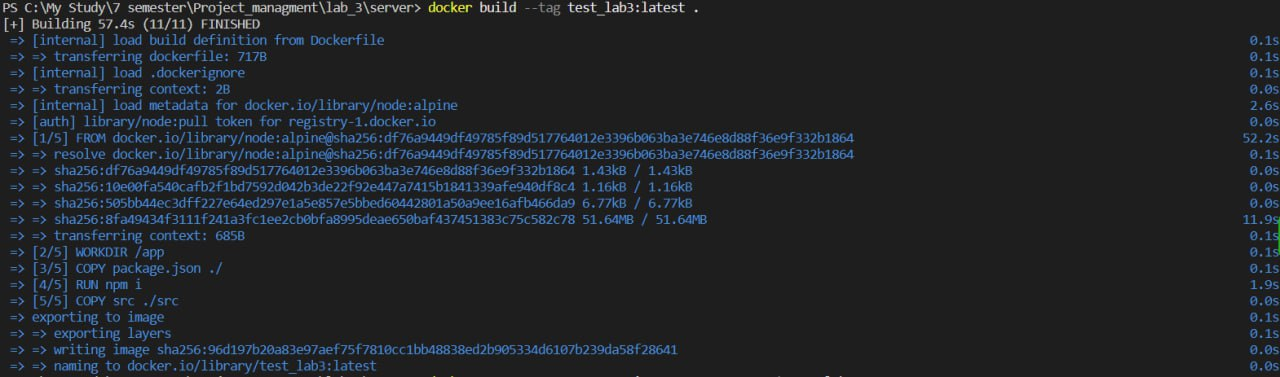


Рисунок 1 – Команда для сборки образа

После завершения сборки образа выполним следующую команду для запуска контейнера «docker run». Для остановки запущенного контейнера используем команду «docker stop [HASH]». Удаление контейнера – «docker rm [HASH]».

Выполнение команд представлено на рисунке 2.

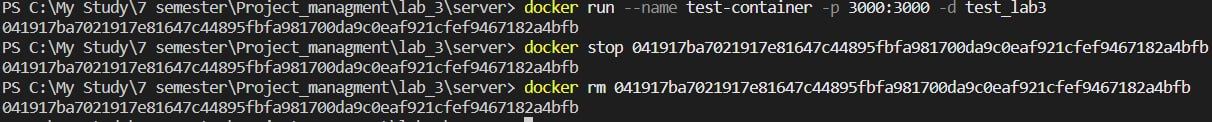


Рисунок 2 – Запуск, остановка, удаление контейнера

В результате запуска получили хэш контейнера, поскольку указали флаг «-d», при котором запуск контейнера осуществляется в фоновом режиме.

Для описания запуска приложения через «docker-compose» создадим проект, в котором следующие файлы и папки:

* файл «docker-compose.yml», который будет содержать инструкции, необходимые для запуска и настройки сервисов;
* папка «server», содержащая файлы, необходимые для обеспечения работы сервера;
* папка «client», содержащая файлы клиентского приложения.

Код в файле Python-сервера следующий:

|  |
| --- |
| import http.server  import socketserver  handler = http.server.SimpleHTTPRequestHandler  # Порт 1234  with socketserver.TCPServer(("", 1234), handler) as httpd:  # Постоянно слушает    httpd.serve\_forever() |

Данный код позволяет создать простой веб-сервер, который будет отдавать клиентам файл «index.html», содержимое которого позже будет выводиться на веб-странице.

В файле «index.html» запишем следующий текст, который будет передаваться клиенту: «Docker-Compose is magic!».

В директории сервера создадим «Dockerfile», в котором опишем инструкции для выполнения сервера:

|  |
| --- |
| FROM python:latest  ADD server.py /server/  ADD index.html /server/  WORKDIR /server/ |

Код в файле директории клиента:

|  |
| --- |
| import urllib.request  fp = urllib.request.urlopen("http://localhost:1234/")  encodedContent = fp.read()  decodedContent = encodedContent.decode("utf8")  print(decodedContent)  # Закрываем соединение с сервером.  fp.close() |

Благодаря данному коду клиентское приложение может загрузить данные с сервера и вывести их на экран.

Создадим для клиента «Dockerfile», ответственный за формирование среды, в которой будет работать клиентское Python-приложение:

|  |
| --- |
| FROM python:latest  ADD client.py /client/  WORKDIR /client/ |

Код в файле «docker-compose.yml», расположенного в корневой папке проекта:

|  |
| --- |
| version: "3"  services:   server:     build: server/     command: python ./server.py     ports:       - 1234:1234   client:     build: client/     command: python ./client.py     # используется для описания типа сети.     network\_mode: host     # позволяет указать, должен ли сервис, прежде чем  # запуститься, ждать, когда будут готовы к работе другие сервисы     depends\_on:       - server |

Для сборки проекта используем команду «docker-compose build» (рис. 3). После успешной сборки проекта запускаем контейнер командой «docker-compose up» (рис. 4).



Рисунок 3 – Сборка командой «docker-compose build»

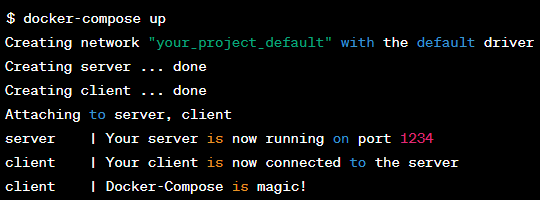


Рисунок 4 – Запуск контейнера командой «docker-compose up»

После выполнения команд клиент загружает текст с сервера: «Docker-Compose is magic!».

В результате выполнения лабораторной работы были приобретены практические навыки по использованию «Docker» для упрощения процессов разработки, тестирования и развертывания приложений с помощью контейнеризации.