**РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное**

**образовательное учреждение высшего образования**

**«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций**

**«Основы работы с Dockerfile»**

**Отчет по лабораторной работе**

**по дисциплине «Анализ данных»**

|  |
| --- |
| Выполнил студент группы ИВТ-б-о-21-1 |
| Богдашов Артём Владимирович |
| Подпись студента\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Работа защищена « » \_\_\_\_\_\_\_\_\_\_\_\_20\_\_г. |
| Проверил Воронкин Р.А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |
|  |

Ставрополь 2023

**Цель работы:** овладеть навыками создания и управления контейнерами Docker для разработки, доставки и запуска приложений. Понимание процесса создания Dockerfile, сборки и развертывания контейнеров Docker, а также оптимизации их производительности и безопасности.

**Ход работы:**

**Создание простого web-приложения на Python с использованием Dockerfile.**

Создайте проект веб-приложения на Python, включая код приложения и необходимые файлы.

Настраиваем виртуальное окружение:

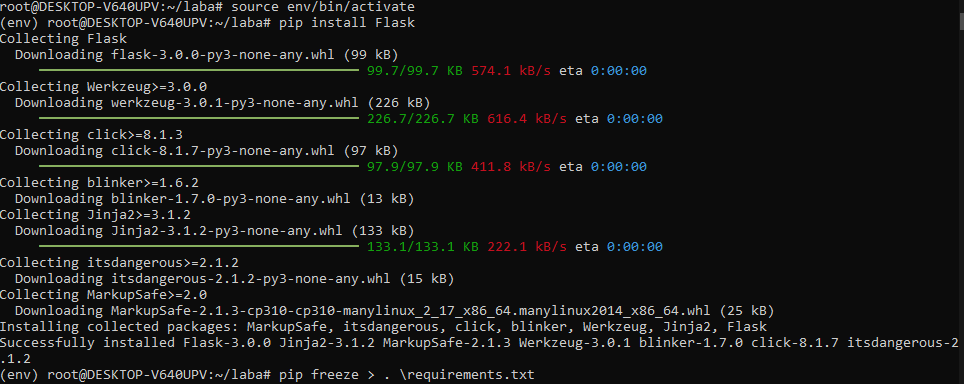


Рисунок 1. Настройка виртуального окружения

Код программы на Python (файл app.py):

# app.py

from flask import Flask, request

app = Flask(\_\_name\_\_)

@app.route('/')

def hello():

user\_name = request.args.get('name', 'Guest')

return f'Hello, {user\_name}!'

if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='0.0.0.0', port=5000)

Код html:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>FlaskApp</title>

</head>

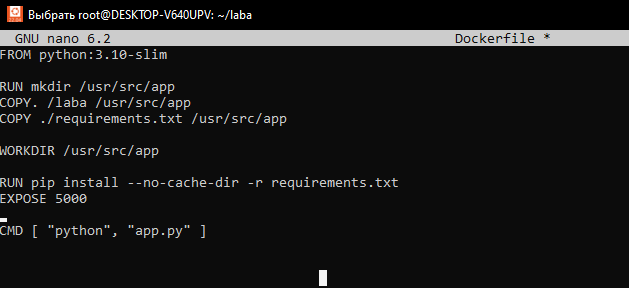
<body>

<h1>{{ message }}</h1>

</body>

</html>

Создайте Dockerfile для сборки образа Docker вашего приложения. Определите инструкции для сборки образа, включая копирование файлов, установку зависимостей и настройку команд запуска.



Результат работы программы:

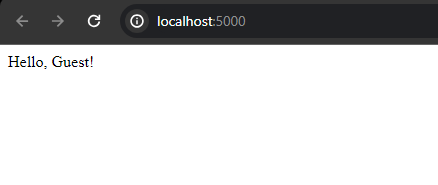


Рисунок 2. Результат работы программы в браузере

**Установка дополнительных пакетов в образ Docker.**

Создайте многоэтапной Dockerfile, состоящий из двух этапов: этап сборки и этап выполнения. На этапе сборки установите дополнительный пакет, такой как библиотеку NumPy, используя команду RUN. На этапе выполнения скопируйте созданное приложение из этапа сборки и укажите команду запуска.

Код Dockerfile:

FROM python:3.10-slim

WORKDIR /usr/src/app

COPY myapp /usr/src/app

COPY ./requirements.txt /usr/src/app

RUN pip install --no-cache-dir -r requirements.txt

EXPOSE 5000

CMD ["python", "app.py"]

Сборка и запуск контейнера:



Рисунок 3. Сборка контейнера

**Настройка переменных окружения среды в образе Docker.**

Изменим Dockerfile:

# Stage 1: Build Stage

FROM python:3.10-slim AS builder

WORKDIR /usr/src/app

COPY myapp /usr/src/app

COPY requirements.txt /usr/src/app

# Install dependencies

RUN pip install --no-cache-dir -r requirements.txt

# Stage 2: Production Stage

FROM python:3.10-slim

WORKDIR /usr/src/app

# Copy only necessary files from the builder stage

COPY --from=builder /usr/src/app /usr/src/app

EXPOSE 5000

CMD ["python", "app.py"]

Сборка контейнера:

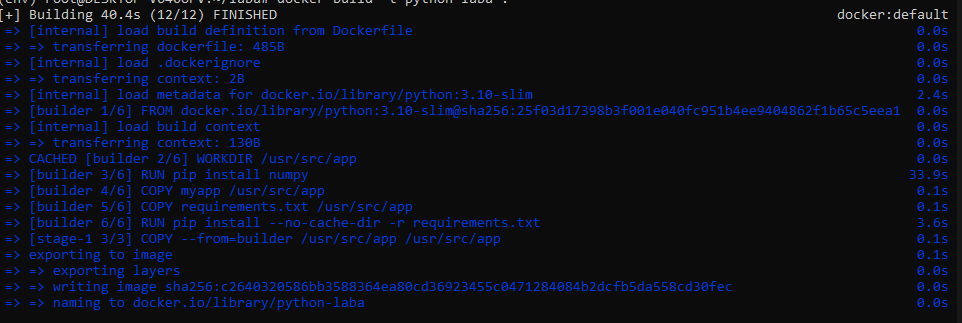


Рисунок 4. Сборка контейнера

**Контрольные вопросы:**

**1. Что такое Dockerfile?**

Dockerfile — это текстовый файл, который содержит инструкции для автоматизированного создания образа Docker. Dockerfile определяет, какие операции и конфигурации должны быть выполнены внутри контейнера при его создании.

**2. Какие основные команды используются в Dockerfile?**

FROM - Указывает базовый образ, на основе которого будет создан новый образ.

RUN - Выполняет команды в процессе создания образа.

CMD - Указывает команду, которая будет выполняться при запуске контейнера из образа.

COPY - Копирует файлы из хоста в образ.

EXPOSE - Указывает порты, которые будут открыты в контейнере.

**3. Для чего используется команда FROM?**

FROM: Эта строка указывает базовый образ, который будет использоваться для сборки нового образа.

**4. Для чего используется команда WORKDIR?**

WORKDIR: Эта строка устанавливает рабочую директорию для контейнера.

**5. Для чего используется команда COPY?**

COPY: Эта строка копирует файлы из хоста в образ Docker.

**6. Для чего используется команда RUN?**

RUN: Эта строка выполняет команды в процессе сборки образа.

**7. Для чего используется команда CMD?**

CMD: Эта строка указывает команду, которая будет выполняться при запуске контейнера.

**8. Для чего используется команда EXPOSE?**

EXPOSE: Эта строка указывает порты, которые должны быть открыты в контейнере.

**9. Для чего используется команда ENV?**

Самый простой способ настроить переменную среды в образе Docker — это использовать команду ENV.

**10. Для чего используется команда USER?**

Команда USER в Dockerfile указывает пользователя, от имени которого будет выполняться основная команда контейнера (CMD или ENTRYPOINT).

**11. Для чего используется команда HEALTHCHECK?**

HEALTHCHECK — инструкции, которые Docker может использовать для проверки работоспособности запущенного контейнера.

**12. Для чего используется команда LABEL?**

LABEL — описывает метаданные. Например — сведения о том, кто создал и поддерживает образ.

**13. Для чего используется команда ARG?**

ARG — задаёт переменные для передачи Docker во время сборки образа.

**14. Для чего используется команда ONBUILD?**

Инструкция ONBUILD добавляет к образу инструкцию-триггер, которая будет выполнена позже, когда образ будет использоваться в качестве основы для другой сборки.

**15. Что такое многоэтапная сборка?**

Многоэтапный Dockerfile состоит из двух основных этапов:

1. Этап сборки: Этот этап отвечает за компиляцию и сборку приложения. Он использует базовый образ с необходимыми инструментами для сборки, такими как компилятор Golang и соответствующие зависимости.

2. Этап выполнения: Этот этап отвечает за запуск и выполнение приложения. Он использует более минимальный базовый образ, например, Alpine Linux, содержащий только необходимые библиотеки для выполнения приложения.

**16. Какие преимущества использования многоэтапной сборки?**

Уменьшение размера образа: при использовании многоэтапных сборок только необходимые файлы для выполнения приложения включаются в окончательный образ, что значительно уменьшает его размер. Повышение безопасности: Многоэтапные сборки уменьшают риск уязвимостей безопасности, поскольку они изолируют этапы сборки и выполнения, ограничивая доступ к ненужным инструментам и зависимостям.

**17. Какие недостатки использования многоэтапной сборки?**

Сложность конфигурации и поддержки процесса сборки может возрасти. Необходимо следить за последовательностью этапов, управлять зависимостями и обеспечивать корректное выполнение каждого этапа. Это требует дополнительных знаний и времени на настройку, особенно для больших и сложных проектов. Кроме того, многоэтапная сборка Docker требует наличия основного образа системы, который может быть достаточно большим и содержать лишние компоненты. Это может увеличить размер окончательного образа, что негативно отразится на скорости его развертывания и потреблении ресурсов.

**18. Как определить базовый образ в Dockerfile?**

FROM node:latest

**19. Как определить рабочую директорию в Dockerfile?**

WORKDIR /usr/src/app

**20. Как скопировать файлы в образ Docker?**

COPY. /my-app /usr/src/app/

**21. Как выполнить команды при сборке образа Docker?**

Команда RUN выполняет команды в процессе создания образа.

**22. Как указать команду запуска контейнера?**

Команда CMD в Dockerfile указывает команду, которая будет выполняться при запуске контейнера. Она может состоять из одной или нескольких команд, разделенных пробеламиКоманда ENTRYPOINT в Dockerfile указывает исполняемый файл, который будет использоваться в качестве основной точки входа в контейнер.

**23. Как открыть порты в контейнере?**

Запуск образа с флагом -p перенаправляет общедоступный порт на частный порт внутри контейнера.

**24. Как задать переменные среды в образе Docker?**

Самый простой способ настроить переменную среды в образе Docker –это использовать команду ENV .Вы также можете настроить переменные среды в образе Docker с помощью файла .env. Чтобы использовать файл .env для настройки переменных среды в образе Docker, вы должны добавить команду ADD .env /app/.env в Dockerfile.Вы также можете настроить переменные среды при запуске контейнера. Для этого используйте флаг --env или -e .

**25. Как изменить пользователя, от имени которого будет**

**выполняться контейнер?**

При помощи команды USER.

**26. Как добавить проверку работоспособности к контейнеру?**

При помощи команды HEALTHCHECK.

**27. Как добавить метку к контейнеру?**

При помощи команды LABEL.

**28. Как передать аргументы при сборке образа Docker?**

При помощи команды ARG.

**29. Как выполнить команду при первом запуске контейнера?**

При помощи команды ENTRYPOINT.

**30. Как определить зависимости между образами Docker?**

При помощи команды ONBUILD.

**Вывод:** в ходе данной лабораторной работы были освоены навыки создания и управления контейнерами Docker для разработки, доставки и запуска приложений. Понимание процесса создания Dockerfile, сборки и развёртывания контейнеров Docker, а также оптимизации их производительности и безопасности.