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

Уланбекова Айканыш Уланбековна

« » _____ 2023г.

Подпись студента_____

Работа защищена « »_____20__г.

Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2023

Цель занятия: 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.

Порядок выполнения работы:

Задача 1: создать простое веб-приложение на Python, которое принимает имя пользователя в качестве параметра URL и возвращает приветствие с именем пользователя. Используйте Dockerfile для сборки образа Docker вашего приложения и запустите контейнер из этого образа.

```
FROM python:3.10-slim

RUN mkdir /usr/src/app
COPY ./my-app /usr/src/app
COPY ./requirements.txt /usr/src/app

WORKDIR /usr/src/app

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

EXPOSE 5000

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

Рисунок 1. Образ веб-приложения

```
from datetime import datetime
from flask import Flask

app = Flask(__name__)

@app.route("/<name>")
def hello_world(name):
    return f"Привет, {name}"

if __name__ == "__main__":
    app.run(host="0.0.0.0")
```

Рисунок 2. Код приложения

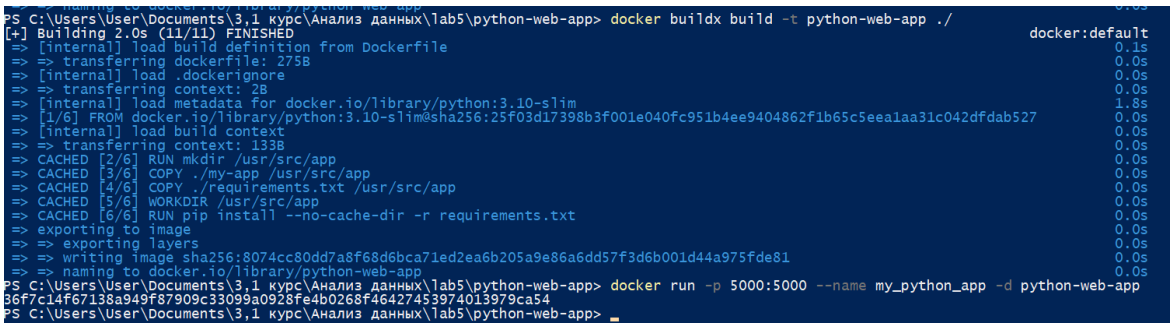

Рисунок 3. Запуск

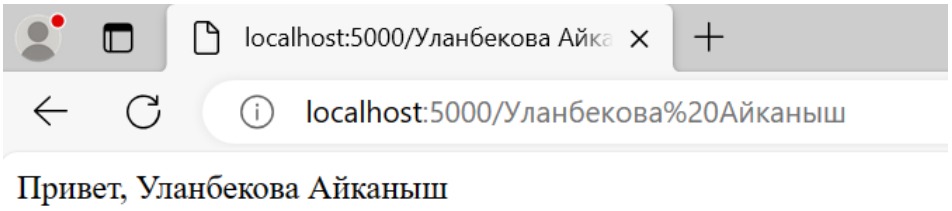

Рисунок 4. Вывод

Задача 2: установить дополнительный пакет, например библиотеку NumPy для Python, в образ Docker веб-приложения.

```
FROM python:3.10 as builder

RUN pip install numpy

FROM python:3.10-slim as runner

RUN mkdir /usr/src/app
COPY ./my-app /usr/src/app
COPY ./requirements.txt /usr/src/app

WORKDIR /usr/src/app

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

EXPOSE 5000

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

Рисунок 5. Добавление NumPy в образ

Задача 3: настроить переменную среды, например URL базы данных, в образе Docker веб-приложения. Используйте команду ENV в Dockerfile для определения переменной среды и сделайте ее доступной для приложения.

```
FROM python:3.10 as builder

RUN pip install numpy

FROM python:3.10-slim as runner

RUN mkdir /usr/src/app
COPY ./my-app /usr/src/app
COPY ./requirements.txt /usr/src/app

WORKDIR /usr/src/app

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

EXPOSE 5000

ENV DATABASE_URL=localhost:5432/fake

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

Рисунок 6. Добавили ENV

```
from datetime import datetime
from flask import Flask
import os

app = Flask(__name__)

@app.route("/<name>")
def hello_world(name):
    return f"Привет, {name}"

@app.route("/env")
def env():
    return f"{os.environ['DATABASE_URL']}"

if __name__ == "__main__":
    app.run(host="0.0.0.0")
```

Рисунок 7. Добавили вывод ENV

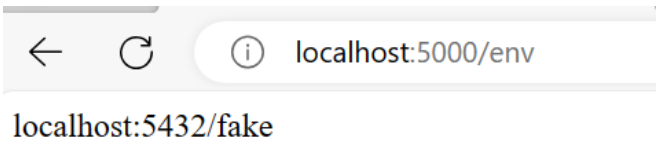

Рисунок 8. Результат

Задача 4: скопировать необходимые файлы, такие как статические файлы или конфигурационные файлы, в образ Docker веб-приложения. Используйте команду COPY в Dockerfile для определения файлов для копирования и их местоположения в образе.

```
FROM python:3.10 as builder

RUN pip install numpy

FROM python:3.10-slim as runner

RUN mkdir /usr/src/app
COPY ./my-app /usr/src/app
COPY ./requirements.txt /usr/src/app

WORKDIR /usr/src/app

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

EXPOSE 5000

ENV DATABASE_URL=localhost:5432/fake

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

Рисунок 9. Добавили инструкцию COPY

Задание 5: выполнить команды инициализации или настройки при запуске контейнера веб-приложения. Используйте команду RUN в Dockerfile для определения команд для выполнения и их параметров.

```
FROM python:3.10 as builder

RUN pip install numpy

FROM python:3.10-slim as runner

RUN mkdir /usr/src/app
COPY ./my-app /usr/src/app
COPY ./requirements.txt /usr/src/app

WORKDIR /usr/src/app

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

EXPOSE 5000

ENV DATABASE_URL=localhost:5432/fake

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

Рисунок 10. Добавили инструкцию RUN

Контрольные вопросы:

1. Что такое Dockerfile?

Dockerfile - это текстовый файл, который содержит инструкции для автоматизированного создания образа Docker.

2. Какие основные команды используются в Dockerfile?

Конструкции Dockerfile:

1. FROM: указывает базовый образ.

2. COPY и ADD: копируют файлы в образ.

3. RUN: выполняет команды внутри образа.

4. CMD: задает команду по умолчанию для контейнера.

5. ENTRYPOINT: определяет исполняемую команду при запуске контейнера.

6. EXPOSE: объявляет порт, который контейнер будет слушать.

7. ENV: устанавливает переменные среды.

8. ARG: определяет аргументы для сборки образа.

3. Для чего используется команда FROM?

Команда FROM используется в Dockerfile для указания базового образа, на основе которого будет создаваться новый образ.

4. Для чего используется команда WORKDIR?

Команда WORKDIR используется в Dockerfile для установки рабочего каталога внутри контейнера.

5. Для чего используется команда COPY?

Команда COPY в Dockerfile используется для копирования файлов и директорий из вашего хост-системы в образ контейнера.

6. Для чего используется команда RUN?

Команда RUN в Dockerfile используется для выполнения команд во время сборки образа контейнера.

7. Для чего используется команда CMD?

Команда CMD в Dockerfile используется для указания команды, которая будет выполнена при запуске контейнера на основе этого образа.

8. Для чего используется команда EXPOSE?

Команда EXPOSE используется для указания портов, которые контейнер будет слушать во время выполнения.

9. Для чего используется команда ENV?

Команда ENV используется для определения переменных среды, которые будут доступны во время выполнения контейнера.

10. Для чего используется команда USER?

Команда USER используется для указания пользователя, от имени которого будет выполняться контейнер.

11. Для чего используется команда HEALTHCHECK?

Команда HEALTHCHECK используется для добавления проверки работоспособности контейнера.

12. Для чего используется команда LABEL?

Команда LABEL используется для добавления меток к образу Docker.

13. Для чего используется команда ARG?

Команда ARG используется для передачи аргументов при сборке образа Docker.

14. Для чего используется команда ONBUILD?

Команда ONBUILD используется в Dockerfile для определения команд, которые будут выполнены при использовании вашего образа в качестве базового образа для другого образа.

15. Что такое многоэтапная сборка?

Многоэтапная сборка в Docker позволяет создавать образы, используя несколько этапов, каждый из которых может выполнять определенные задачи.

16. Какие преимущества использования многоэтапной сборки?

Преимущества многоэтапной сборки включают уменьшение размера образа, улучшение безопасности и упрощение процесса сборки.

17. Какие недостатки использования многоэтапной сборки?

Недостатки многоэтапной сборки могут включать сложность настройки и поддержки для более сложных сценариев сборки.

18. Как определить базовый образ в Dockerfile?

Базовый образ определяется с помощью команды FROM в Dockerfile.

19. Как определить рабочую директорию в Dockerfile?

Рабочая директория определяется с помощью команды WORKDIR в Dockerfile.

20. Как скопировать файлы в образ Docker?

Файлы копируются в образ Docker с помощью команды COPY или ADD в Dockerfile.

21. Как выполнить команды при сборке образа Docker?

Команды выполняются при сборке образа Docker с помощью команды RUN в Dockerfile.

22. Как указать команду запуска контейнера?

Команда запуска контейнера указывается с помощью команды CMD или ENTRYPOINT в Dockerfile.

23. Как открыть порты в контейнере?

Порты открываются в контейнере с помощью команды EXPOSE в Dockerfile и опций при запуске контейнера.

24. Как задать переменные среды в образе Docker?

Переменные среды задаются в образе Docker с помощью команды ENV в Dockerfile.

25. Как изменить пользователя, от имени которого будет выполняться контейнер?

Пользователь изменяется с помощью команды USER в Dockerfile.

26. Как добавить проверку работоспособности к контейнеру?

Проверка работоспособности добавляется к контейнеру с помощью команды HEALTHCHECK в Dockerfile.

27. Как добавить метку к контейнеру?

Метка добавляется к контейнеру с помощью команды LABEL в Dockerfile.

28. Как передать аргументы при сборке образа Docker?

Аргументы передаются при сборке образа Docker с помощью команды ARG в Dockerfile.

29. Как выполнить команду при первом запуске контейнера?

Команда выполняется при первом запуске контейнера с помощью команды ONBUILD в Dockerfile.

30. Как определить зависимости между образами Docker?

Зависимости между образами Docker определяются через инструкции в Dockerfile, такие как FROM для базового образа и COPY для копирования файлов из других образов.