Лабораторная работа 2

Цель работы

Создать два Dockerfile — один с плохими практиками, другой с их исправлением. Провести анализ ошибок, описать, почему те или иные подходы считаются плохими, и как их исправить. Рассмотреть также плохие практики использования контейнера.

Telegram Bot

В качестве примера приложения был использован телеграм-бот, разработанный на Python с использованием библиотеки Telebot. Бот взаимодействует с пользователями через Telegram API с использованием токена.

Код бота также используется для демонстрации Dockerfile.

Код Telegram-бота для загрузки данных на сервер

```
import os
import zipfile
from telebot import TeleBot
from telebot.types import Message

# Задайте токен вашего Telegram-бота
TOKEN = os.getenv("TOKEN")
STORAGE_PATH = "storage"

bot = TeleBot(TOKEN)

# Убедимся, что директория для загрузок существует
os.makedirs(STORAGE_PATH, exist_ok=True)
```

```
@bot.message_handler(commands=["start"])
def send_welcome(message: Message):
    bot.reply_to(message, "Привет! Отправь мне файлы, и я сохраню их на сервере
в формате ZIP.")

@bot.message_handler(content_types=["document"])
def handle_document(message: Message):
    try:
        file_info = bot.get_file(message.document.file_id)
        downloaded_file = bot.download_file(file_info.file_path)

        # Сохраняем файл на сервер
        file_path = os.path.join(STORAGE_PATH, message.document.file_name)
        with open(file_path, "wb") as f:
            f.write(downloaded_file)

        # Архивируем в ZIP
        zip_path = os.path.join(STORAGE_PATH, "files.zip")
        with zipfile.ZipFile(zip_path, "a") as zipf:
            zipf.write(file_path, arcname=message.document.file_name)

        bot.reply_to(message, f"Файл {message.document.file_name} успешно
сохранен и добавлен в ZIP!")
    except Exception as e:
        bot.reply_to(message, f"Ошибка при обработке файла: {e}")

bot.polling()
```

Команды для создания и запуска контейнера

1. **Создание образа:**

```
docker build -t badbot .
```

2. **Запуск контейнера:**

```
docker run --name badtgbot -e TOKEN="ваш_токен" -d badbot
```

Плохой Dockerfile (BAD DOCKERFILE)

```
FROM python:latest
ADD bot.py /bot.py
ENV TOKEN="123456:ABCDEF"
RUN pip install pyTelegramBotAPI
ENTRYPOINT python bot.py
```

Хороший Dockerfile (GOOD DOCKERFILE)

```
FROM python:3.12
WORKDIR /app
COPY bot.py /app/bot.py
RUN pip install --no-cache-dir pyTelegramBotAPI
ENV TOKEN="000"
ENTRYPOINT ["python", "bot.py"]
```

Анализ и исправления

1. **FROM python:latest → FROM python:3.12**
   Использование `latest` приводит к неопределенности окружения. Лучше указывать конкретную версию.

2. **ADD → COPY**
   `ADD` имеет дополнительные функции, которые могут быть избыточными или небезопасными. Для копирования файлов достаточно использовать `COPY`.

3. **ENV TOKEN="123456:ABCDEF" → ENV TOKEN="000"**
   Никогда не включайте реальные данные (токены, пароли) в Dockerfile. Используйте заглушки или передавайте значения через переменные окружения при запуске.

4. **ENTRYPOINT python bot.py → ENTRYPOINT ["python", "bot.py"]**
   Формат JSON-массива предпочтителен, так как обеспечивает точную передачу аргументов без интерпретации оболочки.

Вывод

Работа над этой лабораторной помогла изучить ключевые принципы работы с Docker, а также понять, как создавать надежные и безопасные Docker-образы. Исправление ошибок позволило обеспечить стабильность, предсказуемость и безопасность контейнеров.