Департамент образования и науки города Москвы

Государственное автономное образовательное учреждение высшего образования города Москвы

«Московский городской педагогический университет»

Институт цифрового образования

Департамент информатики, управления и технологий

ДИСЦИПЛИНА:

Интеграция и развертывание программного обеспечения с помощью контейнеров

**Лабораторная работа 2.1**

**Создание Dockerfile и сборка образа**

Выполнила: Сергеева А. И., группа: АДЭУ-211

Преподаватель: Босенко Т.М.

Москва

2025

**Цель работы:** научиться создавать Dockerfile и собирать образы Docker для приложений.

**Задачи:**

1. Создать Dockerfile для указанного приложения.
2. Собрать образ Docker с использованием созданного Dockerfile.
3. Запустить контейнер из собранного образа и проверить его работоспособность.
4. Выполнить индивидуальное задание.

**Ход работы:**

**Вариант 12.** Создайте Dockerfile для приложения на Node.js, которое подключается к базе данных MongoDB и выводит список коллекций.

Для выполнения индивидуального задания для начала было необходимо перейти в каталог для проекта, как видно на рисунке 1.

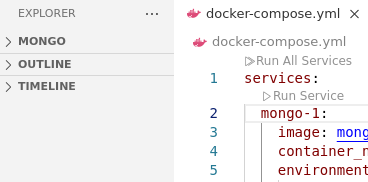


Рисунок 1- Переход в папку проекта по MongoDB в VS code

Далее было необходимо создать файл приложения, который называется index.js на рисунках 2-5. В начале кода импортируются необходимые модули, а именно express (фреймворк для Node.js, который позволяет легко создавать веб-приложения и API) и mongoose (библиотека для работы с MongoDB, которая упрощает взаимодействие с базой данных, позволяя использовать схемы и модели). Далее создается экземпляр приложения, устанавливается порт, на котором приложение слушает запросы, по умолчанию 28203. Настраивается подключение к MongoDB, которая берется из переменной окружения MONGO\_URI от уже существующего docker-compose.yml. app.get('/collections', ...) выводит коллекции, при успешном выполнении выводится JSON-ответ с массивом коллекций.

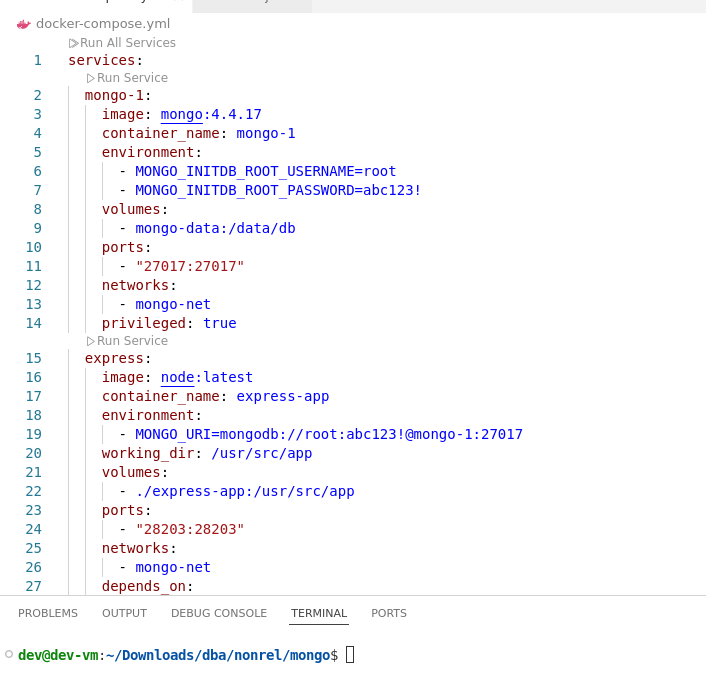


Рисунок 2 – docker-compose.yml для подключения к MongoDB

Изначально была 7 версия, которая не поддерживалась из-за слабого процессора и проблем с AVX, что было видно в логах контейнера mongo-1. Далее для работы с ним была выбрана 4 версия, которая не нуждается в AVX, образ был пересобран на основе новой версии и стал корректно работать.

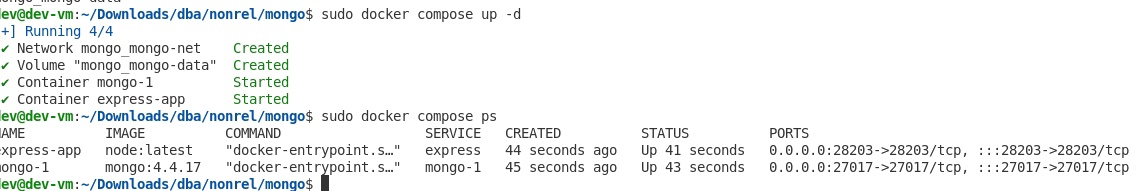


Рисунок 3 – Успешный запуск контейнеров

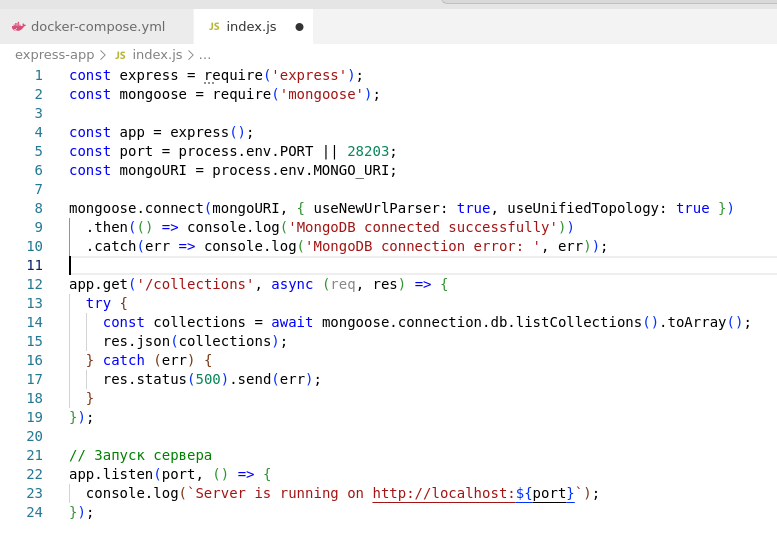


Рисунок 4 – Настройка app.js



Рисунок 5 –Файл с зависимостями

Далее был создан Dockerfile, чтобы подключиться к MongoDB и вывести коллекции, он был добавлен в директорию, как и другие файлы на рисунке 6.

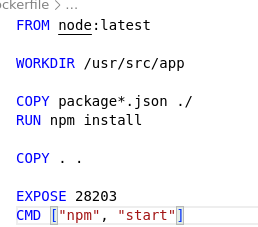


Рисунок 6 – Dockerfile для запуска MongoDB и вывода коллекций

Далее запускаем контейнер на рисунках 7-8 и проверяем их работу.

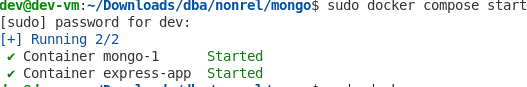


Рисунок 7 – Запуск контейнера

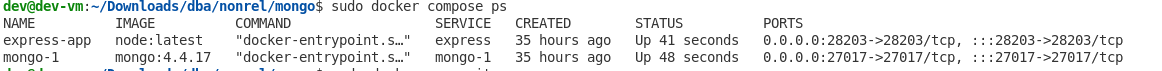


Рисунок 8 – Проверка запущенных контейнеров

После данных шагов был выполнен переход mongo shell, чтобы была возможность создать тестовую базу и коллекции для проверки созданного Dockerfile на рисунках 9-11, тестовая база называется testDB, в нее была добавлена 1 коллекция с 1 документом. Подключение выполнялось с указанием пользователя и паролем.

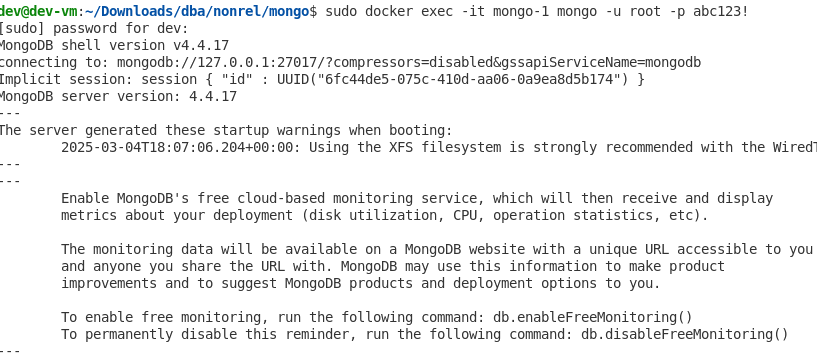


Рисунок 9 – Переход в mongo shell



Рисунок 10 – Создание базы данных

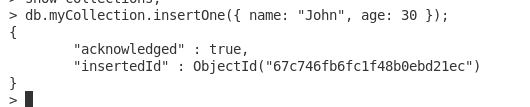


Рисунок 11 – Создание коллекции

Далее необходимо внести небольшие изменения в код, чтобы задать подключение к конкретной базе данных на рисунке 12, а именно изменить ссылку с указанием конкретной базы и в подключении указать нужную базу данных.

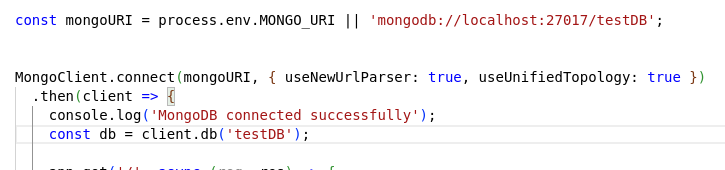


Рисунок 12 – Изменение подключения к базе

Затем была проведена проверка в браузере и действительно выводится нужная коллекция на рисунке 13.

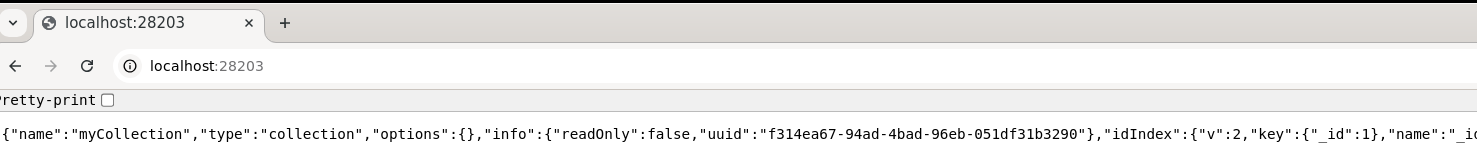


Рисунок 13 – Вывод коллекций

**Общий вывод**:

В ходе лабораторной работы был создан Dockerfile, позволяющий смотреть коллекции в MongoDB. Были запущены 2 контейнера: по нашему приложению и по самому mongo. В процессе выполнения были изучены особенности современных и старых версий mongo, от 5 версии необходимо, чтобы процессор поддерживал AVX. Для обхождения данного условия достаточно установить старую версию, в ходе работы была установлена 4 версия. Была создана тестовая база, коллекция и документ в ней, чтобы в дальнейшем проверить работоспособность кода, а также настроить подключение к базе в коде и указать ссылку, по которой будет возможность увидеть коллекции. Кроме Dockerfile необходимы еще были файлы node.js и файл с указанными зависимостями. Также были изучены команды в mongo shell. По результатам работы поставленные цель и задачи были выполнены.

**Ответы на контрольные вопросы:**

**1. Что такое Dockerfile и для чего он используется?**

Dockerfile — это текстовый файл, содержащий набор инструкций для автоматической сборки образа Docker. Он описывает, как установить программное обеспечение, скопировать файлы, задать переменные среды и настроить окружение для вашего приложения. Dockerfile упрощает процесс создания образов, позволяя автоматизировать и документировать сборку приложений.

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

FROM: указывает базовый образ, на основе которого будет происходить сборка.

RUN: выполняет команды в процессе сборки, например, установку пакетов или копирование файлов.

COPY: копирует файлы из локальной файловой системы в образ (например, код приложения).

ADD: помимо копирования файлов, может также разархивировать архивы.

CMD: задает команду по умолчанию, которая будет выполняться при запуске контейнера.

ENTRYPOINT: задает основную команду, которая будет выполняться при запуске контейнера, позволяя передавать аргументы.

EXPOSE: указывает порты, которые контейнер будет прослушивать.

ENV: задает переменные окружения для контейнера.

**3. Как выполняется сборка образа Docker с использованием Dockerfile?**

docker build -t имя\_образа .

**4. Как запустить контейнер из собранного образа?**

docker run -d --name имя\_контейнера -p 5000:5000 имя\_образа

**5. Каковы преимущества использования Dockerfile для создания образов Docker?**

Автоматизация: Позволяет автоматизировать процесс сборки образов, избегая ручных ошибок.

Версионирование и повторяемость: Образы можно создавать с использованием версионного контроля, что упрощает повторное создание и развертывание.

Документирование: Dockerfile служит документом, объясняющим, как именно был собран образ, что упрощает его поддержку и понимание.

Портируемость: Образы, созданные с помощью Dockerfile, можно легко переносить между различными средами и платформами.

Модульность: Dockerfile позволяет создавать образы исходя из модульных компонентов, что упрощает управление зависимостями и конфигурациями.