Министерство науки и высшего образования Российской Федерации

федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский экономический университет имени Г.В. Плеханова»

**МОСКОВСКИЙ ПРИБОРОСТРОИТЕЛЬНЫЙ ТЕХНИКУМ**

специальность 09.02.07 «Информационные системы и программирование»

Квалификация: Программист

ПРАКТИЧЕСКИЕ РАБОТЫ

ПО МДК 04.02 «Обеспечение качества функционирования компьютерных систем»

|  |  |
| --- | --- |
| Выполнил студент  группы П50-3-22  Полежайковский Алексей Дмитриевич | Проверил преподаватель  \_\_\_\_\_\_\_\_\_\_\_ К.С. Образцова  «\_\_\_» \_\_\_\_\_\_\_\_\_ 2024 года |

Москва 2024

**Содержание**

[Практическая работа №1 3](#_Toc209705080)

[Практическая работа №2 9](#_Toc209705081)

[Практическая работы №3 13](#_Toc209705082)

[Практическая работа №4 18](#_Toc209705083)

# Практическая работа №1

Цель работы: изучить создание docker образов, их сборку и запуск на примере написания Dockerfile для готовых проектов.

Ход работы:

1. Калькулятор java

Для данного проекта используем образ openjdk:alpine. Создаем Dockerfile и вписываем команды для запуска контейнера.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Dockerfile для проекта калькулятора на java

Далее необходимо собрать образ.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Сборка образа

Далее необходимо запустить контейнер с образом. В данном случае запускаем в интерактивном режиме с помощью ключей -it.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Запуска calc-java

1. Калькулятор python

Для нашего образа используем образ python:alpine для сборки.

Изображение выглядит как текст, Шрифт, снимок экрана, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Dockerfile для calc-py

Используем команды для сборки и запуска как для calc-java.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Сборка образа calc-py

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Запуск calc-py

1. Калькулятор dart

Для калькулятора на dart используем dart:stable.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 7 - Dockerfile для dart-calc

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Сборка образа dart-calc

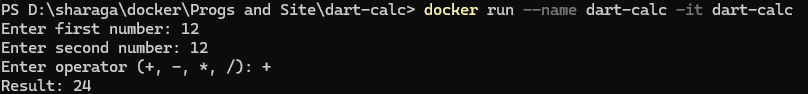


Рисунок - Запуск dart-calc

1. Игра на java

Для game-java повторяем dockerfile из пункта 1.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 10 - Dockerfile для game-java

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Сборка образа game-java

Изображение выглядит как текст, снимок экрана, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Запуск game-java

1. Игра камень ножницы бумага на python

Для игры на питоне повторяем dockerfile из пункта 2.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Dockerfile rpcgame

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Сборка образа rpc

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Запуск rpc

1. Веб сайт

Для вебсайта используем nginx:alpine. В Dockerile для вебсайта вместо обычного расположения копируем файлы в директорию хранения html в nginx. Далее с помощью команды EXPOSE указываем 80 порт (необязательно).

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Dockerfile nginx

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Сборка образа сайта

Запускаем контейнер как обычно, но добавляем ключ -d для запуска в фоновом режиме и добавляем ключ -p для прокидывания порта на хост машину. Слева указываем порт н хосте, слева порт контейнера.



Рисунок - Запуск сайта

Изображение выглядит как амфибия, шляпа, лягушка, Настоящая лягушка

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Страница сайта в браузере

Вывод: в результате практической работы были изучены базовые принципы работы с docker и написаны Dockerfile’ы для проектов на разных языках.

# Практическая работа №2

Цель работы: изучить принцип работы docker compose на примере проектов на node.js и Django.

Ход работы:

1. Проект на node.js

В качестве проекта на node.js представлен проект по сохранению времени в базу данных. Компоненты: API, frontend, MySQL.

Пишем dockerfile для создания образов api и frontend.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Dockerfile фронтенда

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Dockerfile api

Далее создаем файл compose.yml в котором будет конфигурация для запуска всех сервисов.

Изображение выглядит как текст, дисплей, снимок экрана, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Compose.yml

Далее запускаем и заходим на веб панель фронтенда.

Изображение выглядит как текст, снимок экрана, дизайн

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Фронтенд.

1. Проект на Django

Второй проект использует Django, postgresql и adminer для администрирования БД.

Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Dockerfile Django

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - compose.yml

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Веб панель Django

Вывод: в результате практической работы были изучены принципы работы с docker compose, а также с помощью него развернуты 2 сервиса, включающие в себя приложения и сервера БД.

# Практическая работы №3

Цель работы: изучить принципы работы с файловой системой docker контейнеров, изучить volumes и bind mounts, применить знания на практике.

Ход работы:

1. Создание образа

Создаем Dockerfile для образа на основе nginx.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Dockerfile

1. Запуск контейнера с volume

Запускаем контейнер с пробрасыванием портов и указываем volume для доступа к файлам.



Рисунок - Запуск контейнера

При запуске переходим на /index.html.

Изображение выглядит как птица, облако, небо, крыло

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Первый запуск

1. Смена файлов в контейнере

Для проверки смены файла меняем удаляем старый файл картинки и вставляем новый через docker cp.

Изображение выглядит как текст, снимок экрана, Шрифт, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок – Удаление



Рисунок - Копирование нового файла

Обновляем на странице кэш и смотрим на результат.

Изображение выглядит как зарисовка, рисунок, мультфильм, черно-белый

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Новая картинка

Картинка успешно обновилась.

1. Запуск контейнера с bind mount

Используем ту же команду что и для запуска с volume но вместо volume указываем нужную нам директорию файловой системы хоста.



Рисунок - Запуск с bind mount

При переходе на index.html видим первую картинку.

Изображение выглядит как облако, текст, плакат, черно-белый

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Первая картинка

Для смены картинки на сайте нужно будет просто поменять ссылку на картинку в файле в привязанной директории.



Рисунок - Ссылка на картинку

После смены ссылки мы увидим новую картинку.

Изображение выглядит как зарисовка, рисунок, черно-белый, мультфильм

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Новая картинка

Вывод: в результате практической были изучены принципы работы с gприменены знания на практике в работающем проекте.

# Практическая работа №4

Цель работы: изучить базовые принципы работы с xserver, развернуть графическое приложение в docker с помощью xserver.

Ход работы:

1. Запуск xeyes

Скачиваем xserver, его используют для отображения окон на unix подобных системах. Пишем Dockerfile. В нем указываем базовый образ ubuntu. После этого пишем обновление пакетов, скачиваем x11-apps, xauth. В ENV пишем название сети host.docker.internal(то как определяется хост машина докер контейнерами).

Изображение выглядит как текст, снимок экрана, визитная карточка, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок – Dockerfile

Запускаем контейнер из образа.

Изображение выглядит как текст, снимок экрана, круг

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Глазки

1. Запуск calc.jar

Для запуска калькулятора создаем такой же Dockerfile, но добавляем скачивание jdk и копирование бинарника.

Изображение выглядит как текст, снимок экрана, визитная карточка, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок – Dockerfile

Запускаем контейнер.

Изображение выглядит как электроника, текст, калькулятор, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок - Калькулятор

Вывод: в результате практической работы были изучены принципы работы с xserver, запущены 2 приложения внутри docker контейнеров с помощью xserver.