

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

Тема:

«Создание Dockerfile и сборка образа»

Выполнил(а): Морозова Валерия 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.

Вариант 10. Создайте Dockerfile для веб-сервера Apache, который обслуживает статический HTML-файл с надписью "Hello, Apache!".

```
dev@dev-vm:~$ mkdir apache-app
dev@dev-vm:~$ cd apache-app
dev@dev-vm:~/apache-app$ Valeria M
```

Рисунок 1. Создан новый каталог в терминале



Рисунок 2. Создан HTML-файл с надписью "Hello, Apache!"

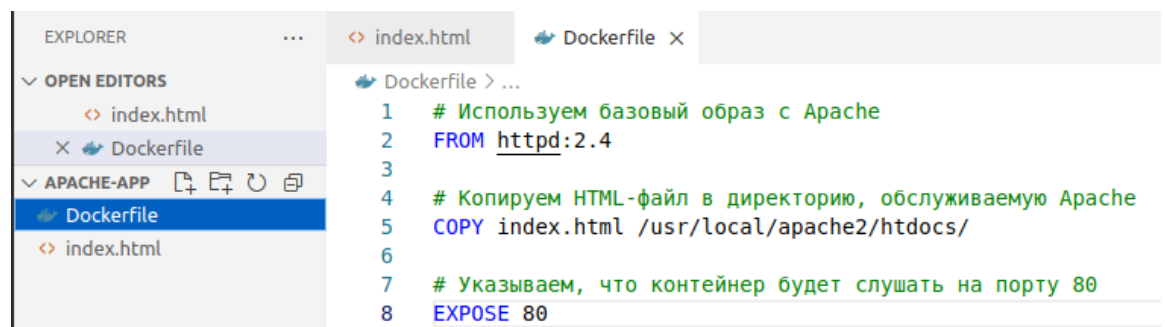


Рисунок 3. Создан Dockerfile

```

dev@dev-vm:~/apache-app$ docker build -t apache-app .
2025/03/09 02:53:46 in: [string{}]
2025/03/09 02:53:46 Parsed entitlements: []
[+] Building 22.6s (7/7) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile                0.0s
=> => transferring dockerfile: 352B                                0.0s
=> [internal] load metadata for docker.io/library/httpd:2.4        3.0s
=> [internal] load .dockerignore                                  0.0s
=> => transferring context: 2B                                       0.0s
=> [internal] load build context                                    0.0s
=> => transferring context: 273B                                      0.0s
=> [1/2] FROM docker.io/library/httpd:2.4@sha256:10381816bb7e60ae3a9db3784f2966a8910b6ff07c4da54bd2d62d267 19.2s
=> => resolve docker.io/library/httpd:2.4@sha256:10381816bb7e60ae3a9db3784f2966a8910b6ff07c4da54bd2d62d2671 0.0s
=> => sha256:10381816bb7e60ae3a9db3784f2966a8910b6ff07c4da54bd2d62d2671c8ab6e 10.16kB / 10.16kB 0.0s
=> => sha256:0de612e991352926be994158377c9f14147fe31e3ae92927f3b1b37dbf88ab10 7.88kB / 7.88kB 0.0s

```

Рисунок 4. Создан образ с помощью команды в терминале

```

dev@dev-vm:~/apache-app$ docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
apache-app          latest          6afe62ce5fbe   7 minutes ago   148MB

```

Рисунок 5. Проверка запуска образа

```

dev@dev-vm:~/apache-app$ docker run -p 80:80 apache-app
time="2025-03-09T03:04:35+03:00" level=error msg="Error waiting for container: driver failed programming external connectivity on endpoint elegant_mahavira (19982a2071eb7e349d5e32153363489ccc71b33542a5ef2deb747a545fc73c1): Bind for 0.0.0.0:80 failed: port is already allocated"
docker: Error response from daemon: driver failed programming external connectivity on endpoint elegant_mahavira (19982a2071eb7e349d5e32153363489ccc71b33542a5ef2deb747a545fc73c1): Bind for 0.0.0.0:80 failed: port is already allocated

```

Рисунок 6. Ошибка при запуске контейнера

Ввиду того, что порт, на котором было необходимо запустить контейнер, уже занят, как показала ошибка, поэтому была выполнена команда, чтобы выявить, какие процессы занимают порт.

```

dev@dev-vm:~/apache-app$ sudo lsof -i :80
[sudo] password for dev:
COMMAND    PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
docker-pr 6963 root    7u   IPv4 39128      0t0  TCP *:http (LISTEN)
docker-pr 6971 root    7u   IPv6 39129      0t0  TCP *:http (LISTEN)

```

Рисунок 7. Приложения, которые используют порт 80

```

dev@dev-vm:~/apache-app$ sudo kill -9 6963
dev@dev-vm:~/apache-app$ sudo kill -9 6971
dev@dev-vm:~/apache-app$ sudo lsof -i :80
dev@dev-vm:~/apache-app$ docker run -p 80:80 apache-app
docker: Error response from daemon: driver failed programming external connectivity on endpoint priceless_keldysh (5b85b40d25b6042ce799479e13a26226e382516934870c388573211897052a6f): Bind for 0.0.0.0:80 failed: port is already allocated

```

Рисунок 8. Завершение процессов с помощью их PID

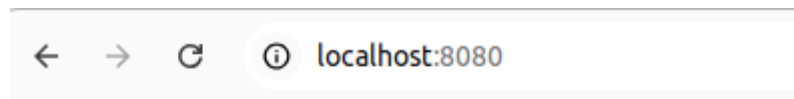
Как видно на рисунке 8, процессы завершились, но это не помогло запустить контейнер. Поэтому был изменен порт для запуска, на котором контейнер успешно запустился (рисунок 9).

```

dev@dev-vm:~/apache-app$ docker run -d -p 8080:80 apache-app
e7b37cd9cf222ce3e2e6681da95ba5f2fafd612f20527b6c34d30e3b7e679176
dev@dev-vm:~/apache-app$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS
e7b37cd9cf22   apache-app    "httpd-foreground"      25 seconds ago Up 24 seconds   0.0.0.0

```

Рисунок 9. Запуск контейнера на новом порте



Hello, Apache!

Рисунок 10. Проверка работоспособности

Чтобы увидеть это сообщение нужно было открыть веб-браузер и перейти по адресу `http://localhost:8080`. Проверка работоспособности прошла успешно, теперь контейнер можно остановить по его идентификатору.

```
dev@dev-vm:~/apache-app$ docker ps
CONTAINER ID   IMAGE                                NAMES                                COMMAND
e7b37cd9cf22   apache-app                          goofy_lederberg                     "httpd-foreground"
8080->80/tcp, [::]:8080->80/tcp
c1e5cea0bf2d   nodejs-redis-app                   recommendation_app                  "docker-entrypoint.s..."
3000->3000/tcp, [::]:3000->3000/tcp
1aa9e7aafe3a   oliver006/redis_exporter:latest    redis_exporter                      "/redis_exporter"
9121->9121/tcp, [::]:9121->9121/tcp
607eddc7e551   redis:latest                        "docker-entrypoint.s..."
6379->6379/tcp, [::]:6379->6379/tcp
a5bb7206afbd   postgres:16                         "docker-entrypoint.s..."
5432->5432/tcp, [::]:5432->5432/tcp
ec4919b750da   dpage/pgadmin4:latest              "docker-entrypoint.s..."
80->80/tcp, [::]:80->80/tcp, 443/tcp
pgadmin
dev@dev-vm:~/apache-app$ docker stop e7b37cd9cf22
e7b37cd9cf22
```

Рисунок 11. Остановка контейнера

Выводы:

1. Создан Dockerfile для веб-сервера Apache, который обслуживает статический HTML-файл с надписью "Hello, Apache!"
2. Собран образ Docker с использованием созданного Dockerfile.
3. Запустить контейнер из собранного образа и проверена его работоспособность.
4. Выполнено индивидуальное задание.

Контрольные вопросы:

Что такое Dockerfile и для чего он используется?

Dockerfile — это текстовый файл, содержащий ряд инструкций по созданию образа Docker. Он определяет базовый образ, код приложения, зависимости и любые другие конфигурации, необходимые для создания образа.

Какие основные инструкции используются в Dockerfile?

FROM: Указывает базовый образ для использования.

COPY: Копирует файлы из локальной файловой системы в образ.

RUN: Выполняет команды в образе во время процесса сборки.

CMD: Указывает команду для запуска при запуске контейнера из образа.

Как выполняется сборка образа Docker с использованием Dockerfile?

Создаётся Dockerfile. Это специальный файл с инструкциями для Docker по созданию нового образа.

Dockerfile помещают в корень папки приложения. Если уже есть папка с приложением, то Dockerfile помещают в корень проекта.

При запуске команды docker build демон Docker считывает Dockerfile и выполняет каждую инструкцию по порядку. Каждая инструкция создаёт облегчённый снимок файловой системы, доступный только для чтения, известный как слой.

Docker кэширует неизменные слои для ускорения последующих сборок. Слои, которые не менялись с момента предыдущей сборки, кэшируются и используются повторно, что позволяет избежать необходимости перестраивать эти слои и значительно сократить время сборки.

После выполнения всех инструкций Docker создаёт окончательный образ, содержащий приложение и его зависимости, готовый к использованию для запуска контейнеров.

Как запустить контейнер из собранного образа?

Чтобы запустить контейнер из собранного образа в Docker, нужно выполнить команду `docker run`. Она создаёт из образа контейнер и запускает его.

Каковы преимущества использования Dockerfile для создания образов Docker?

Автоматизация процесса. Dockerfile позволяет указать все шаги для создания образа и выполнить многие изменения автоматически, что занимает больше времени при ручном создании.

Возможность создавать изображения для разных версий. Например, если нужно протестировать одну и ту же настройку на разных версиях ОС.

Понятная другим пользователям структура. По систематическим шагам в Dockerfile легко узнать, какая именно конфигурация изменилась в базовом изображении.

Возможность использовать совместно с командой или организацией. Для этого можно разместить Dockerfile в частном репозитории Docker Hub.

Возможность передавать Dockerfile другим пользователям и легко обновлять. Это позволяет изменять изображение в зависимости от требований, например усилить безопасность, добавить или обновить сведения о пользователе.