Департамент образования города Москвы

Государственное автономное образовательное учреждение
высшего образования города Москвы
«Московский городской педагогический университет»

Институт цифрового образования
Департамент информатики, управления и технологий

Лабораторная работа 1.1
по дисциплине «Интеграция и развертывание программного обеспечения с помощью контейнеров»

Тема: «Установка и настройка Docker. Работа с контейнерами Docker»

Направление подготовки 38.03.05 – бизнес-информатика
Профиль подготовки «Аналитика данных и эффективное управление»
(очная форма обучения)

Выполнила:
Студентка группы АДЭУ-211
Белик Мария Константиновна

Преподаватель:
Босенко Т.М.

Москва
2025

ВВЕДЕНИЕ

Цель работы: освоить процесс установки и настройки Docker, научиться работать с контейнерами и образами Docker.

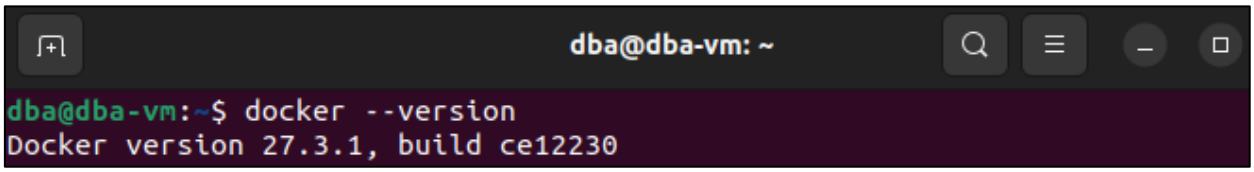
Задачи:

1. Установить Docker на локальный компьютер.
2. Проверить корректность установки Docker.
3. Ознакомиться с основными командами Docker CLI для работы с образами и контейнерами.
4. Выполнить индивидуальное задание.

Вариант 1. Загрузить образ *nginx*, запустить контейнер, настроить маршрутизацию портов и проверить доступность веб-сервера.

ХОД РАБОТЫ

1. Проверка наличия docker – установлена его версия 27.3.1:



2. Добавление текущего пользователя в группу docker для удобства выполнения последующих команд. Проверка успешности на примере выполнения команды *docker images* (вывод списка локальных образов docker) – она отработала без использования *sudo:*

```
dba@dba-vm:~$ sudo usermod -aG docker dba newgrp docker
Usage: usermod [options] LOGIN

Options:
  -b, --badnames                allow bad names
  -c, --comment COMMENT         new value of the GECOS field
  -d, --home HOME_DIR           new home directory for the user account
  -e, --expiredate EXPIRE_DATE  set account expiration date to EXPIRE_DATE
  -f, --inactive INACTIVE       set password inactive after expiration
                                to INACTIVE
  -g, --gid GROUP               force use GROUP as new primary group
  -G, --groups GROUPS           new list of supplementary GROUPS
  -a, --append                  append the user to the supplemental GROUPS
                                mentioned by the -G option without removing
                                the user from other groups
  -h, --help                    display this help message and exit
  -l, --login NEW_LOGIN         new value of the login name
  -L, --lock                    lock the user account
  -m, --move-home               move contents of the home directory to the
                                new location (use only with -d)
  -o, --non-unique              allow using duplicate (non-unique) UID
  -p, --password PASSWORD       use encrypted password for the new password
  -R, --root CHROOT_DIR         directory to chroot into
  -P, --prefix PREFIX_DIR       prefix directory where are located the /etc/*
 files
  -s, --shell SHELL             new login shell for the user account
  -u, --uid UID                 new UID for the user account
  -U, --unlock                  unlock the user account
  -v, --add-subuids FIRST-LAST  add range of subordinate uids
  -V, --del-subuids FIRST-LAST  remove range of subordinate uids
  -w, --add-subgids FIRST-LAST  add range of subordinate gids
  -W, --del-subgids FIRST-LAST  remove range of subordinate gids
  -Z, --selinux-user SEUSER     new SELinux user mapping for the user account

dba@dba-vm:~$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
hello-world   latest    74cc54e27dc4   4 weeks ago    10.1kB
myapp         latest    3e1b602bae81   3 months ago   649MB
postgres      latest    d57ed788c154   4 months ago   434MB
dba@dba-vm:~$
```

3. Проверка, что docker запущен и может загружать образы и запускать контейнеры:

```
dba@dba-vm:~$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent i
t
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

dba@dba-vm:~$
```

4. Просмотр текущих запущенных контейнеров (таких нет). Вывод списка всех контейнеров, которые были когда-либо запущены (таких всего 4):

```
dba@dba-vm:~$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
dba@dba-vm:~$ docker ps -a
CONTAINER ID   IMAGE         COMMAND                  CREATED          STATUS
                     PORTS                                          NAMES
7c31e3197d12   hello-world   "/hello"                 51 seconds ago   Exited
 (0) 51 seconds ago                                                 beautiful_h
ofstadter
09711d438536   hello-world   "/hello"                 10 minutes ago   Exited
 (0) 10 minutes ago                                                 musing_leav
itt
a457dbff5f72   myapp         "python -m uvicorn s…"   3 months ago     Exited
 (255) 4 weeks ago    0.0.0.0:8000->8000/tcp, :::8000->8000/tcp   myapp
fc7c96c36fe7   postgres      "docker-entrypoint.s…"   3 months ago     Exited
 (255) 4 weeks ago    0.0.0.0:5432->5432/tcp, :::5432->5432/tcp   mydb
dba@dba-vm:~$
```

5. В соответствии с 1 вариантом: загрузка образа *nginx*.:

```
dba@dba-vm:~$ docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
c29f5b76f736: Pull complete
e19db8451adb: Pull complete
24ff42a0d907: Pull complete
c558df217949: Pull complete
976e8f6b25dd: Pull complete
6c78b0ba1a32: Pull complete
84cade77a831: Pull complete
Digest: sha256:91734281c0ebfc6f1aea979cffeed5079cfe786228a71cc6f1f46a228cde6e
34
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
dba@dba-vm:~$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
nginx         latest    97662d24417b   2 weeks ago    192MB
hello-world   latest    74cc54e27dc4   4 weeks ago    10.1kB
myapp         latest    3e1b602bae81   3 months ago   649MB
postgres      latest    d57ed788c154   4 months ago   434MB
dba@dba-vm:~$
```

6. Запуск контейнера *nginx* с настройкой маршрутизации порта 80:80. Проверка текущих запущенных контейнеров – наш запущен:

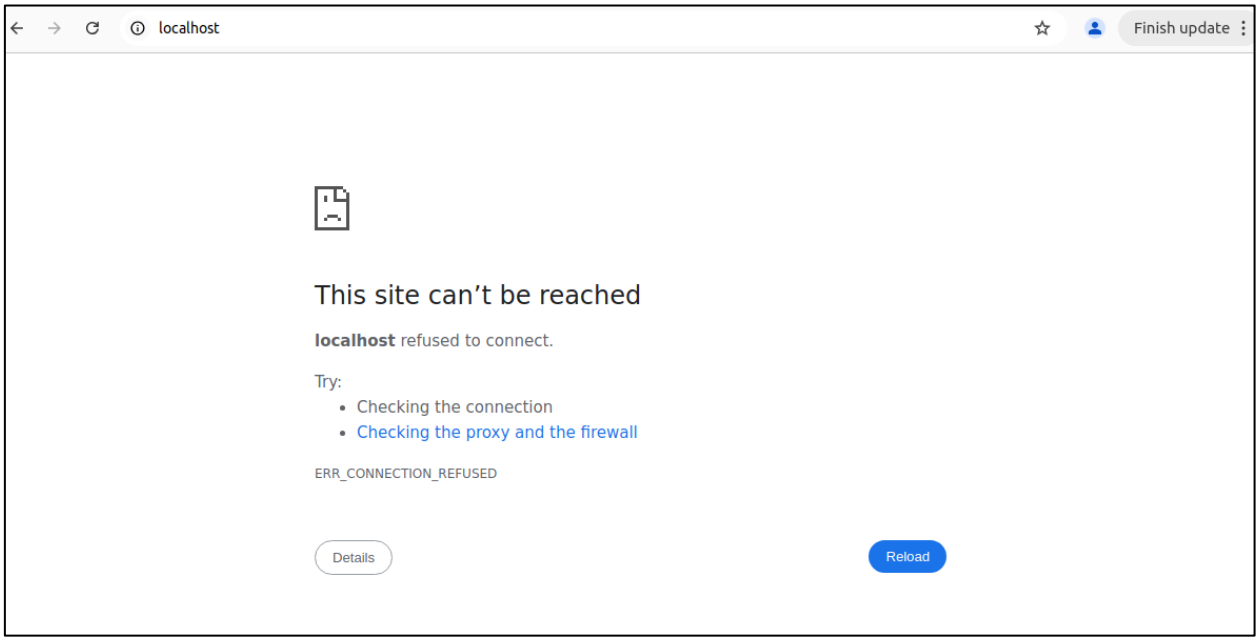
```
dba@dba-vm:~$ docker run -d -p 80:80 nginx
d017306f883af8114cca76f50e96e499869e39af1ff111e088ad88f1226fb389
dba@dba-vm:~$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS
    PORTS                                  NAMES
d017306f883a   nginx     "/docker-entrypoint.…"   6 seconds ago   Up 6 second
s   0.0.0.0:80->80/tcp, :::80->80/tcp    optimistic_williamson
```

7. Проверка доступности веб-сервера:

8. В конце остановка и удаление контейнера:

```
dba@dba-vm:~$ docker container stop d017306f883a
d017306f883a
dba@dba-vm:~$ docker container rm d017306f883a
d017306f883a
dba@dba-vm:~$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
dba@dba-vm:~$
```

Проверка, что веб-сервер стал недоступным после остановки контейнера:



ВЫВОДЫ

В ходе выполнения данной лабораторной работы были выполнены все поставленные задачи:

1. проверена корректность установки Docker;
2. успешно отработаны основные команды Docker CLI для работы с образами и контейнерами (*docker images, docker ps, docker ps -a, docker stop, docker rm*);
3. выполнено индивидуальное задание – загрузка образа nginx, запуск контейнера с настройкой маршрутизации, доступность веб-сервера.

Таким образом была достигнута цель работы - освоить процесс установки и настройки Docker, научиться работать с контейнерами и образами Docker.

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое Docker и для чего он используется?

Docker — это платформа, которая позволяет упаковать в контейнер приложение со всем окружением и зависимостями, а затем доставить и запустить его в целевой системе.

Приложение, упакованное в контейнер, изолируется от операционной системы и других приложений. Поэтому разработчики могут не задумываться, в каком окружении будет работать их приложение, а инженеры по эксплуатации — единообразно запускать приложения и меньше заботиться о системных зависимостях.

Использование:

- Архитектура микросервисов: контейнеризация идеально подходит для микросервисов, где приложения разбиваются на более мелкие независимые сервисы. Каждый сервис может работать в своем собственном контейнере, что упрощает управление и масштабирование.
- Непрерывная интеграция/непрерывное развертывание (CI/CD): контейнеры облегчают конвейеры CI/CD, предоставляя согласованные среды для создания, тестирования и развертывания приложений. Это приводит к более быстрым циклам выпуска и более надежным развертываниям.
- Гибридные облачные развертывания: организации могут использовать контейнеры для развертывания приложений в нескольких облачных провайдерах или локальной инфраструктуре, обеспечивая гибкость и избегая привязки к поставщику.
- Разработка и тестирование: разработчики могут создавать изолированные среды для тестирования новых функций или исправления ошибок, не затрагивая основное приложение. Это позволяет проводить быстрые эксперименты и итерации.

— Модернизация устаревших приложений: организации могут контейнеризировать устаревшие приложения, чтобы сделать их более портативными и простыми в управлении. Это может продлить срок службы старых приложений, одновременно модернизируя их развертывание.
— Большие данные и машинное обучение: контейнеры можно использовать для развертывания моделей обработки данных и машинного обучения, что позволяет специалистам по данным работать в согласованных средах и масштабировать свои приложения по мере необходимости.

2. Какие преимущества дает использование контейнеров Docker по сравнению с виртуальными машинами?

Эффективность: контейнеры требуют меньше накладных расходов, чем ВМ, поскольку они совместно используют ОС хоста. Это приводит к более быстрому запуску и лучшему использованию ресурсов.

Масштабируемость: контейнеры можно легко масштабировать вверх или вниз в зависимости от спроса. Вы можете быстро развернуть новые экземпляры контейнеров для обработки возросшей нагрузки.

Согласованность в разных средах: контейнеры инкапсулируют все зависимости, гарантируя, что приложения будут работать одинаково при разработке, тестировании и производстве. Это устраняет проблему «работает на моей машине».

Упрощенное развертывание: с контейнерами развертывание приложений становится простым. Вы можете упаковать свое приложение и его зависимости в один образ контейнера, который можно легко распространять и развертывать.

Изоляция: контейнеры обеспечивают уровень изоляции между приложениями. Это означает, что, если один контейнер выходит из строя, это не влияет на другие контейнеры, работающие на том же хосте.

Быстрая разработка и тестирование: разработчики могут быстро создавать, тестировать и итерировать приложения в изолированных средах, не беспокоясь о конфликтах с другими приложениями.

3. Что такое образ Docker и как он связан с контейнерами?

Образ Docker — это легкий, автономный и исполняемый пакет, который включает в себя все необходимое для запуска программного обеспечения, включая код, среду выполнения, библиотеки, переменные среды и файлы конфигурации. Образы — это шаблоны, доступные только для чтения, используемые для создания контейнеров.

Контейнер Docker — это запускаемый экземпляр образа Docker. Контейнеры создаются из образов и могут быть запущены, остановлены, перемещены и удалены. В отличие от образов, контейнеры изменяемы и могут изменяться во время выполнения.

То есть, по сути, мы берем неизменяемый шаблон (образ) и на его основе запускаем контейнер.

4. Какие основные команды Docker CLI вы узнали в ходе выполнения лабораторной работы?

`docker pull` – загрузка образа

`docker run` – создание и запуск контейнера из образа

`docker ps` – вывод списка всех запущенных контейнеров

`docker ps -a` - вывод списка всех контейнеров, включая остановленные

`docker stop` – остановка запущенного контейнера

`docker start` – запуск остановленного контейнера

`docker rm` – удаление остановленного контейнера (прежде необходимо остановить контейнер!)

`docker images` – вывод списка всех загруженных образов

5. **Как можно настроить маршрутизацию портов при запуске контейнера Docker?**

Для настройки маршрутизации портов при запуске контейнера Docker используется параметр *-p* или *--publish*. Этот параметр позволяет сопоставить порты контейнера с портами хоста.

```
docker run -p <порт_хоста>:<порт_контейнера> <имя_образа>
```

Пример настройки:

```
docker run -d -p 80:80 nginx
```

Параметр -d в команде означает, что контейнер будет запущен в фоновом режиме (detached mode). Это позволяет продолжать использовать терминал после запуска контейнера, не блокируя его.