Федеральное государственное бюджетное образовательное учреждение высшего образования

Московский государственный технический университет

им. Н. Э. Баумана

Национальный исследовательский университет (МГТУ им. Н.Э. Баумана)

Практическое задание:

«Облачные решения. Основы контейнеризации Docker»

Выполнил: Шарафутдинов Э. М.

Группа: РЛ6-61Б

Москва 2023 г.

Содержание

Цель работы

Ознакомиться с системой контейнеризации Docker. Научиться создавать, устанавливать и проводить базовые операции управления контейнерами в системе контейнеризации Docker.

Краткие теоретические сведения

Контейнеризация — метод виртуализации, при котором ядро операционной системы поддерживает несколько изолированных экземпляров пространства пользователя вместо одного. Эти экземпляры (обычно называемые контейнерами или зонами) с точки зрения выполняемых в них процессов идентичны отдельному экземпляру операционной системы. Для систем на базе Unix эта технология похожа на улучшенную реализацию механизма chroot. Ядро обеспечивает полную изолированность контейнеров, поэтому программы из разных контейнеров не могут воздействовать друг на друга.

В отличие от аппаратной виртуализации, при которой эмулируется аппаратное окружение и может быть запущен широкий спектр гостевых операционных систем, в контейнере может быть запущен экземпляр операционной системы только с тем же ядром, что и у хостовой операционной системы (все контейнеры узла используют общее ядро). При этом при контейнеризации отсутствуют дополнительные ресурсные накладные расходы на эмуляцию виртуального оборудования и запуск полноценного экземпляра операционной системы, характерные при аппаратной виртуализации

Порядок выполнения работы

1. Установка Docker на ВМ.

```
emmmm@debian:~$ sudo apt-get update

Мы полагаем, что ваш системный администратор изложил вам основы
безопасности. Как правило, всё сводится к трём следующим правилам:

    №1) Уважайте частную жизнь других.
    №2) Думайте, прежде что-то вводить.
    №3) С большой властью приходит большая ответственность.

[sudo] пароль для emmmm:
Сущ:1 http://security.debian.org/debian-security bullseye-security InRelease
Сущ:2 http://deb.debian.org/debian bullseye InRelease
Сущ:3 http://deb.debian.org/debian bullseye-updates InRelease
```

Рисунок 1 - обновление базы пакетного репозитория apt

```
emmmm@debian:~$ sudo apt-get install -y ca-certificates curl gnupg lsb-relea
se docker.io docker-compose mcedit
Чтение списков пакетов… Готово
Построение дерева зависимостей… Готово
Чтение информации о состоянии… Готово
Заметьте, вместо «mcedit» выбирается «mc»
Уже установлен пакет ca-certificates самой новой версии (20210119).
Уже установлен пакет gnupg самой новой версии (2.2.27-2+deb11u2).
gnupg помечен как установленный вручную.
Уже установлен пакет lsb-release самой новой версии (11.1.0).
lsb-release помечен как установленный вручную.
Будут установлены следующие дополнительные пакеты:
  binutils binutils-common binutils-x86-64-linux-gnu cgroupfs-mount
  containerd libbinutils libctf-nobfd0 libctf0 libintl-perl
  libintl-xs-perl libmodule-find-perl libmodule-scandeps-perl
  libproc-processtable-perl libsort-naturally-perl libterm-readkey-perl
  mc-data needrestart python3-attr python3-cached-property
  python3-distutils python3-docker python3-dockerpty python3-docopt
  python3-importlib-metadata python3-jsonschema python3-lib2to3
  python3-more-itertools python3-pyrsistent python3-setuptools
  python3-texttable python3-websocket python3-yaml python3-zipp runc tini
```

Рисунок 2 - установка базовых пакетов Docker

```
emmmm@debian:~$ systemctl status docker.service docker.socket
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2023-06-13 20:37:10 MSK; 6min ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 2944 (dockerd)
      Tasks: 7
     Memory: 62.3M
        CPU: 946ms
     CGroup: /system.slice/docker.service
             └─2944 /usr/sbin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

● docker.socket - Docker Socket for the API
     Loaded: loaded (/lib/systemd/system/docker.socket; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2023-06-13 20:37:06 MSK; 7min ago
   Triggers: ● docker.service
     Listen: /run/docker.sock (Stream)
      Tasks: 0 (limit: 1114)
     Memory: 0B
        CPU: 2ms
     CGroup: /system.slice/docker.socket
```

Рисунок 3 - проверка, запущен ли демон Docker

2. Подготовка проекта описания контейнеризуемого приложения.

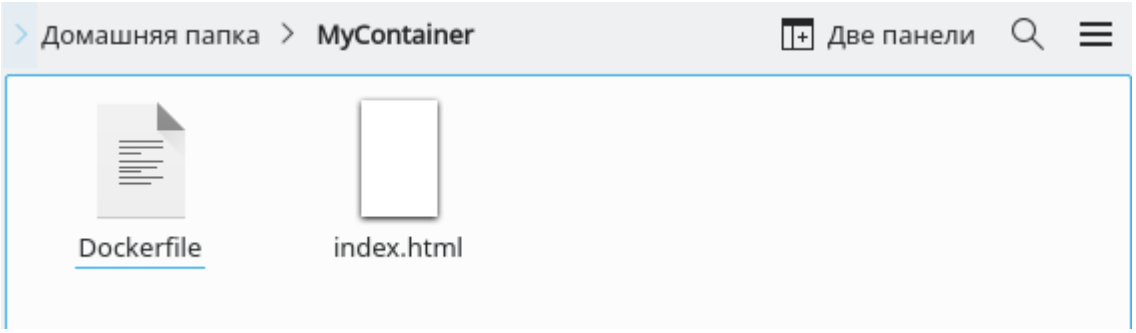

Рисунок 4 - папка MyContainer с Docker- и html-файлами

Dockerfile

```
1  FROM nginx:alpine
2  COPY index.html /usr/share/nginx/html/index.html
```

Рисунок 5 - Docker-файл

index.html

```
1  <!DOCTYPE html>
2  <html lang="ru">
3  <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width,initial-
   ↪ scale=1.0">
6     <title> Привет от Nginx в Docker-е</title>
7     <style>
8         h1{
9             font-weight:lighter;
10            font-family:Arial,Helvetica,sans-serif;
11        }
12    </style>
13 </head>
14 <body>
15
16 <h1>
17    Привет!
18 </h1>
19
20 </body>
21 </html>
22
```

Рисунок 6 - html-файл

3. Просмотр доступных образов.

Листинг 1 - вывод доступных образов командой `docker images ls`

```
emmmm@debian:~/MyContainer$ docker images ls
REPOSITORY                    TAG       IMAGE ID       CREATED          SIZE
nginx-container               latest    b790e75566eb   40 minutes ago   40.7MB
```

```
nginx-container              v1        b790e75566eb   40 minutes ago   40.7MB
```

4. Сборка основного контейнера.

Листинг 2 - сборка контейнера командой `docker build`

```
emmmm@debian:~/MyContainer$ docker build -t nginSending
build context to Docker daemon  3.072kB

Step 1/2 : FROM nginx:alpine

 ---> 2bc7edbc3cf2

Step 2/2 : COPY index.html
/usr/share/nginx/html/index.html

 ---> Using cache

 ---> b790e75566eb

Successfully built b790e75566eb

x-container:v1 -t nginx-container:latest
```

5. Запуск контейнера с привязкой к портам виртуальной машины командой docker run `--name bmstu-practice-nginx -p 80:80 -d nginx-container`

6. Просмотр журнала приложения, работающего в контейнере, командой `docker logs bmstu-practice-nginx`

7. Завершение работы с контйенером.

Листинг 3 - остановка контейнера

```
emmmm@debian:~/MyContainer$ docker exec -it
bmstu-practice-nginx /bin/sh
#cat /etc/nginx/nginx.conf
#exit
docker stop bmstu-practice-nginx
```

```
emmmm@debian:~/MyContainer$ docker ps docker restart
bmstu-practice-nginx
```

8. Перезапуск контейнера командой `docker restart bmstu-practice-nginx`.

Выводы

В ходе выполнения практического задания мы успешно ознакомились с основами работы с системой контейнеризации Docker. Мы изучили принципы и особенности контейнеризации, а также научились создавать, устанавливать и проводить базовые операции управления контейнерами в Docker.