1T33: Cloud Architecture

Lab- 2A: Laboratory on Docker and Docker Compose

Objective:

To provide students with a thorough understanding of Docker and Docker Compose, enabling them to containerize applications, manage multi-container setups, and apply best practices for containerization in real-world scenarios. Participants will learn to build, deploy, and manage containerized applications efficiently.

Outcomes:

By the end of this laboratory, participants will be able to:

1. Understand and Apply Containerization Concepts:
   - Grasp the fundamentals of Docker and containerization, distinguishing it from traditional virtualization.
   - Use Docker to containerize applications, creating and managing Docker images and containers.
2. Develop and Manage Multi-Container Applications:
   - Utilize Docker Compose to define and orchestrate multi-container applications.
   - Connect and manage the interactions between different containers (e.g., web applications and databases) using Docker Compose.
3. Implement Advanced Docker Features:
   - Apply advanced Docker and Docker Compose features, such as networking, volumes, and environment variables, to build scalable and maintainable applications.
   - Utilize Docker best practices to write efficient Dockerfiles, secure containers, and manage persistent data storage.
4. Troubleshoot and Optimize Docker Applications:
   - Diagnose and resolve common issues in Docker and Docker Compose environments, using logs and debugging tools.
   - Optimize Docker applications for performance and security, ensuring they run efficiently in production environments.

System Requirements:

Ubuntu Linux with Internet connectivity

Step-by-step Procedure:

Part 1: Introduction to Docker

1. Objective:

- Understand the basics of Docker, its components, and its use cases.

2. Materials Needed:

- Computer with internet access

- Docker installed (Docker Desktop for Windows/Mac or Docker Engine for Linux)

3. Steps:

1.1. Introduction to Docker

- Overview of containerization

- Differences between VMs and containers

- Docker architecture (Docker Engine, Docker Daemon, Docker Client, Docker Hub)

1.2. Installing Docker

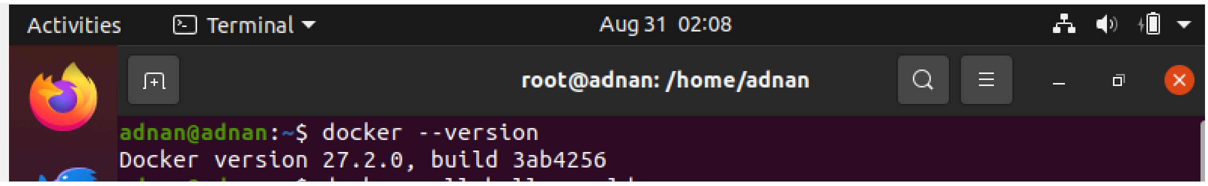
- Installation guide for different OS:

- Windows/Mac: Download and install Docker Desktop from the [Docker website](https://www.docker.com/products/docker-desktop).

- Linux: Follow the instructions on the [Docker Engine installation page](https://docs.docker.com/engine/install/).

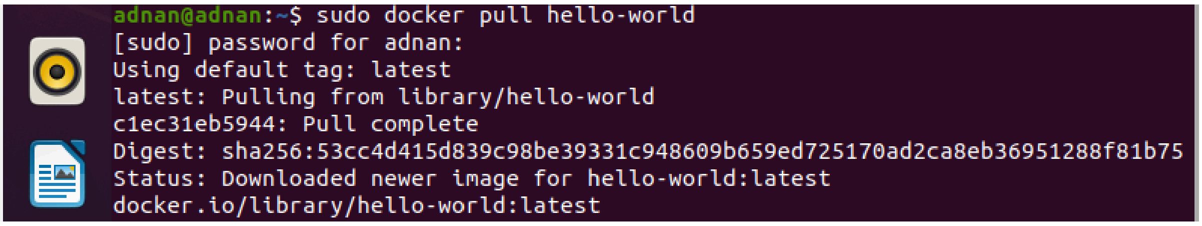
1.3. Docker Commands Basics

- Verify installation with `docker --version`



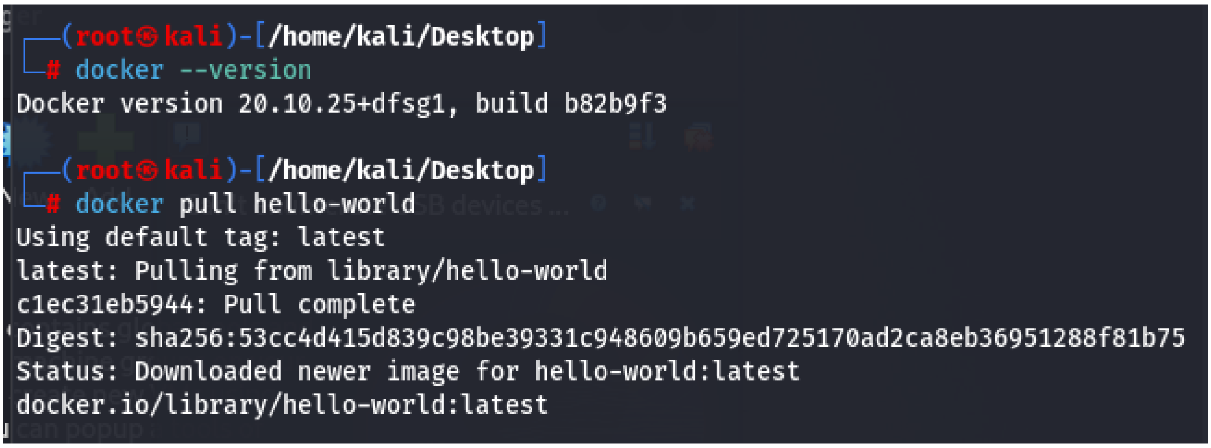
```
adnan@adnan:~$ docker --version
Docker version 27.2.0, build 3ab4256
```

- Pulling images: `docker pull hello-world`



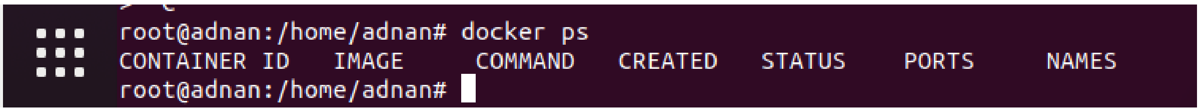
```
adnan@adnan:~$ sudo docker pull hello-world
[sudo] password for adnan:
Using default tag: latest
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:53cc4d415d839c98be39331c948609b659ed725170ad2ca8eb36951288f81b75
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest
```

- Running containers: `docker run hello-world`

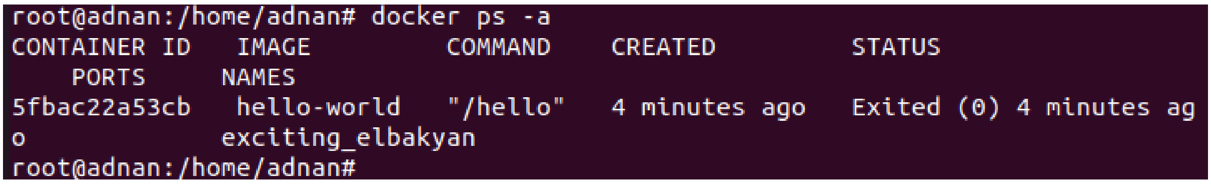


```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# docker --version
Docker version 20.10.25+dfsg1, build b82b9f3

┌──(root㉿kali)-[/home/kali/Desktop]
└─# docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:53cc4d415d839c98be39331c948609b659ed725170ad2ca8eb36951288f81b75
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest
```

- Listing containers: `docker ps`, `docker ps -a`

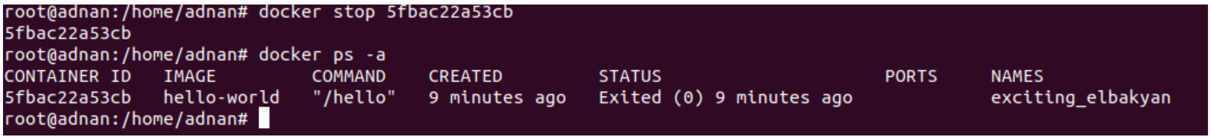


```
root@adnan:/home/adnan# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
root@adnan:/home/adnan# 
```



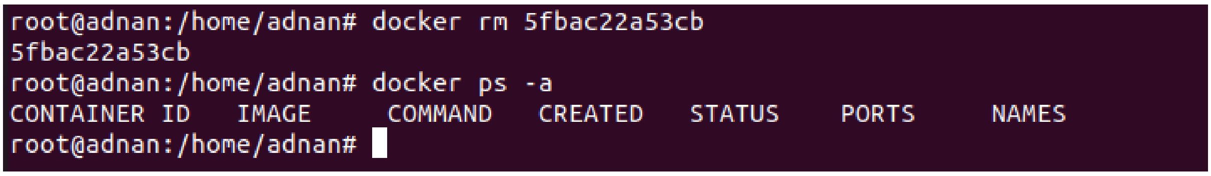
```
root@adnan:/home/adnan# docker ps -a
CONTAINER ID   IMAGE         COMMAND    CREATED         STATUS
    PORTS     NAMES
5fbac22a53cb   hello-world   "/hello"   4 minutes ago   Exited (0) 4 minutes ag
o             exciting_elbakyan
root@adnan:/home/adnan#
```

- Stopping containers: `docker stop [container_id]`



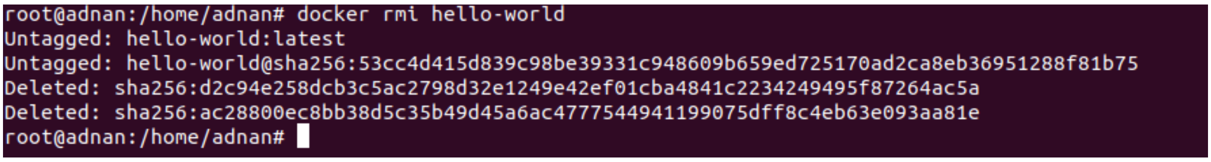
```
root@adnan:/home/adnan# docker stop 5fbac22a53cb
5fbac22a53cb
root@adnan:/home/adnan# docker ps -a
CONTAINER ID   IMAGE         COMMAND    CREATED         STATUS                     PORTS     NAMES
5fbac22a53cb   hello-world   "/hello"   9 minutes ago   Exited (0) 9 minutes ago             exciting_elbakyan
root@adnan:/home/adnan# 
```

- Removing containers: `docker rm [container_id]`



```
root@adnan:/home/adnan# docker rm 5fbac22a53cb
5fbac22a53cb
root@adnan:/home/adnan# docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
root@adnan:/home/adnan# 
```

- Removing images: `docker rmi [image_id]`



```
root@adnan:/home/adnan# docker rmi hello-world
Untagged: hello-world:latest
Untagged: hello-world@sha256:53cc4d415d839c98be39331c948609b659ed725170ad2ca8eb36951288f81b75
Deleted: sha256:d2c94e258dcb3c5ac2798d32e1249e42ef01cba4841c2234249495f87264ac5a
Deleted: sha256:ac28800ec8bb38d5c35b49d45a6ac4777544941199075dff8c4eb63e093aa81e
root@adnan:/home/adnan# 
```

1T33: Cloud Architecture

Part 2: Working with Docker Images and Containers

1. Objective:

- Learn how to create, manage, and work with Docker images and containers.

2. Materials Needed:

- Docker installed

3. Steps:

2.1. Creating a Dockerfile

- Understanding Dockerfile syntax and commands (FROM, RUN, CMD, COPY, EXPOSE, etc.)

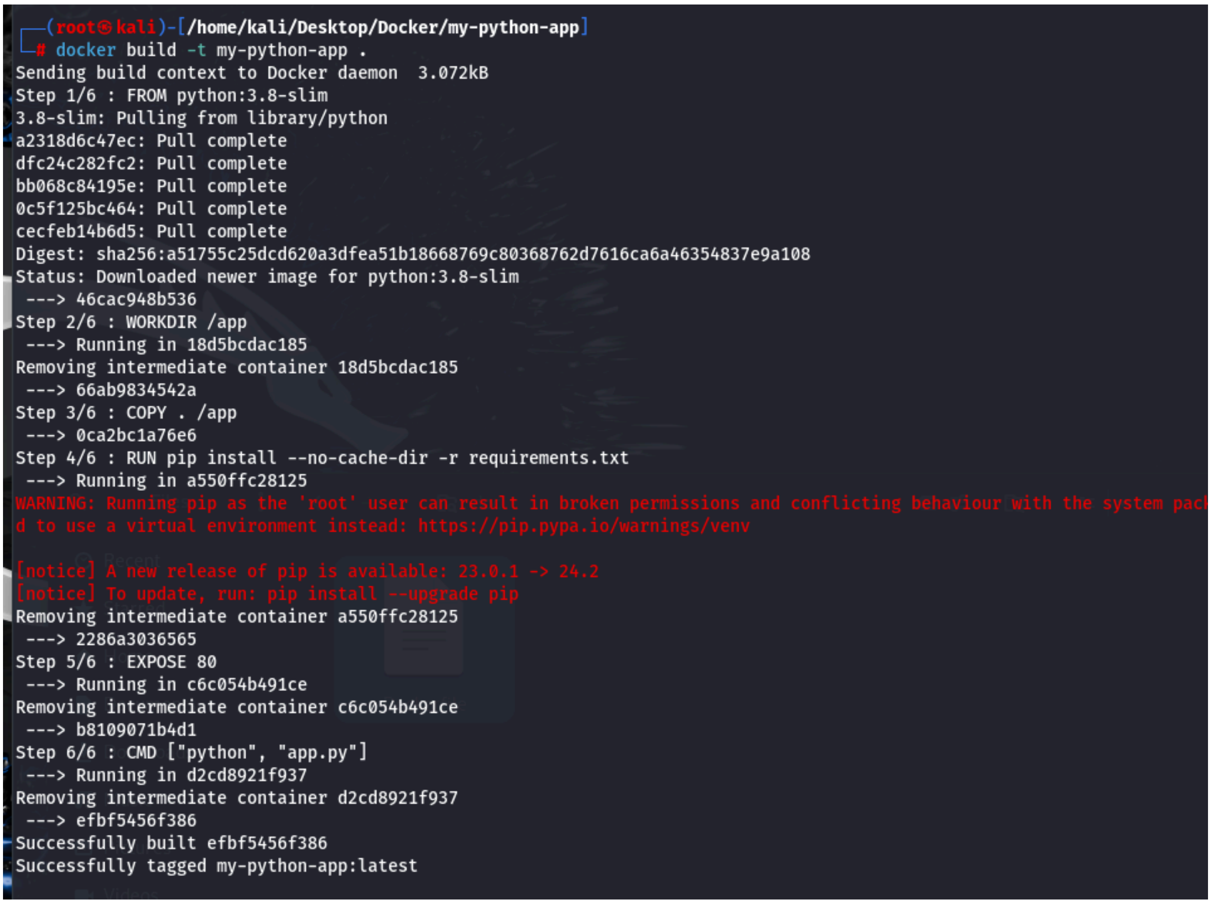
Example Dockerfile:

```Dockerfile
# Use an official Python runtime as a parent image
FROM python:3.8-slim

# Set the working directory in the container
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install any needed packages specified in requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Make port 80 available to the world outside this container
EXPOSE 80

# Run app.py when the container launches
CMD ["python", "app.py"]
```

- Creating a simple Dockerfile for a Python application

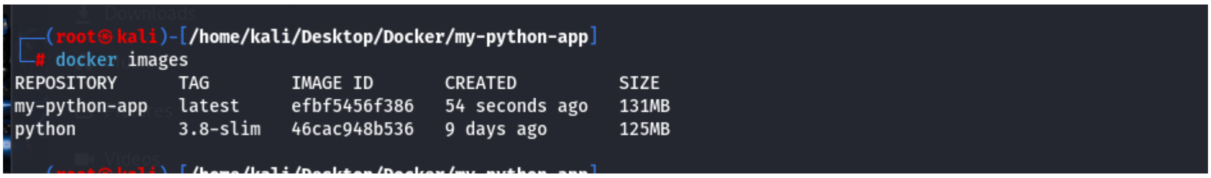
2.2. Building Docker Images

- Building an image from a Dockerfile: `docker build -t my-python-app .`



```
┌──(root㉿kali)-[/home/kali/Desktop/Docker/my-python-app]
└─# docker build -t my-python-app .
Sending build context to Docker daemon  3.072kB
Step 1/6 : FROM python:3.8-slim
3.8-slim: Pulling from library/python
a2318d6c47ec: Pull complete
dfc24c282fc2: Pull complete
bb068c84195e: Pull complete
0c5f125bc464: Pull complete
cecfeb14b6d5: Pull complete
Digest: sha256:a51755c25dcd620a3dfea51b18668769c80368762d7616ca6a46354837e9a108
Status: Downloaded newer image for python:3.8-slim
 ---> 46cac948b536
Step 2/6 : WORKDIR /app
 ---> Running in 18d5bcdac185
Removing intermediate container 18d5bcdac185
 ---> 66ab9834542a
Step 3/6 : COPY . /app
 ---> 0ca2bc1a76e6
Step 4/6 : RUN pip install --no-cache-dir -r requirements.txt
 ---> Running in a550ffc28125
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system pack
d to use a virtual environment instead: https://pip.pypa.io/warnings/venv

[notice] A new release of pip is available: 23.0.1 -> 24.2
[notice] To update, run: pip install --upgrade pip
Removing intermediate container a550ffc28125
 ---> 2286a3036565
Step 5/6 : EXPOSE 80
 ---> Running in c6c054b491ce
Removing intermediate container c6c054b491ce
 ---> b8109071b4d1
Step 6/6 : CMD ["python", "app.py"]
 ---> Running in d2cd8921f937
Removing intermediate container d2cd8921f937
 ---> efbf5456f386
Successfully built efbf5456f386
Successfully tagged my-python-app:latest
```

- Listing Docker images: `docker images`

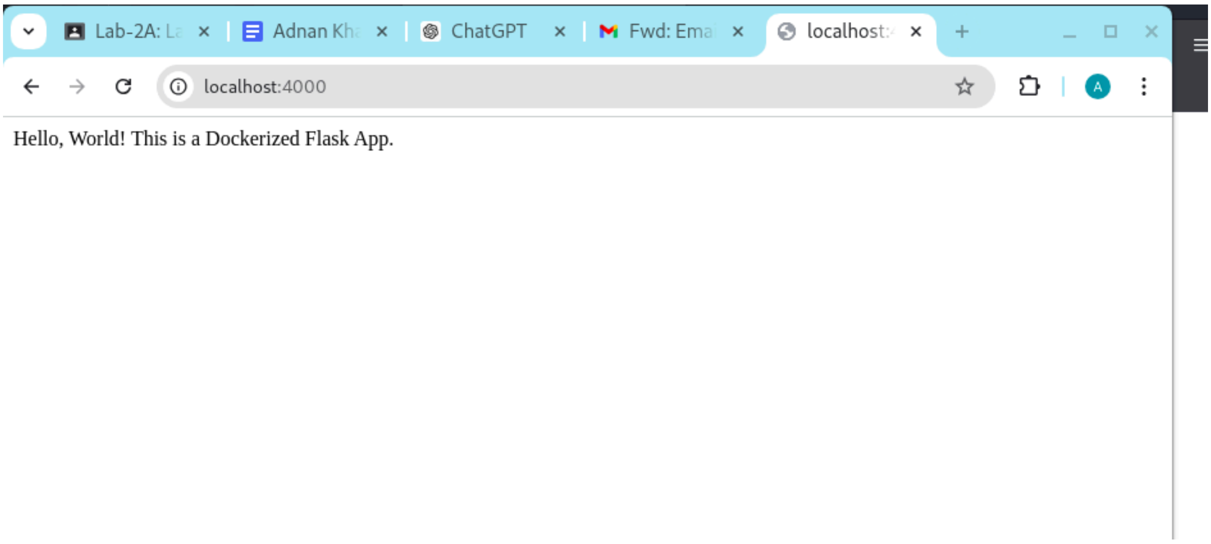


```
┌──(root㉿kali)-[/home/kali/Desktop/Docker/my-python-app]
└─# docker images
REPOSITORY      TAG        IMAGE ID       CREATED          SIZE
my-python-app   latest     efbf5456f386   54 seconds ago   131MB
python          3.8-slim   46cac948b536   9 days ago       125MB

┌──(root㉿kali)-[/home/kali/Desktop/Docker/my-python-app]
```

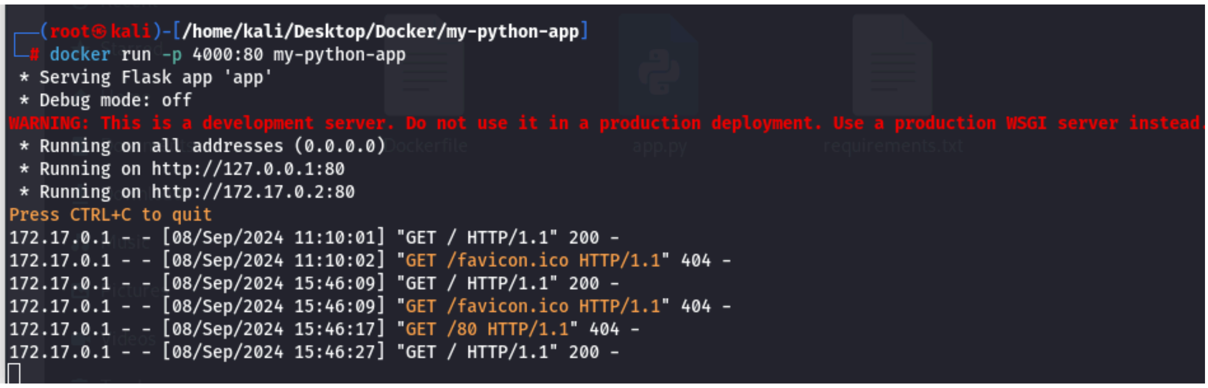
2.3. Running Docker Containers

- Running a container from an image: `docker run -p 4000:80 my-python-app`

- Accessing the running application in a browser: `http://localhost:4000`



Hello, World! This is a Dockerized Flask App.

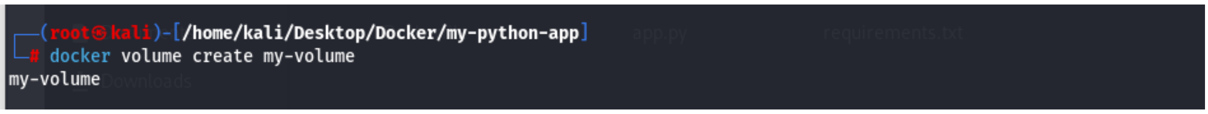


```
┌──(root㉿kali)-[/home/kali/Desktop/Docker/my-python-app]
└─# docker run -p 4000:80 my-python-app
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:80
 * Running on http://172.17.0.2:80
Press CTRL+C to quit
172.17.0.1 - - [08/Sep/2024 11:10:01] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [08/Sep/2024 11:10:02] "GET /favicon.ico HTTP/1.1" 404 -
172.17.0.1 - - [08/Sep/2024 15:46:09] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [08/Sep/2024 15:46:09] "GET /favicon.ico HTTP/1.1" 404 -
172.17.0.1 - - [08/Sep/2024 15:46:17] "GET /80 HTTP/1.1" 404 -
172.17.0.1 - - [08/Sep/2024 15:46:27] "GET / HTTP/1.1" 200 -
```

1T33: Cloud Architecture

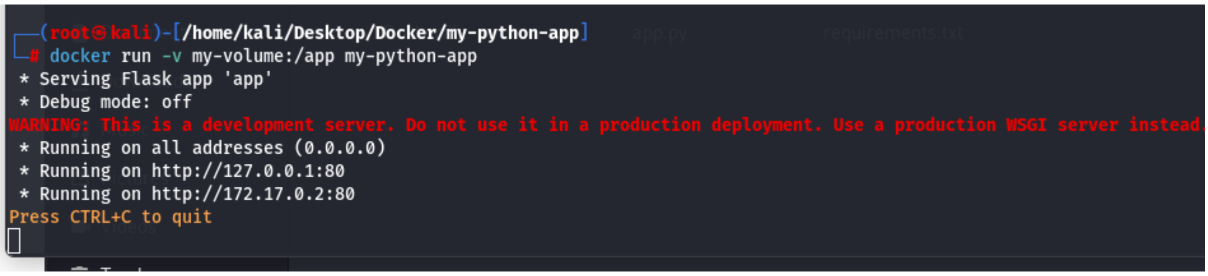
2.4. Managing Data with Volumes

- Creating and using Docker volumes: `docker volume create my-volume`



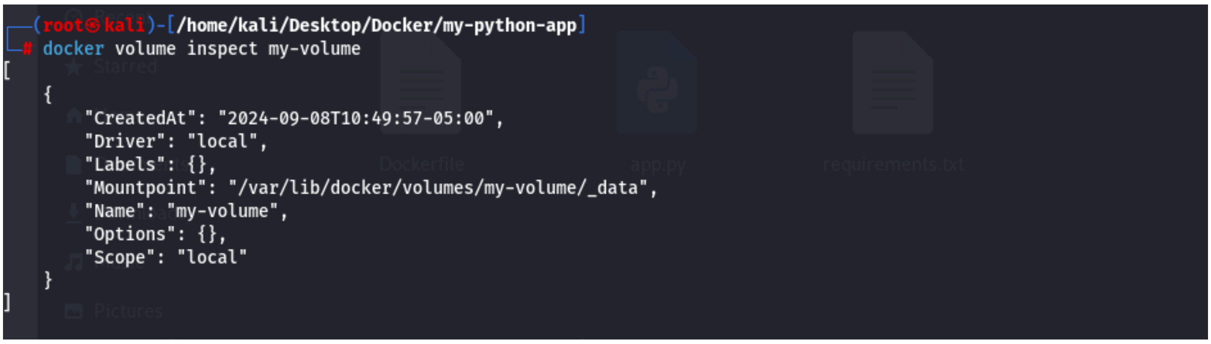
```
┌──(root㉿kali)-[/home/kali/Desktop/Docker/my-python-app]
└─# docker volume create my-volume
my-volume
```

- Mounting volumes: `docker run -v my-volume:/app my-python-app`



```
┌──(root㉿kali)-[/home/kali/Desktop/Docker/my-python-app]
└─# docker run -v my-volume:/app my-python-app
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:80
 * Running on http://172.17.0.2:80
Press CTRL+C to quit
```

- Inspecting volumes: `docker volume inspect my-volume`



```
┌──(root㉿kali)-[/home/kali/Desktop/Docker/my-python-app]
└─# docker volume inspect my-volume
[
    {
        "CreatedAt": "2024-09-08T10:49:57-05:00",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/my-volume/_data",
        "Name": "my-volume",
        "Options": {},
        "Scope": "local"
    }
]
```

Part 3: Networking and Linking Containers

1. Objective:

- Understand Docker networking and how to link multiple containers.

2. Materials Needed:

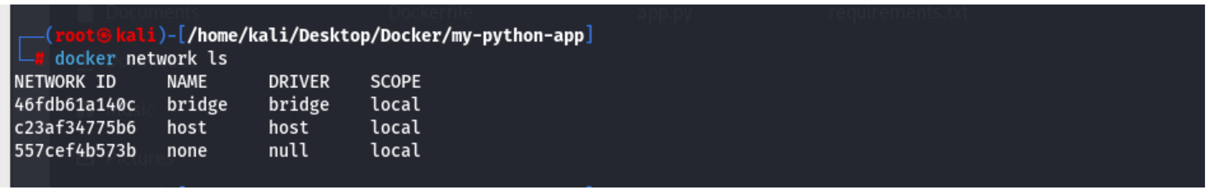
- Docker installed

3. Steps:

3.1. Docker Networking Basics

- Overview of Docker networking (bridge, host, overlay networks)

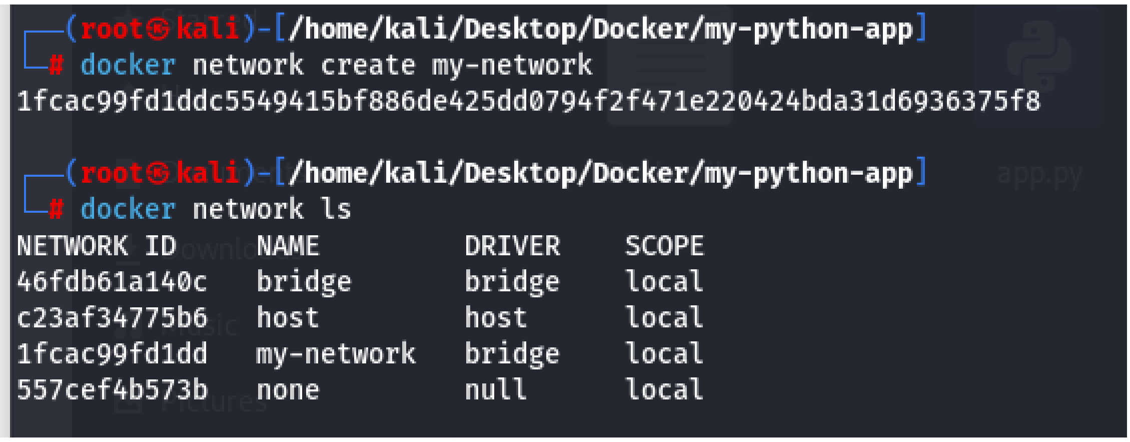
- Listing networks: `docker network ls`



```
┌──(root㉿kali)-[/home/kali/Desktop/Docker/my-python-app]
└─# docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
46fdb61a140c   bridge    bridge    local
c23af34775b6   host      host      local
557cef4b573b   none      null      local
```

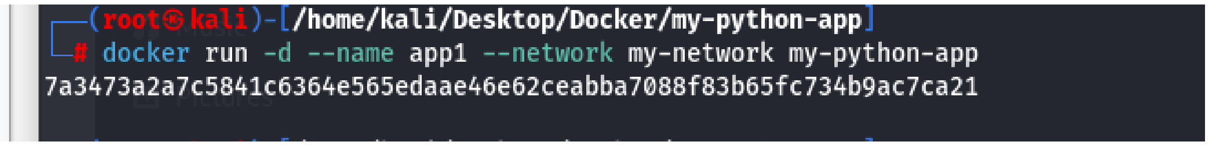
3.2. Creating a User-Defined Network

- Creating a custom bridge network: `docker network create my-network`



```
┌──(root㉿kali)-[/home/kali/Desktop/Docker/my-python-app]
└─# docker network create my-network
1fcac99fd1ddc5549415bf886de425dd0794f2f471e220424bda31d6936375f8

┌──(root㉿kali)-[/home/kali/Desktop/Docker/my-python-app]
└─# docker network ls
NETWORK ID     NAME         DRIVER    SCOPE
46fdb61a140c   bridge       bridge    local
c23af34775b6   host         host      local
1fcac99fd1dd   my-network   bridge    local
557cef4b573b   none         null      local
```

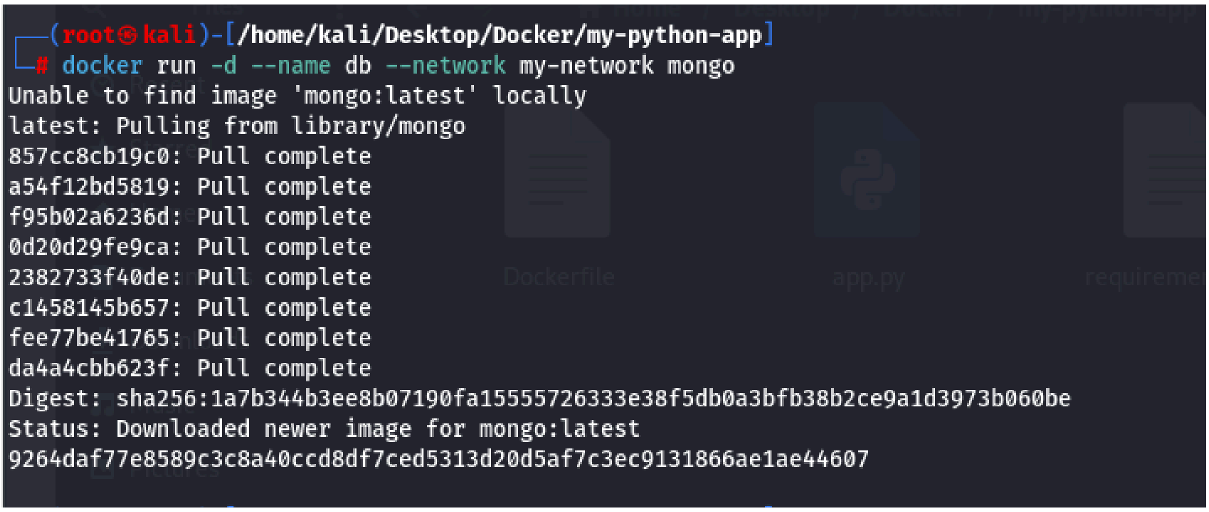
- Running containers in the custom network: `docker run -d --name app1 --network my-network my-python-app`



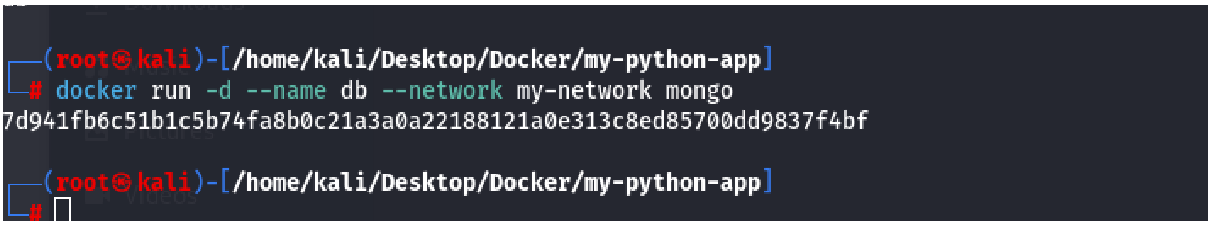
```
┌──(root㉿kali)-[/home/kali/Desktop/Docker/my-python-app]
└─# docker run -d --name app1 --network my-network my-python-app
7a3473a2a7c5841c6364e565edaae46e62ceabba7088f83b65fc734b9ac7ca21
```

3.3. Linking Containers

- Running a database container: `docker run -d --name db --network my-network mongo`



```
┌──(root㉿kali)-[/home/kali/Desktop/Docker/my-python-app]
└─# docker run -d --name db --network my-network mongo
Unable to find image 'mongo:latest' locally
latest: Pulling from library/mongo
857cc8cb19c0: Pull complete
a54f12bd5819: Pull complete
f95b02a6236d: Pull complete
0d20d29fe9ca: Pull complete
2382733f40de: Pull complete
c1458145b657: Pull complete
fee77be41765: Pull complete
da4a4cbb623f: Pull complete
Digest: sha256:1a7b344b3ee8b07190fa15555726333e38f5db0a3bfb38b2ce9a1d3973b060be
Status: Downloaded newer image for mongo:latest
9264daf77e8589c3c8a40ccd8df7ced5313d20d5af7c3ec9131866ae1ae44607
```



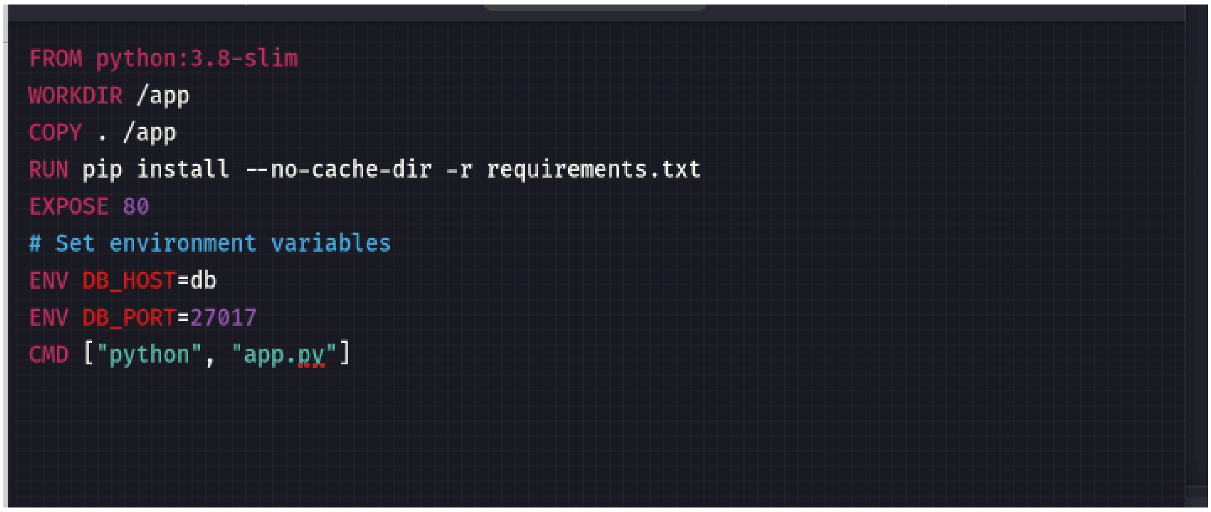
```
┌──(root㉿kali)-[/home/kali/Desktop/Docker/my-python-app]
└─# docker run -d --name db --network my-network mongo
7d941fb6c51b1c5b74fa8b0c21a3a0a22188121a0e313c8ed85700dd9837f4bf

┌──(root㉿kali)-[/home/kali/Desktop/Docker/my-python-app]
└─# 
```

- Updating the application to connect to the database using environment variables

Example Dockerfile with Environment Variables:

```Dockerfile
# Use an official Python runtime as a parent image
FROM python:3.8-slim


# Set the working directory in the container
WORKDIR /app


# Copy the current directory contents into the container at /app
COPY . /app


# Install any needed packages specified in requirements.txt
```

```
RUN pip install --no-cache-dir -r requirements.txt

# Make port 80 available to the world outside this container
EXPOSE 80

# Set environment variables
ENV DB_HOST=db
ENV DB_PORT=27017

# Run app.py when the container launches
CMD ["python", "app.py"]
```



```
FROM python:3.8-slim
WORKDIR /app
COPY . /app
RUN pip install --no-cache-dir -r requirements.txt
EXPOSE 80
# Set environment variables
ENV DB_HOST=db
ENV DB_PORT=27017
CMD ["python", "app.py"]
```

Part 4: Introduction to Docker Compose

1. Objective:

- Learn how to define and run multi-container Docker applications using Docker Compose.

2. Materials Needed:

- Docker Compose installed

3. Steps:

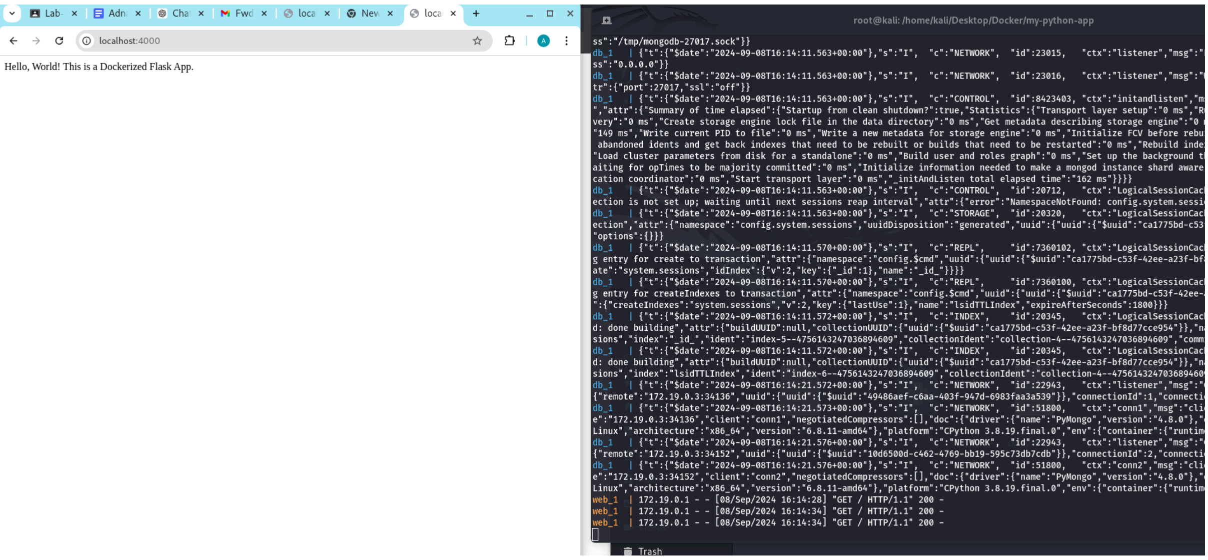
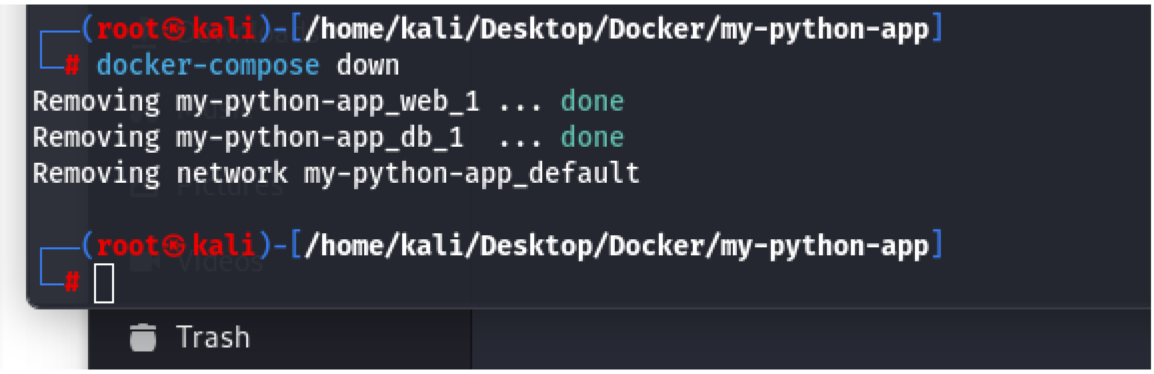
4.1. Introduction to Docker Compose

- Overview of Docker Compose and its use cases
- Installing Docker Compose

4.2. Creating a `docker-compose.yml` File

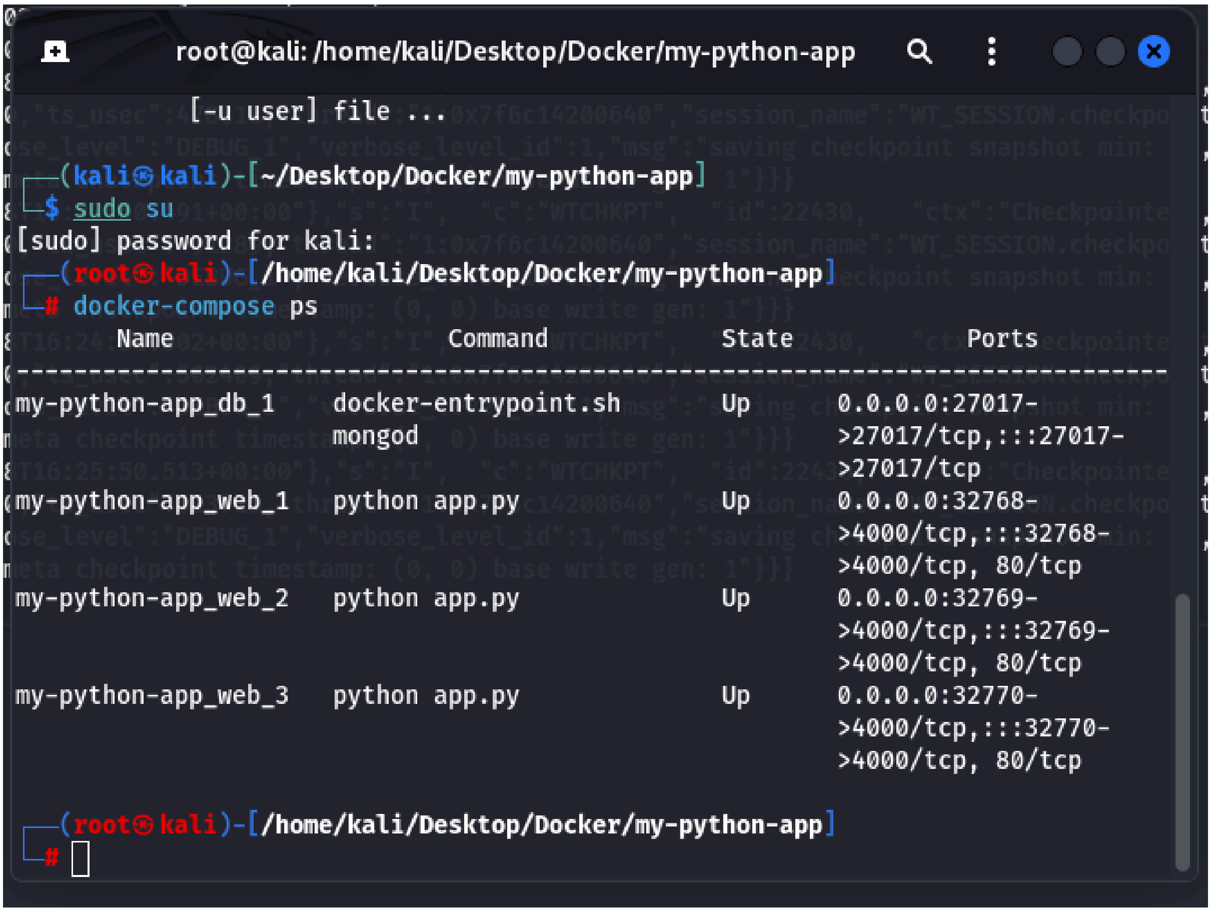
- Understanding the `docker-compose.yml` syntax
- Creating a `docker-compose.yml` for a multi-container application

Example `docker-compose.yml`:

```yaml
version: '3'
services:
  web:
    image: my-python-app
    build: .
    ports:
      - "4000:80"
    environment:
      - DB_HOST=db
      - DB_PORT=27017
  db:
    image: mongo
    ports:
      - "27017:27017"
```
```

4.3. Running Multi-Container Applications

- Starting the application: `docker-compose up`



Lab
Adn
Cha
Fwd
loca
New
loca
localhost:4000
root@kali: /home/kali/Desktop/Docker/my-python-app
Trash

Hello, World! This is a Dockerized Flask App.

```
ss":"/tmp/mongodb-27017.sock"}}
db_1    | {"t":{"$date":"2024-09-08T16:14:11.563+00:00"},"s":"I",  "c":"NETWORK",  "id":23015,   "ctx":"listener","msg":"
ss":"0.0.0.0"}}
db_1    | {"t":{"$date":"2024-09-08T16:14:11.563+00:00"},"s":"I",  "c":"NETWORK",  "id":23016,   "ctx":"listener","msg":"
tr":{"port":27017,"ssl":"off"}}
db_1    | {"t":{"$date":"2024-09-08T16:14:11.563+00:00"},"s":"I",  "c":"CONTROL",  "id":8423403, "ctx":"initandlisten","m
","attr":{"Summary of time elapsed":{"Startup from clean shutdown?":true,"Statistics":{"Transport layer setup":"0 ms","R
very":"0 ms","Create storage engine lock file in the data directory":"0 ms","Get metadata describing storage engine":"0
"149 ms","Write current PID to file":"0 ms","Write a new metadata for storage engine":"0 ms","Initialize FCV before rebu
 abandoned idents and get back indexes that need to be rebuilt or builds that need to be restarted":"0 ms","Rebuild inde
"Load cluster parameters from disk for a standalone":"0 ms","Build user and roles graph":"0 ms","Set up the background t
aiting for opTimes to be majority committed":"0 ms","Initialize information needed to make a mongod instance shard aware
cation coordinator":"0 ms","Start transport layer":"0 ms","_initAndListen total elapsed time":"162 ms"}}}}
db_1    | {"t":{"$date":"2024-09-08T16:14:11.563+00:00"},"s":"I",  "c":"CONTROL",  "id":20712,   "ctx":"LogicalSessionCac
ection is not set up; waiting until next sessions reap interval","attr":{"error":"NamespaceNotFound: config.system.sessi
db_1    | {"t":{"$date":"2024-09-08T16:14:11.563+00:00"},"s":"I",  "c":"STORAGE",  "id":20320,   "ctx":"LogicalSessionCac
ection","attr":{"namespace":"config.system.sessions","uuidDisposition":"generated","uuid":{"uuid":{"$uuid":"ca1775bd-c53
"options":{}}}
db_1    | {"t":{"$date":"2024-09-08T16:14:11.570+00:00"},"s":"I",  "c":"REPL",     "id":7360102, "ctx":"LogicalSessionCac
g entry for create to transaction","attr":{"namespace":"config.$cmd","uuid":{"uuid":{"$uuid":"ca1775bd-c53f-42ee-a23f-bf
ate":"system.sessions","idIndex":{"v":2,"key":{"_id":1},"name":"_id_"}}}}
db_1    | {"t":{"$date":"2024-09-08T16:14:11.570+00:00"},"s":"I",  "c":"REPL",     "id":7360100, "ctx":"LogicalSessionCac
g entry for createIndexes to transaction","attr":{"namespace":"config.$cmd","uuid":{"uuid":{"$uuid":"ca1775bd-c53f-42ee-
":{"createIndexes":"system.sessions","v":2,"key":{"lastUse":1},"name":"lsidTTLIndex","expireAfterSeconds":1800}}}
db_1    | {"t":{"$date":"2024-09-08T16:14:11.572+00:00"},"s":"I",  "c":"INDEX",    "id":20345,   "ctx":"LogicalSessionCac
d: done building","attr":{"buildUUID":null,"collectionUUID":{"uuid":{"$uuid":"ca1775bd-c53f-42ee-a23f-bf8d77cce954"}},"n
sions","index":"_id_","ident":"index-5--4756143247036894609","collectionIdent":"collection-4--4756143247036894609","comm
db_1    | {"t":{"$date":"2024-09-08T16:14:11.572+00:00"},"s":"I",  "c":"INDEX",    "id":20345,   "ctx":"LogicalSessionCac
d: done building","attr":{"buildUUID":null,"collectionUUID":{"uuid":{"$uuid":"ca1775bd-c53f-42ee-a23f-bf8d77cce954"}},"n
sions","index":"lsidTTLIndex","ident":"index-6--4756143247036894609","collectionIdent":"collection-4--475614324703689460
db_1    | {"t":{"$date":"2024-09-08T16:14:21.572+00:00"},"s":"I",  "c":"NETWORK",  "id":22943,   "ctx":"listener","msg":"
{"remote":"172.19.0.3:34136","uuid":{"uuid":{"$uuid":"49486aef-c6aa-403f-947d-6983faa3a539"}},"connectionId":1,"connecti
db_1    | {"t":{"$date":"2024-09-08T16:14:21.573+00:00"},"s":"I",  "c":"NETWORK",  "id":51800,   "ctx":"conn1","msg":"cli
e":"172.19.0.3:34136","client":"conn1","negotiatedCompressors":[],"doc":{"driver":{"name":"PyMongo","version":"4.8.0"},"
Linux","architecture":"x86_64","version":"6.8.11-amd64"},"platform":"CPython 3.8.19.final.0","env":{"container":{"runtim
db_1    | {"t":{"$date":"2024-09-08T16:14:21.576+00:00"},"s":"I",  "c":"NETWORK",  "id":22943,   "ctx":"listener","msg":"
{"remote":"172.19.0.3:34152","uuid":{"uuid":{"$uuid":"10d6500d-c462-4769-bb19-595c73db7cdb"}},"connectionId":2,"connecti
db_1    | {"t":{"$date":"2024-09-08T16:14:21.576+00:00"},"s":"I",  "c":"NETWORK",  "id":51800,   "ctx":"conn2","msg":"cli
e":"172.19.0.3:34152","client":"conn2","negotiatedCompressors":[],"doc":{"driver":{"name":"PyMongo","version":"4.8.0"},"
Linux","architecture":"x86_64","version":"6.8.11-amd64"},"platform":"CPython 3.8.19.final.0","env":{"container":{"runtim
web_1   | 172.19.0.1 - - [08/Sep/2024 16:14:28] "GET / HTTP/1.1" 200 -
web_1   | 172.19.0.1 - - [08/Sep/2024 16:14:34] "GET / HTTP/1.1" 200 -
web_1   | 172.19.0.1 - - [08/Sep/2024 16:14:34] "GET / HTTP/1.1" 200 -
```

- Stopping the application: `docker-compose down`



Trash

```
┌──(root㉿kali)-[/home/kali/Desktop/Docker/my-python-app]
└─# docker-compose down
Removing my-python-app_web_1 ... done
Removing my-python-app_db_1  ... done
Removing network my-python-app_default

┌──(root㉿kali)-[/home/kali/Desktop/Docker/my-python-app]
└─# 
```
```

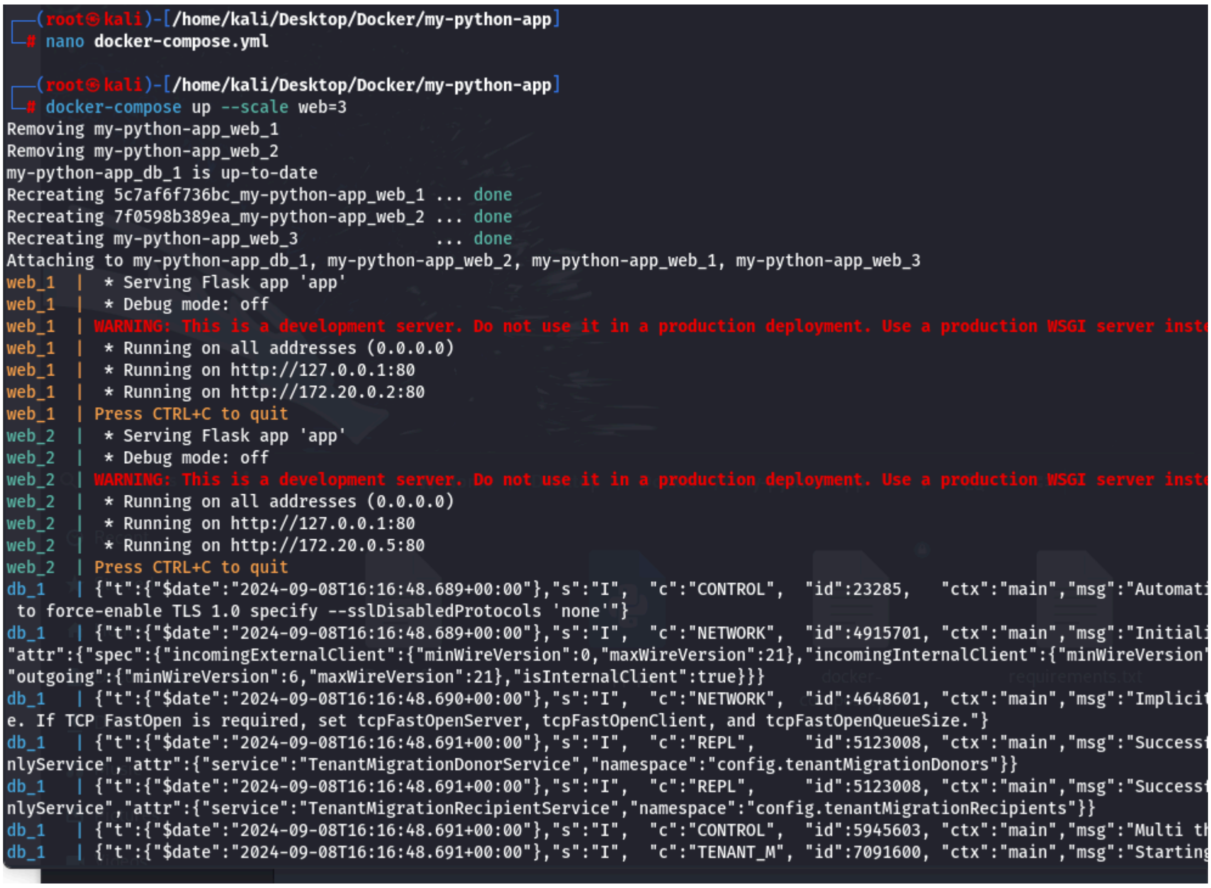
- Listing running services: `docker-compose ps`



```
            [-u user] file ...
┌──(kali㉿kali)-[~/Desktop/Docker/my-python-app]
└─$ sudo su
[sudo] password for kali:
┌──(root㉿kali)-[/home/kali/Desktop/Docker/my-python-app]
└─# docker-compose ps
       Name                 Command          State          Ports
----------------------------------------------------------------------------------
my-python-app_db_1    docker-entrypoint.sh    Up      0.0.0.0:27017-
                      mongod                          >27017/tcp,:::27017-
                                                      >27017/tcp
my-python-app_web_1   python app.py           Up      0.0.0.0:32768-
                                                      >4000/tcp,:::32768-
                                                      >4000/tcp, 80/tcp
my-python-app_web_2   python app.py           Up      0.0.0.0:32769-
                                                      >4000/tcp,:::32769-
                                                      >4000/tcp, 80/tcp
my-python-app_web_3   python app.py           Up      0.0.0.0:32770-
                                                      >4000/tcp,:::32770-
                                                      >4000/tcp, 80/tcp

┌──(root㉿kali)-[/home/kali/Desktop/Docker/my-python-app]
└─#
```

4.4. Managing Docker Compose Applications

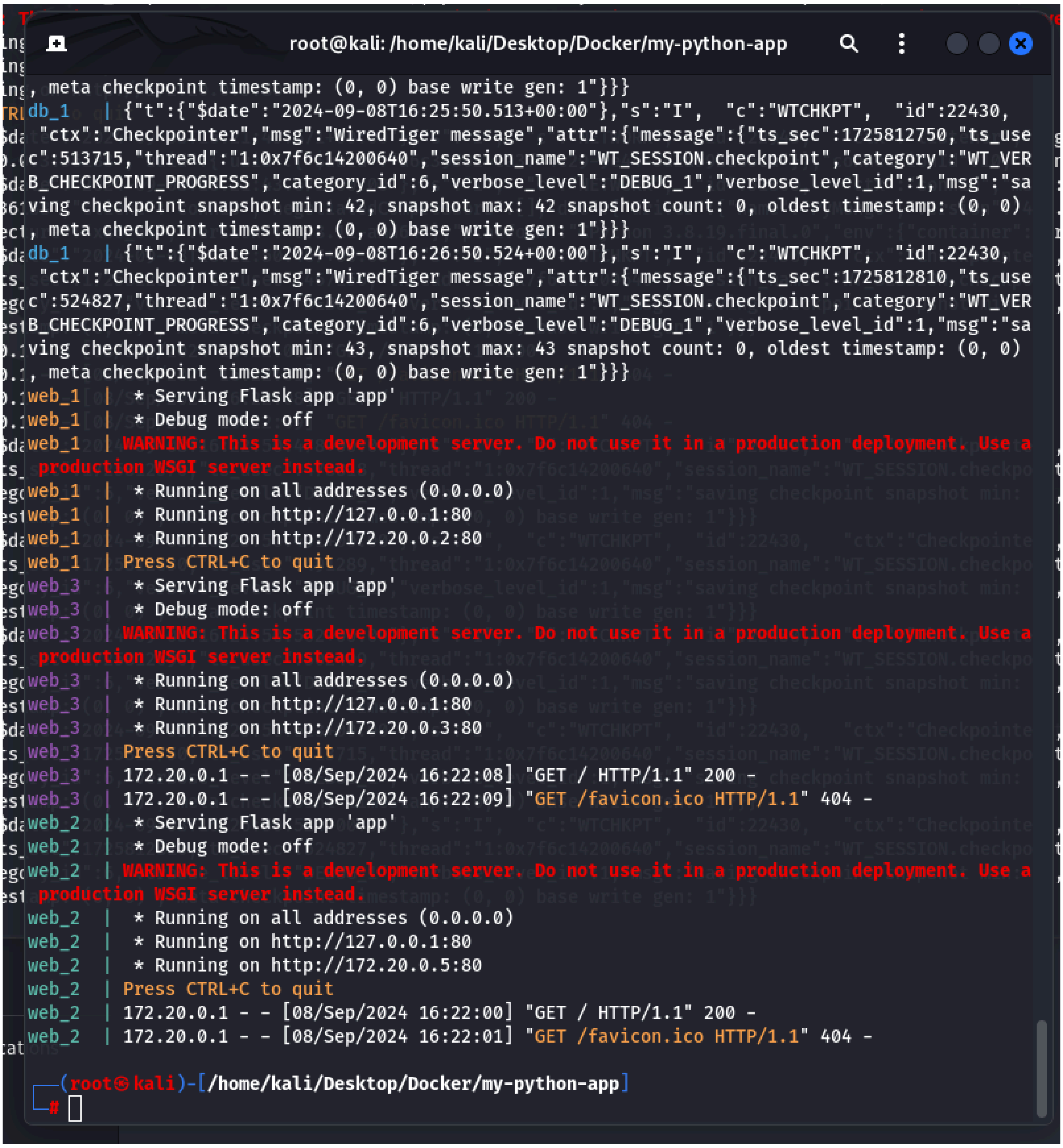
- Scaling services: `docker-compose up --scale web=3`



```
┌──(root㉿kali)-[/home/kali/Desktop/Docker/my-python-app]
└─# nano docker-compose.yml

┌──(root㉿kali)-[/home/kali/Desktop/Docker/my-python-app]
└─# docker-compose up --scale web=3
Removing my-python-app_web_1
Removing my-python-app_web_2
my-python-app_db_1 is up-to-date
Recreating 5c7af6f736bc_my-python-app_web_1 ... done
Recreating 7f0598b389ea_my-python-app_web_2 ... done
Recreating my-python-app_web_3              ... done
Attaching to my-python-app_db_1, my-python-app_web_2, my-python-app_web_1, my-python-app_web_3
web_1  |  * Serving Flask app 'app'
web_1  |  * Debug mode: off
web_1  | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server inste
web_1  |  * Running on all addresses (0.0.0.0)
web_1  |  * Running on http://127.0.0.1:80
web_1  |  * Running on http://172.20.0.2:80
web_1  | Press CTRL+C to quit
web_2  |  * Serving Flask app 'app'
web_2  |  * Debug mode: off
web_2  | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server inste
web_2  |  * Running on all addresses (0.0.0.0)
web_2  |  * Running on http://127.0.0.1:80
web_2  |  * Running on http://172.20.0.5:80
web_2  | Press CTRL+C to quit
db_1   | {"t":{"$date":"2024-09-08T16:16:48.689+00:00"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"main","msg":"Automati
 to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
db_1   | {"t":{"$date":"2024-09-08T16:16:48.689+00:00"},"s":"I",  "c":"NETWORK",  "id":4915701, "ctx":"main","msg":"Initiali
"attr":{"spec":{"incomingExternalClient":{"minWireVersion":0,"maxWireVersion":21},"incomingInternalClient":{"minWireVersion"
"outgoing":{"minWireVersion":6,"maxWireVersion":21},"isInternalClient":true}}}
db_1   | {"t":{"$date":"2024-09-08T16:16:48.690+00:00"},"s":"I",  "c":"NETWORK",  "id":4648601, "ctx":"main","msg":"Implicit
e. If TCP FastOpen is required, set tcpFastOpenServer, tcpFastOpenClient, and tcpFastOpenQueueSize."}
db_1   | {"t":{"$date":"2024-09-08T16:16:48.691+00:00"},"s":"I",  "c":"REPL",     "id":5123008, "ctx":"main","msg":"Successf
nlyService","attr":{"service":"TenantMigrationDonorService","namespace":"config.tenantMigrationDonors"}}
db_1   | {"t":{"$date":"2024-09-08T16:16:48.691+00:00"},"s":"I",  "c":"REPL",     "id":5123008, "ctx":"main","msg":"Successf
nlyService","attr":{"service":"TenantMigrationRecipientService","namespace":"config.tenantMigrationRecipients"}}
db_1   | {"t":{"$date":"2024-09-08T16:16:48.691+00:00"},"s":"I",  "c":"CONTROL",  "id":5945603, "ctx":"main","msg":"Multi th
db_1   | {"t":{"$date":"2024-09-08T16:16:48.691+00:00"},"s":"I",  "c":"TENANT_M", "id":7091600, "ctx":"main","msg":"Starting
```

c

- Viewing logs: `docker-compose logs`



```
, meta checkpoint timestamp: (0, 0) base write gen: 1"}}}
db_1    | {"t":{"$date":"2024-09-08T16:25:50.513+00:00"},"s":"I",  "c":"WTCHKPT",  "id":22430,
 "ctx":"Checkpointer","msg":"WiredTiger message","attr":{"message":{"ts_sec":1725812750,"ts_use
c":513715,"thread":"1:0x7f6c14200640","session_name":"WT_SESSION.checkpoint","category":"WT_VER
B_CHECKPOINT_PROGRESS","category_id":6,"verbose_level":"DEBUG_1","verbose_level_id":1,"msg":"sa
ving checkpoint snapshot min: 42, snapshot max: 42 snapshot count: 0, oldest timestamp: (0, 0)
, meta checkpoint timestamp: (0, 0) base write gen: 1"}}}
db_1    | {"t":{"$date":"2024-09-08T16:26:50.524+00:00"},"s":"I",  "c":"WTCHKPT",  "id":22430,
 "ctx":"Checkpointer","msg":"WiredTiger message","attr":{"message":{"ts_sec":1725812810,"ts_use
c":524827,"thread":"1:0x7f6c14200640","session_name":"WT_SESSION.checkpoint","category":"WT_VER
B_CHECKPOINT_PROGRESS","category_id":6,"verbose_level":"DEBUG_1","verbose_level_id":1,"msg":"sa
ving checkpoint snapshot min: 43, snapshot max: 43 snapshot count: 0, oldest timestamp: (0, 0)
, meta checkpoint timestamp: (0, 0) base write gen: 1"}}}
web_1   |  * Serving Flask app 'app'
web_1   |  * Debug mode: off
web_1   | WARNING: This is a development server. Do not use it in a production deployment. Use a
 production WSGI server instead.
web_1   |  * Running on all addresses (0.0.0.0)
web_1   |  * Running on http://127.0.0.1:80
web_1   |  * Running on http://172.20.0.2:80
web_1   | Press CTRL+C to quit
web_3   |  * Serving Flask app 'app'
web_3   |  * Debug mode: off
web_3   | WARNING: This is a development server. Do not use it in a production deployment. Use a
 production WSGI server instead.
web_3   |  * Running on all addresses (0.0.0.0)
web_3   |  * Running on http://127.0.0.1:80
web_3   |  * Running on http://172.20.0.3:80
web_3   | Press CTRL+C to quit
web_3   | 172.20.0.1 - - [08/Sep/2024 16:22:08] "GET / HTTP/1.1" 200 -
web_3   | 172.20.0.1 - - [08/Sep/2024 16:22:09] "GET /favicon.ico HTTP/1.1" 404 -
web_2   |  * Serving Flask app 'app'
web_2   |  * Debug mode: off
web_2   | WARNING: This is a development server. Do not use it in a production deployment. Use a
 production WSGI server instead.
web_2   |  * Running on all addresses (0.0.0.0)
web_2   |  * Running on http://127.0.0.1:80
web_2   |  * Running on http://172.20.0.5:80
web_2   | Press CTRL+C to quit
web_2   | 172.20.0.1 - - [08/Sep/2024 16:22:00] "GET / HTTP/1.1" 200 -
web_2   | 172.20.0.1 - - [08/Sep/2024 16:22:01] "GET /favicon.ico HTTP/1.1" 404 -

┌──(root㉿kali)-[/home/kali/Desktop/Docker/my-python-app]
└─#
```

- Accessing the web application in a browser: `http://localhost:4000`

Part 5: Advanced Docker Compose Features

1. Objective:

   - Explore advanced features of Docker Compose such as volumes, networks, and environment variables.

2. Materials Needed:

   - Docker Compose installed

3. Steps:

   5.1. Using Volumes in Docker Compose

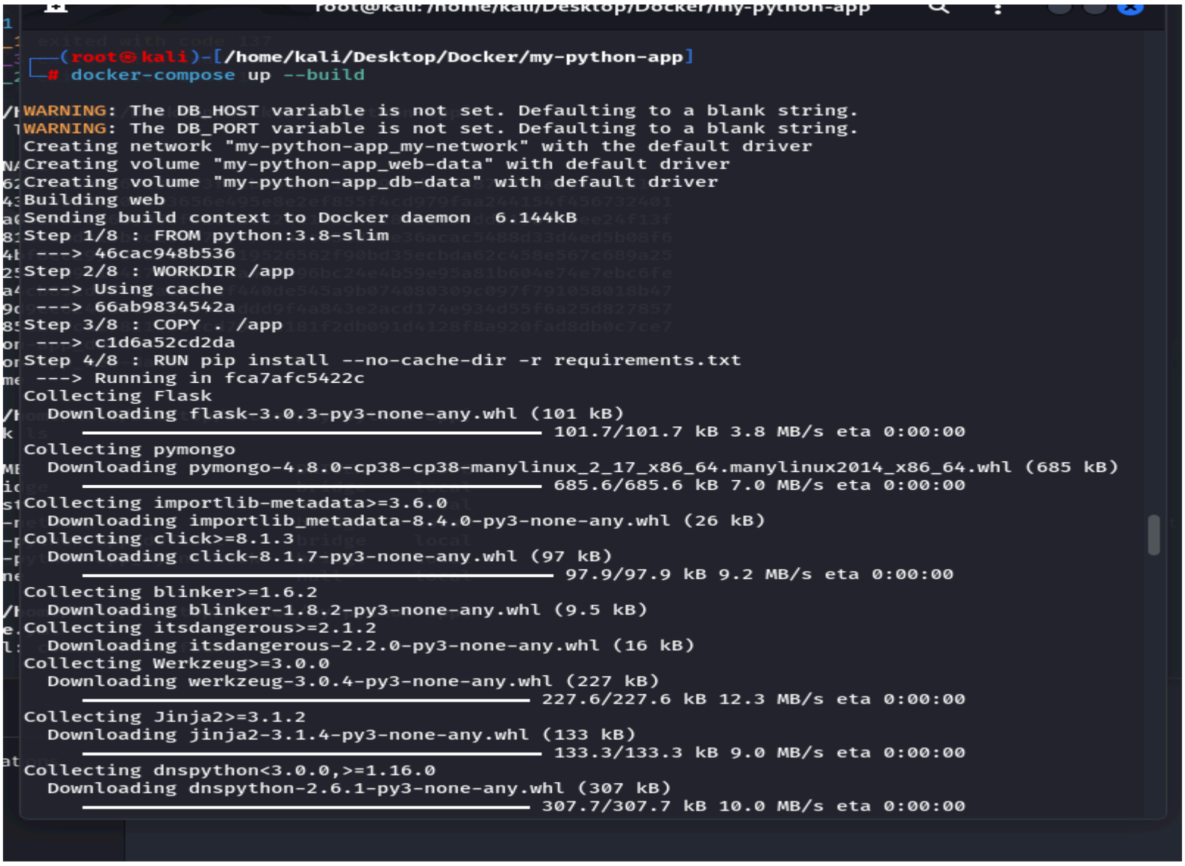
   - Defining volumes in `docker-compose.yml`

   - Mounting volumes for persistent data storage

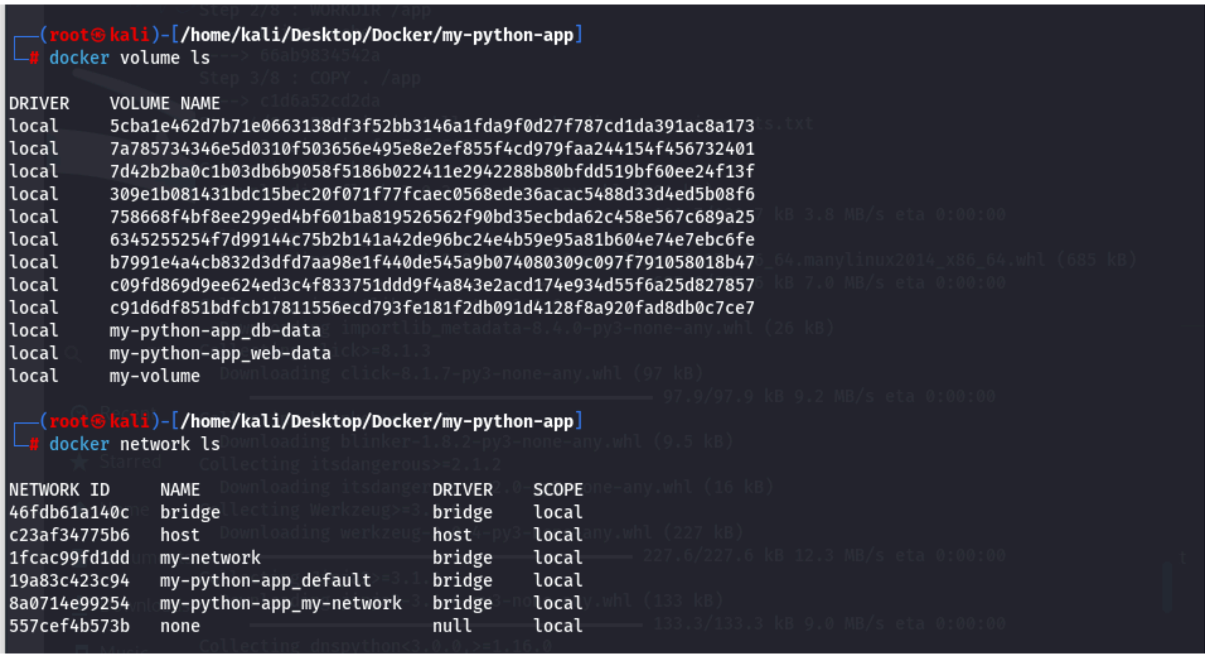
   Example `docker-compose.yml` with Volumes:

```yaml
version: '3'
services:
  web:
    image: my-python-app
    build: .
    ports:
      - "4000:80"
    environment:
      - DB_HOST=db
      - DB_PORT=27017
    volumes:
      - web-data:/app
  db:
    image: mongo
    ports:
      - "27017:27017"
    volumes:
      - db-data:/data/db
volumes:
  web-data:
  db-data:
```

1T33: Cloud Architecture



```
┌──(root㉿kali)-[/home/kali/Desktop/Docker/my-python-app]
└─# docker-compose up --build

WARNING: The DB_HOST variable is not set. Defaulting to a blank string.
WARNING: The DB_PORT variable is not set. Defaulting to a blank string.
Creating network "my-python-app_my-network" with the default driver
Creating volume "my-python-app_web-data" with default driver
Creating volume "my-python-app_db-data" with default driver
Building web
Sending build context to Docker daemon  6.144kB
Step 1/8 : FROM python:3.8-slim
 ---> 46cac948b536
Step 2/8 : WORKDIR /app
 ---> Using cache
 ---> 66ab9834542a
Step 3/8 : COPY . /app
 ---> c1d6a52cd2da
Step 4/8 : RUN pip install --no-cache-dir -r requirements.txt
 ---> Running in fca7afc5422c
Collecting Flask
  Downloading flask-3.0.3-py3-none-any.whl (101 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 101.7/101.7 kB 3.8 MB/s eta 0:00:00
Collecting pymongo
  Downloading pymongo-4.8.0-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (685 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 685.6/685.6 kB 7.0 MB/s eta 0:00:00
Collecting importlib-metadata>=3.6.0
  Downloading importlib_metadata-8.4.0-py3-none-any.whl (26 kB)
Collecting click>=8.1.3
  Downloading click-8.1.7-py3-none-any.whl (97 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 97.9/97.9 kB 9.2 MB/s eta 0:00:00
Collecting blinker>=1.6.2
  Downloading blinker-1.8.2-py3-none-any.whl (9.5 kB)
Collecting itsdangerous>=2.1.2
  Downloading itsdangerous-2.2.0-py3-none-any.whl (16 kB)
Collecting Werkzeug>=3.0.0
  Downloading werkzeug-3.0.4-py3-none-any.whl (227 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 227.6/227.6 kB 12.3 MB/s eta 0:00:00
Collecting Jinja2>=3.1.2
  Downloading jinja2-3.1.4-py3-none-any.whl (133 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 133.3/133.3 kB 9.0 MB/s eta 0:00:00
Collecting dnspython<3.0.0,>=1.16.0
  Downloading dnspython-2.6.1-py3-none-any.whl (307 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 307.7/307.7 kB 10.0 MB/s eta 0:00:00
```



```
┌──(root㉿kali)-[/home/kali/Desktop/Docker/my-python-app]
└─# docker volume ls

DRIVER    VOLUME NAME
local     5cba1e462d7b71e0663138df3f52bb3146a1fda9f0d27f787cd1da391ac8a173
local     7a785734346e5d0310f503656e495e8e2ef855f4cd979faa244154f456732401
local     7d42b2ba0c1b03db6b9058f5186b022411e2942288b80bfdd519bf60ee24f13f
local     309e1b081431bdc15bec20f071f77fcaec0568ede36acac5488d33d4ed5b08f6
local     758668f4bf8ee299ed4bf601ba819526562f90bd35ecbda62c458e567c689a25
local     6345255254f7d99144c75b2b141a42de96bc24e4b59e95a81b604e74e7ebc6fe
local     b7991e4a4cb832d3dfd7aa98e1f440de545a9b074080309c097f791058018b47
local     c09fd869d9ee624ed3c4f833751ddd9f4a843e2acd174e934d55f6a25d827857
local     c91d6df851bdfcb17811556ecd793fe181f2db091d4128f8a920fad8db0c7ce7
local     my-python-app_db-data
local     my-python-app_web-data
local     my-volume

┌──(root㉿kali)-[/home/kali/Desktop/Docker/my-python-app]
└─# docker network ls

NETWORK ID     NAME                       DRIVER    SCOPE
46fdb61a140c   bridge                     bridge    local
c23af34775b6   host                       host      local
1fcac99fd1dd   my-network                 bridge    local
19a83c423c94   my-python-app_default      bridge    local
8a0714e99254   my-python-app_my-network   bridge    local
557cef4b573b   none                       null      local
```

5.2. Using Networks in Docker Compose

- Defining custom networks in `docker-compose.yml`

- Connecting services to custom networks

Example `docker-compose.yml` with Networks:

```yaml
version: '3'
services:
  web:
    image: my-python-app
    build: .
    ports:
      - "4000:80"
    environment:
      - DB_HOST=db
      - DB_PORT=27017
    networks:
      - my-network
  db:
    image: mongo
    ports:
      - "27017:27017"
    networks:
      - my-network
networks:
  my-network:
```

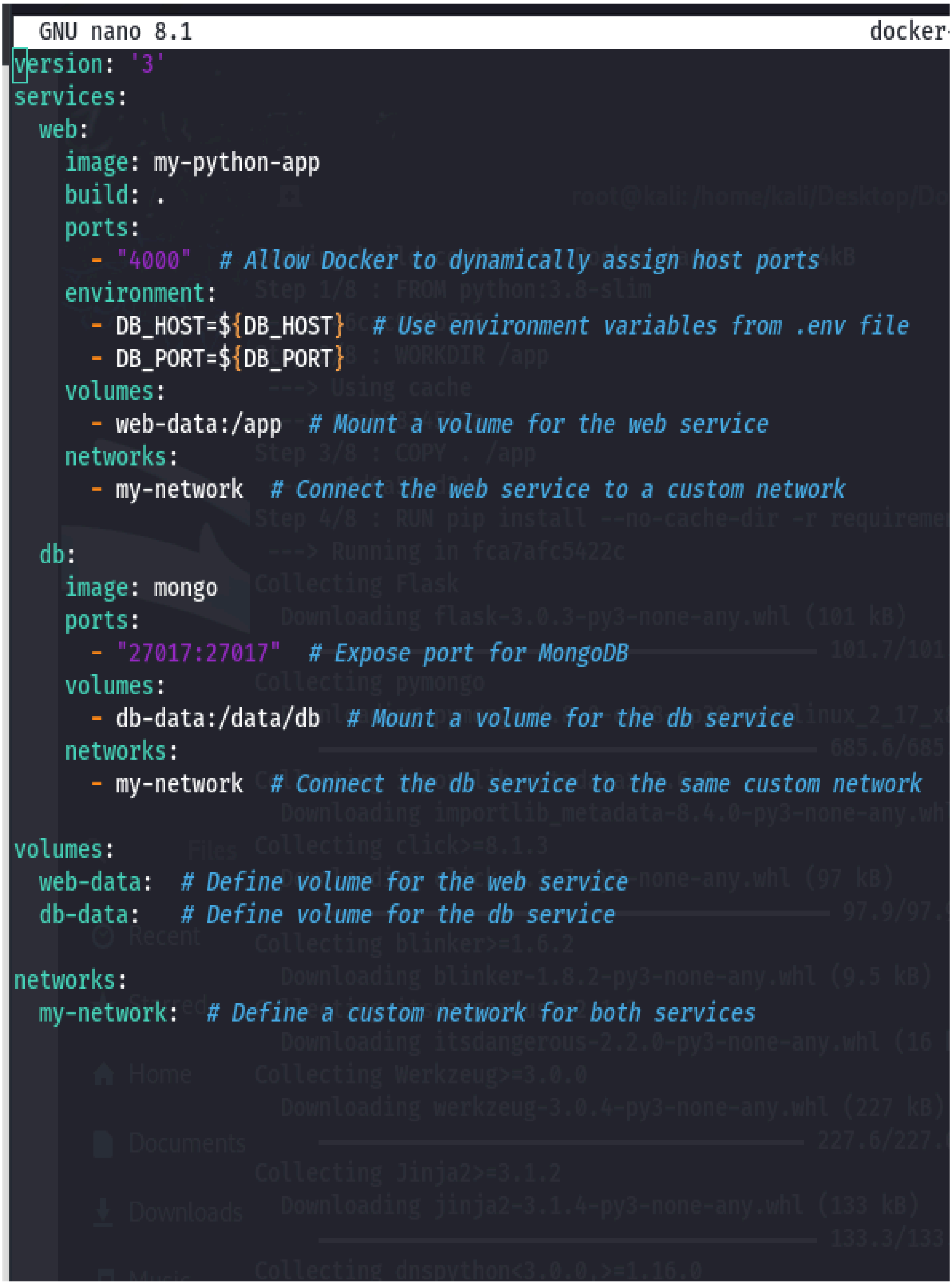
5.3. Using Environment Variables in Docker Compose

- Defining environment variables in `docker-compose.yml`

- Using `.env` files for environment-specific configurations

Example `.env` File:

```
DB_HOST=db
DB_PORT=27017
```



```
GNU nano 8.1                                                              docker-
version: '3'
services:
  web:
    image: my-python-app
    build: .
    ports:
      - "4000"  # Allow Docker to dynamically assign host ports
    environment:
      - DB_HOST=${DB_HOST}  # Use environment variables from .env file
      - DB_PORT=${DB_PORT}
    volumes:
      - web-data:/app  # Mount a volume for the web service
    networks:
      - my-network  # Connect the web service to a custom network

  db:
    image: mongo
    ports:
      - "27017:27017"  # Expose port for MongoDB
    volumes:
      - db-data:/data/db  # Mount a volume for the db service
    networks:
      - my-network  # Connect the db service to the same custom network

volumes:
  web-data:  # Define volume for the web service
  db-data:   # Define volume for the db service

networks:
  my-network:  # Define a custom network for both services
```

Part 6: Best Practices and Troubleshooting

1. Objective:

   - Learn best practices for using Docker and Docker Compose and troubleshoot common issues.

2. Materials Needed:

   - Docker and Docker Compose installed

3. Steps:

   6.1. Docker Best Practices

   - Writing efficient Dockerfiles

   - Managing images and containers effectively

   - Security best practices (e.g., using non-root users, minimizing image size)

   6.2. Docker Compose Best Practices

   - Structuring `docker-compose.yml` files

   - Using multiple Compose files for different environments (e.g., `docker-compose.override.yml`)

   - Managing secrets and environment variables securely

   6.3. Troubleshooting Common Issues

   - Debugging Dockerfile issues

   - Resolving container startup failures

   - Networking issues and how to diagnose them

   - Using Docker and Docker Compose logs for troubleshooting

Final Project: Build a Complete Application

1. Objective:

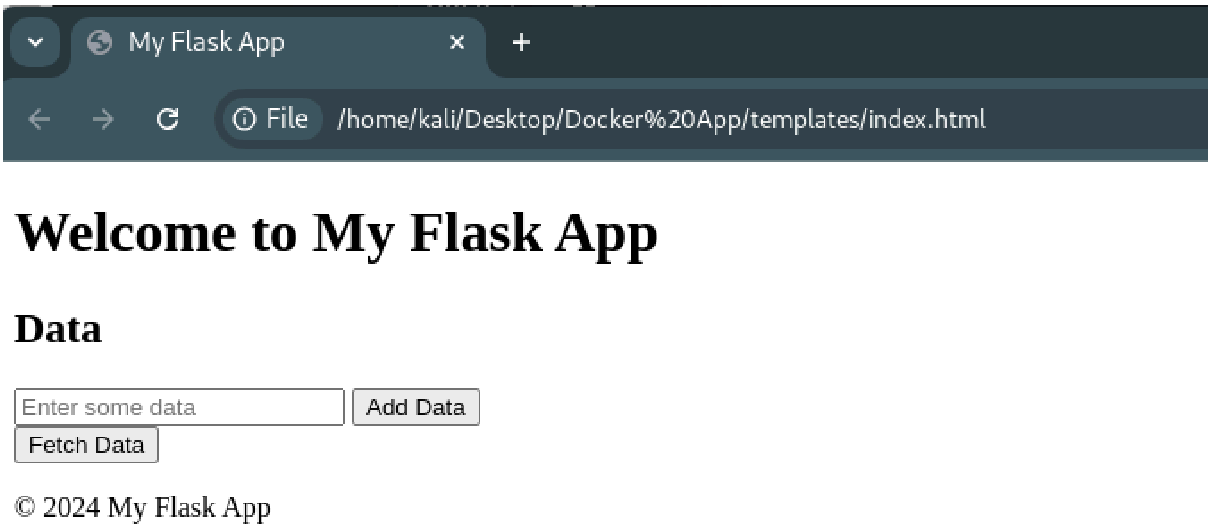
   - Apply the knowledge gained in a real-world scenario by building a complete multi-container application.

2. Materials Needed:

- Docker and Docker Compose installed

3. Project Steps:

- Define the application architecture (e.g., a web application with a database and a cache)
- Create Docker files for each component
- Write a `docker-compose.yml` file to orchestrate the multi-container setup
- Implement the application logic
- Test and debug the application
- Deploy the application using Docker Compose



Welcome to My Flask App

Data

© 2024 My Flask App

This detailed step-by-step guide should help in creating an exhaustive laboratory on Docker and Docker Compose. Each part builds on the previous one, ensuring a comprehensive understanding of Docker and its capabilities.

Instructions with Screenshots and Captions:

Conclusion:

References:

1. Official Docker Documentation:
   - [Docker Overview](https://docs.docker.com/get-started/overview/)
   - [Docker Engine Installation](https://docs.docker.com/engine/install/)
   - [Docker CLI Reference](https://docs.docker.com/engine/reference/commandline/docker/)
   - [Dockerfile Reference](https://docs.docker.com/engine/reference/builder/)
   - [Docker Compose Documentation](https://docs.docker.com/compose/)
2. Books:
   - Turnbull, J. (2018). *The Docker Book: Containerization is the new virtualization*. Turnbull Press.
   - Matthias, K., & Kane, S. (2015). *Docker: Up & Running*. O'Reilly Media.

3. Online Courses and Tutorials:
   - [Docker for Beginners](https://www.coursera.org/learn/docker)
   - [Introduction to Docker](https://www.edx.org/course/introduction-to-docker)
   - [Docker Essentials: A Developer Introduction](https://cognitiveclass.ai/courses/docker-essentials)
4. Community Resources:
   - [Docker Community Forums](https://forums.docker.com/)
   - [Stack Overflow Docker Tag](https://stackoverflow.com/questions/tagged/docker)
   - [Docker Subreddit](https://www.reddit.com/r/docker/)
5. Blog Posts and Articles:

- [Understanding Docker Containers and Images](https://www.redhat.com/en/topics/containers/what-is-a-linux-container)
- [Docker Networking Basics](https://www.digitalocean.com/community/tutorials/an-introduction-to-docker-networking-physical-hosts-containers-and-more)
- [Best Practices for Writing Dockerfiles](https://docs.docker.com/develop/develop-images/dockerfile_best-practices/)

6. Conferences and Talks:
   - DockerCon (Annual Docker Conference)
   - Various talks and webinars available on [YouTube](https://www.youtube.com/user/dockerrun)

By consulting these references, participants can deepen their understanding of Docker and Docker Compose, stay updated with the latest developments, and continue to enhance their containerization skills.