Lesson 07 Demo 02

Configuring Docker Networking for a Microservices Environment

Objective: To configure Docker networking for a microservices environment for seamless communication and scalability among microservices within Docker containers

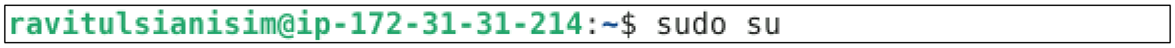
Tools required: Ubuntu

Prerequisites: None

Steps to be followed:
1. Set up a simple HTTP server with Docker
2. Set up a client script with Docker
3. Configure Docker networking for microservices

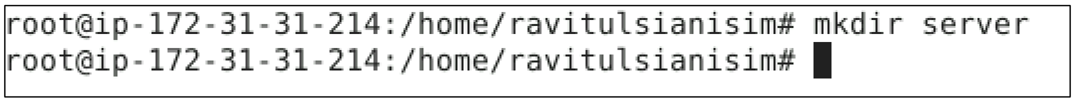
Step 1: Set up a simple HTTP server with Docker

1.1 Run the following command to switch to the root user:
sudo su



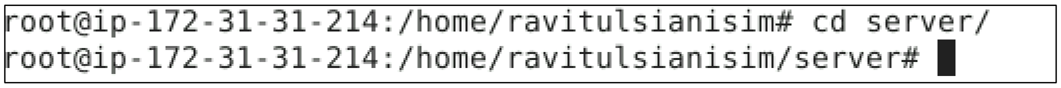
```
ravitulsianisim@ip-172-31-31-214:~$ sudo su
```

1.2 Create a directory using the following command:
mkdir server



```
root@ip-172-31-31-214:/home/ravitulsianisim# mkdir server
root@ip-172-31-31-214:/home/ravitulsianisim#
```

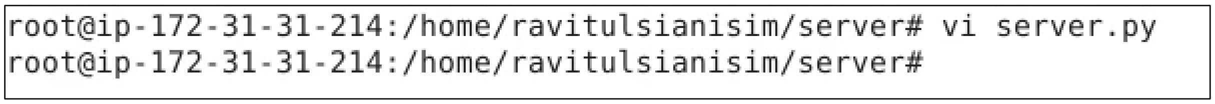
1.3 Navigate inside the directory using the following command:
cd server/



```
root@ip-172-31-31-214:/home/ravitulsianisim# cd server/
root@ip-172-31-31-214:/home/ravitulsianisim/server#
```

1.4 Create a Python file using the following command:

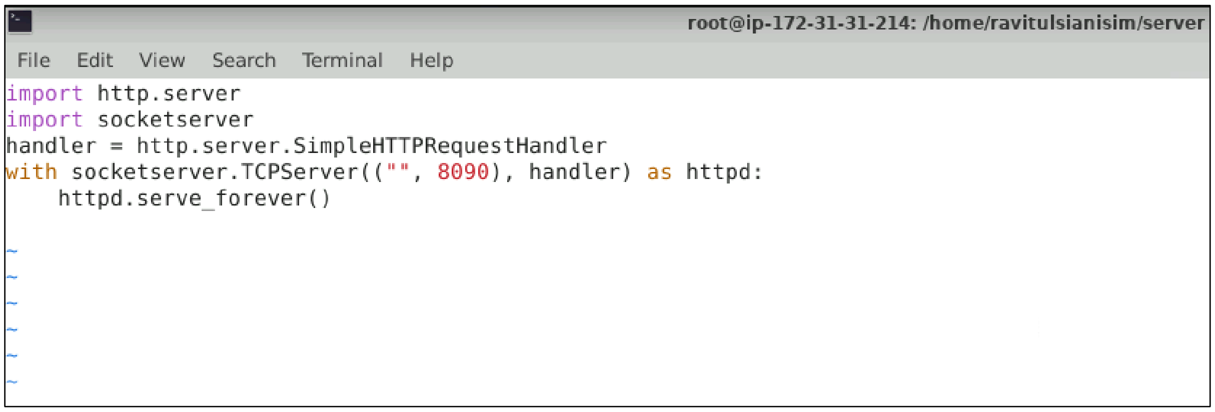
vi server.py



```
root@ip-172-31-31-214:/home/ravitulsianisim/server# vi server.py
root@ip-172-31-31-214:/home/ravitulsianisim/server#
```

1.5 Enter the following script inside the **server.py** file:

import http.server
import socketserver
handler = http.server.SimpleHTTPRequestHandler
with socketserver.TCPServer(("", 8090), handler) as httpd:
  httpd.serve_forever()

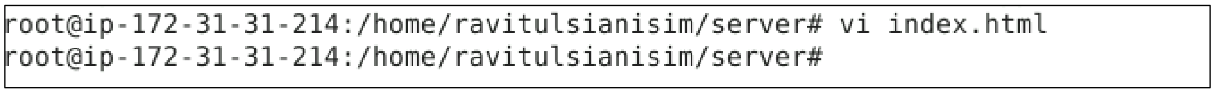

root@ip-172-31-31-214: /home/ravitulsianisim/server
File Edit View Search Terminal Help

```
import http.server
import socketserver
handler = http.server.SimpleHTTPRequestHandler
with socketserver.TCPServer(("", 8090), handler) as httpd:
    httpd.serve_forever()

~
~
~
~
~
~
```

This script creates a simple HTTP server that listens on port 8090 and serves requests indefinitely using the built-in **http.server** module.

1.6 Create a file using the following command:

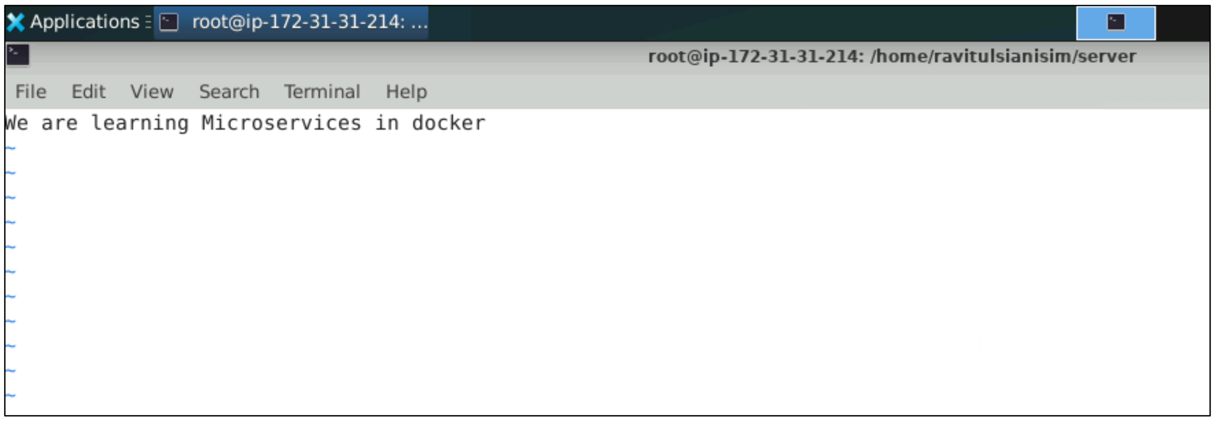
vi index.html



```
root@ip-172-31-31-214:/home/ravitulsianisim/server# vi index.html
root@ip-172-31-31-214:/home/ravitulsianisim/server#
```

1.7 Enter the data inside the **index.html** file as shown in the screenshot below:

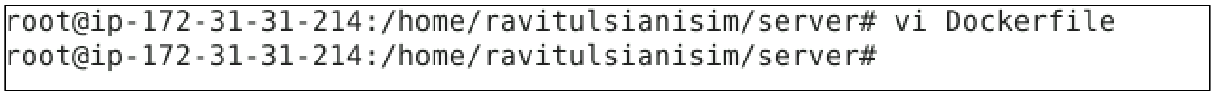
We are learning Microservices in docker


Applications
root@ip-172-31-31-214: ...
root@ip-172-31-31-214: /home/ravitulsianisim/server
File Edit View Search Terminal Help

```
We are learning Microservices in docker
~
~
~
~
~
~
~
~
~
~
~
```

1.8 Create a Dockerfile using the following command:

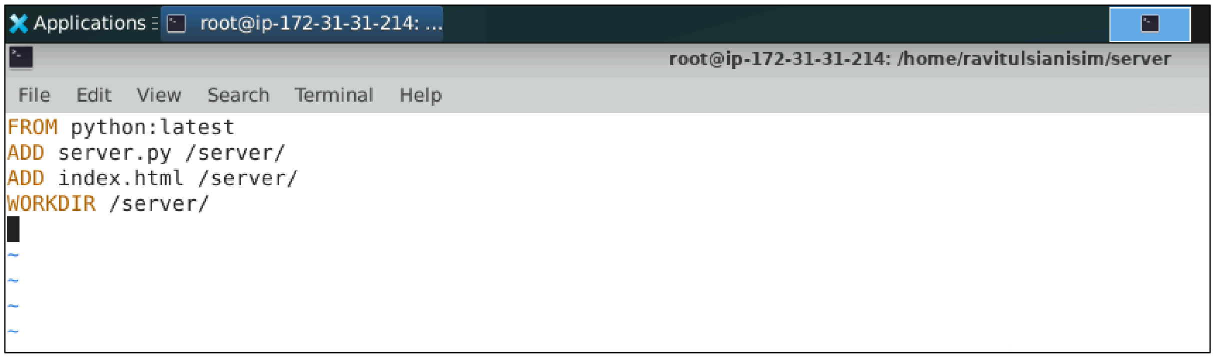
vi Dockerfile



```
root@ip-172-31-31-214:/home/ravitulsianisim/server# vi Dockerfile
root@ip-172-31-31-214:/home/ravitulsianisim/server#
```

1.9 Enter the below script inside the **Dockerfile** file:

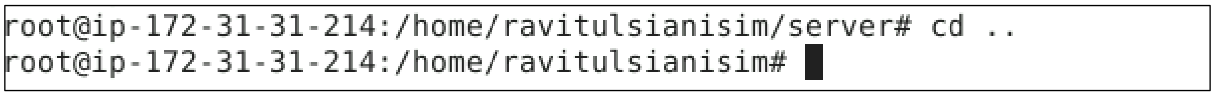
FROM python:latest
ADD server.py /server/
ADD index.html /server/
WORKDIR /server/


Applications
root@ip-172-31-31-214: ...
root@ip-172-31-31-214: /home/ravitulsianisim/server
File Edit View Search Terminal Help

```
FROM python:latest
ADD server.py /server/
ADD index.html /server/
WORKDIR /server/

~
~
~
~
```

This **Dockerfile** sets up a container using the latest Python image, adds **server.py** and **index.html** to the **/server/** directory, and sets **/server/** as the working directory.

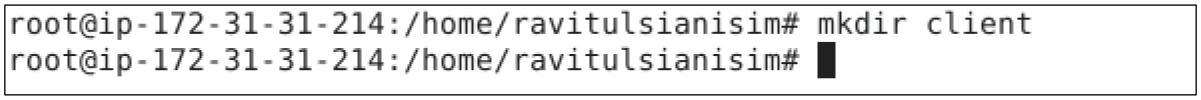
1.10 Exit from the directory using the following command:
cd ..



```
root@ip-172-31-31-214:/home/ravitulsianisim/server# cd ..
root@ip-172-31-31-214:/home/ravitulsianisim#
```

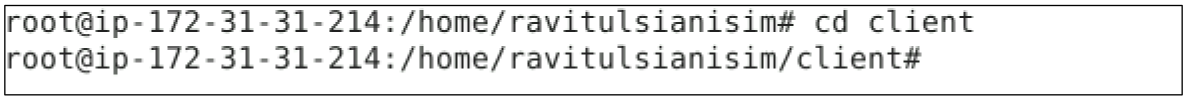
Step 2: Set up a client script with Docker

2.1 Create a **client** directory using the following command**:**
mkdir client



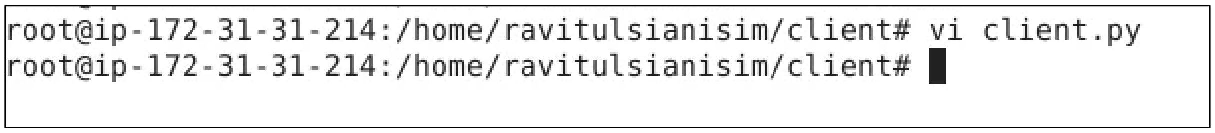
```
root@ip-172-31-31-214:/home/ravitulsianisim# mkdir client
root@ip-172-31-31-214:/home/ravitulsianisim#
```

2.2 Navigate inside the directory using the following command:
cd client



```
root@ip-172-31-31-214:/home/ravitulsianisim# cd client
root@ip-172-31-31-214:/home/ravitulsianisim/client#
```

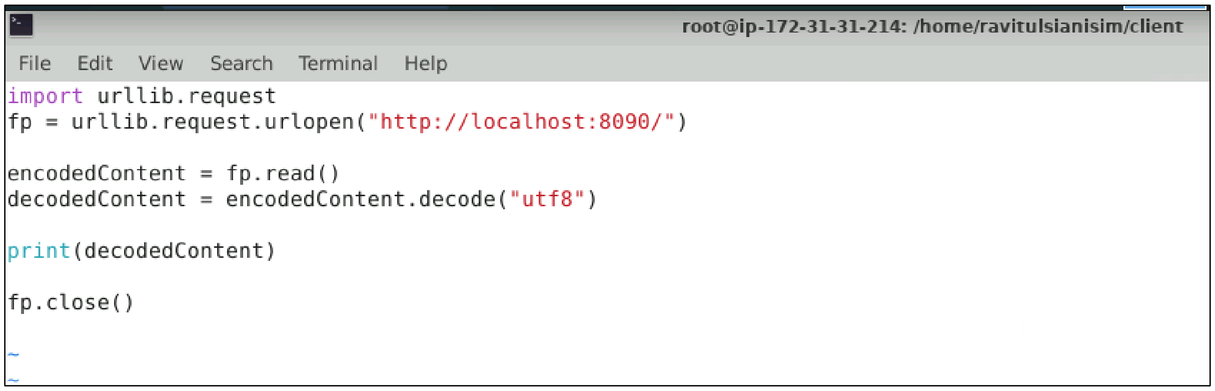
2.3 Create a Python file using the following command:
vi client.py



```
root@ip-172-31-31-214:/home/ravitulsianisim/client# vi client.py
root@ip-172-31-31-214:/home/ravitulsianisim/client#
```

2.4 Enter the below script inside the **client.py** file:

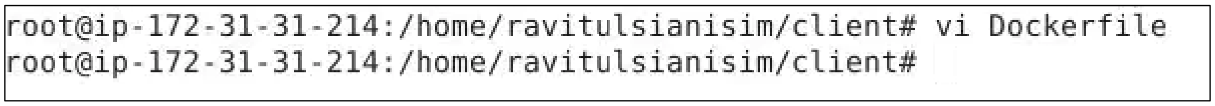
```
import urllib.request
fp = urllib.request.urlopen("http://localhost:8090/")

encodedContent = fp.read()
decodedContent = encodedContent.decode("utf8")

print(decodedContent)

fp.close()
```


root@ip-172-31-31-214: /home/ravitulsianisim/client
File Edit View Search Terminal Help

```
import urllib.request
fp = urllib.request.urlopen("http://localhost:8090/")

encodedContent = fp.read()
decodedContent = encodedContent.decode("utf8")

print(decodedContent)

fp.close()

~
~
```

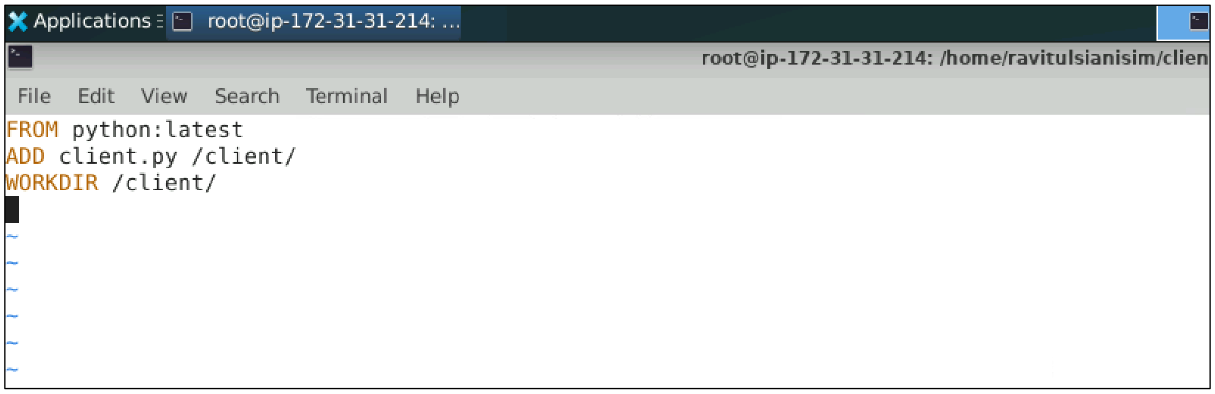
This script fetches the content from a local server at **http://localhost:8090/**, reads and decodes it, prints the decoded content, and then closes the connection.

2.5 Create a Dockerfile using the following command:
vi Dockerfile



```
root@ip-172-31-31-214:/home/ravitulsianisim/client# vi Dockerfile
root@ip-172-31-31-214:/home/ravitulsianisim/client#
```

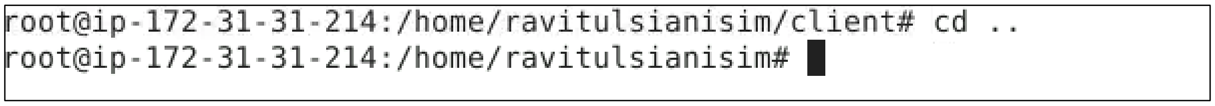
2.6 Enter the below script inside the **Dockerfile** file:
FROM python:latest
ADD client.py /client/
WORKDIR /client/


Applications
root@ip-172-31-31-214: ...
root@ip-172-31-31-214: /home/ravitulsianisim/clien
File Edit View Search Terminal Help

```
FROM python:latest
ADD client.py /client/
WORKDIR /client/

~
~
~
~
~
~
```

This **Dockerfile** sets up a container using the latest Python image, adds **client.py** to the **/client/** directory, and sets **/client/** as the working directory.

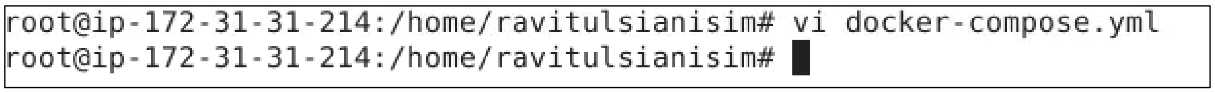
2.7 Exit from the directory using the following command:
cd ..



```
root@ip-172-31-31-214:/home/ravitulsianisim/client# cd ..
root@ip-172-31-31-214:/home/ravitulsianisim#
```

Step 3: Configure Docker networking for microservices

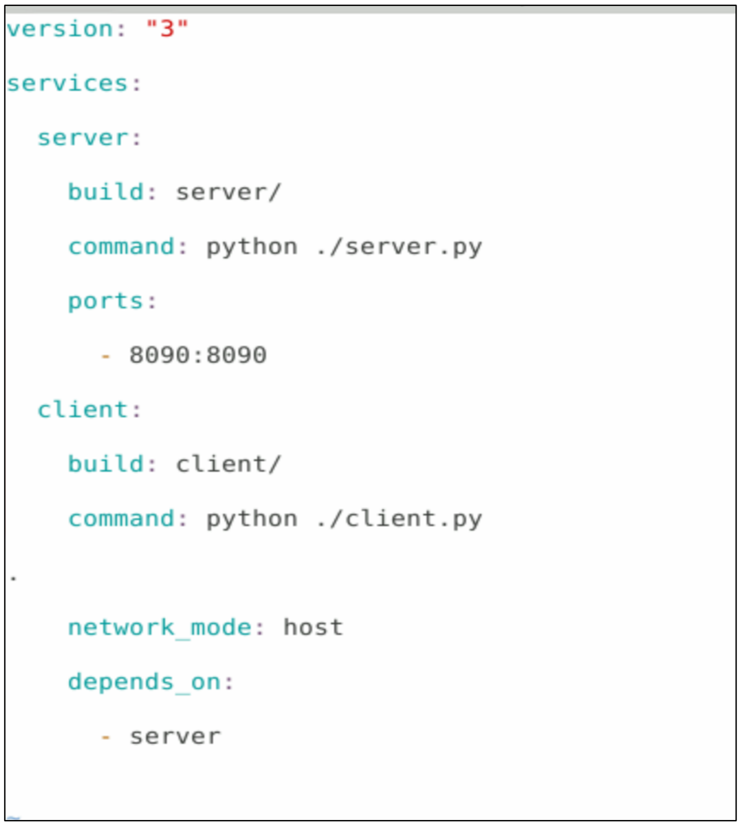
3.1 Create a Docker compose file using the following command:
vi docker-compose.yml



```
root@ip-172-31-31-214:/home/ravitulsianisim# vi docker-compose.yml
root@ip-172-31-31-214:/home/ravitulsianisim#
```

3.2 Enter the below script inside the **docker-compose.yml** file:

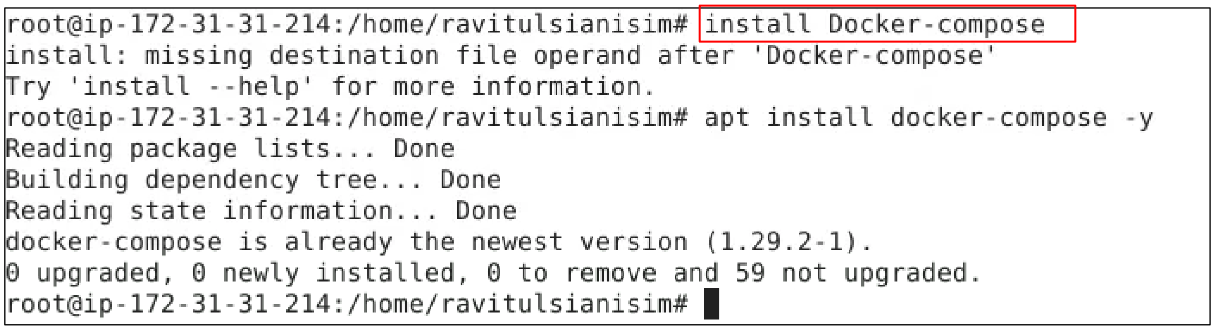
```
version: "3"
services:
  server:
    build: server/
    command: python ./server.py
    ports:
      - 8090:8090
  client:
    build: client/
    command: python ./client.py

    network_mode: host
    depends_on:
      - server
```



```yaml
version: "3"
services:
  server:
    build: server/
    command: python ./server.py
    ports:
      - 8090:8090
  client:
    build: client/
    command: python ./client.py
.
    network_mode: host
    depends_on:
      - server
```

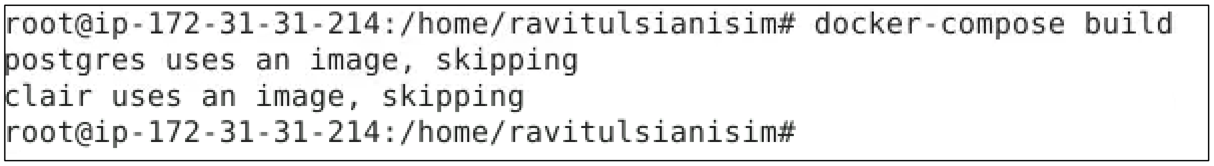
This Docker Compose file version 3 defines services for building and running a server from **server/** on port 8090 and a client from **client/**, where the client depends on the server and shares the host network.

3.3 Run the following command to install Docker Compose in the system:
install Docker-compose



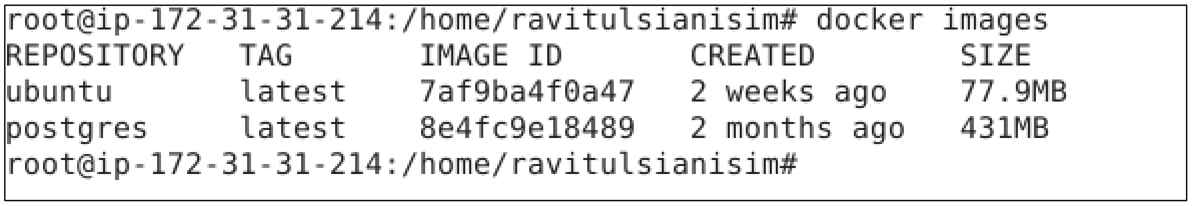
```
root@ip-172-31-31-214:/home/ravitulsianisim# install Docker-compose
install: missing destination file operand after 'Docker-compose'
Try 'install --help' for more information.
root@ip-172-31-31-214:/home/ravitulsianisim# apt install docker-compose -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
docker-compose is already the newest version (1.29.2-1).
0 upgraded, 0 newly installed, 0 to remove and 59 not upgraded.
root@ip-172-31-31-214:/home/ravitulsianisim#
```

3.4 Build the Docker Compose using the following command:
docker-compose build



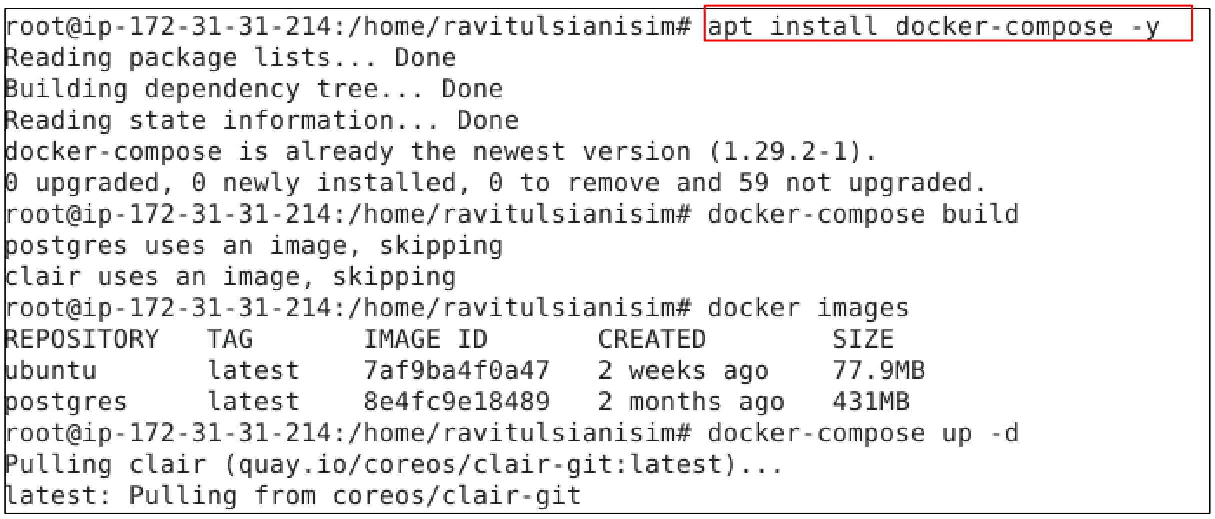
```
root@ip-172-31-31-214:/home/ravitulsianisim# docker-compose build
postgres uses an image, skipping
clair uses an image, skipping
root@ip-172-31-31-214:/home/ravitulsianisim#
```

3.5 List all the Docker images that are currently stored on a system:
docker images



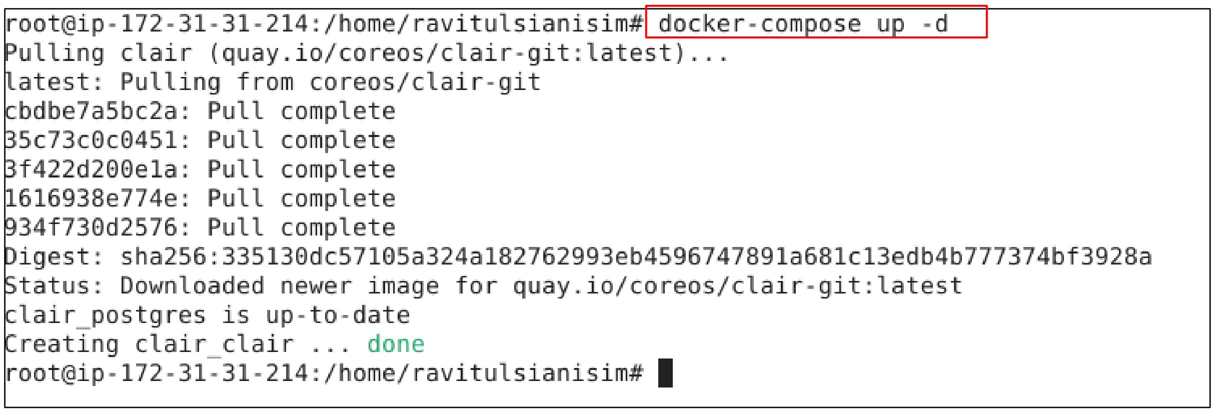
```
root@ip-172-31-31-214:/home/ravitulsianisim# docker images
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
ubuntu       latest    7af9ba4f0a47   2 weeks ago     77.9MB
postgres     latest    8e4fc9e18489   2 months ago    431MB
root@ip-172-31-31-214:/home/ravitulsianisim#
```

3.6 Run the following command to install the Docker Compose:
apt install docker-compose -y



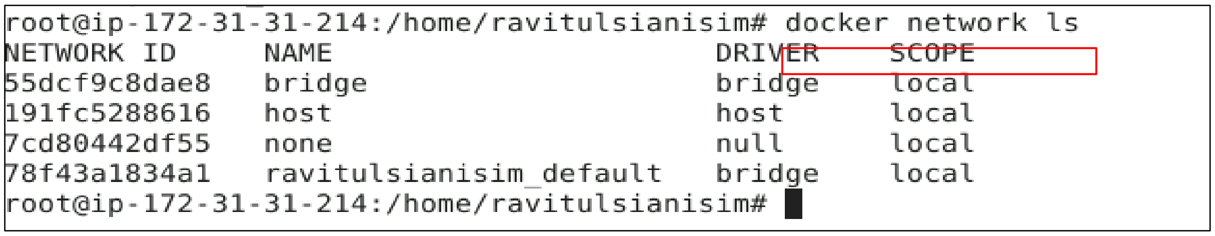
```
root@ip-172-31-31-214:/home/ravitulsianisim# apt install docker-compose -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
docker-compose is already the newest version (1.29.2-1).
0 upgraded, 0 newly installed, 0 to remove and 59 not upgraded.
root@ip-172-31-31-214:/home/ravitulsianisim# docker-compose build
postgres uses an image, skipping
clair uses an image, skipping
root@ip-172-31-31-214:/home/ravitulsianisim# docker images
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
ubuntu       latest    7af9ba4f0a47   2 weeks ago     77.9MB
postgres     latest    8e4fc9e18489   2 months ago    431MB
root@ip-172-31-31-214:/home/ravitulsianisim# docker-compose up -d
Pulling clair (quay.io/coreos/clair-git:latest)...
latest: Pulling from coreos/clair-git
```

3.7 Start the Docker containers defined in a Docker Compose file:
docker-compose up -d



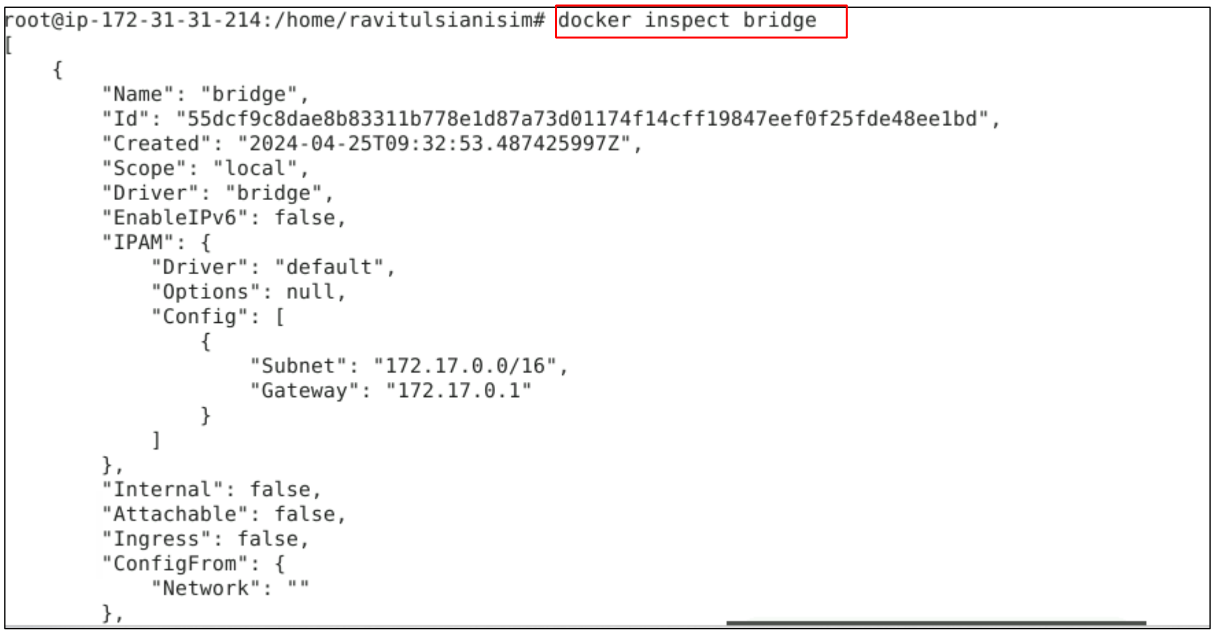
```
root@ip-172-31-31-214:/home/ravitulsianisim# docker-compose up -d
Pulling clair (quay.io/coreos/clair-git:latest)...
latest: Pulling from coreos/clair-git
cbdbe7a5bc2a: Pull complete
35c73c0c0451: Pull complete
3f422d200e1a: Pull complete
1616938e774e: Pull complete
934f730d2576: Pull complete
Digest: sha256:335130dc57105a324a182762993eb4596747891a681c13edb4b777374bf3928a
Status: Downloaded newer image for quay.io/coreos/clair-git:latest
clair_postgres is up-to-date
Creating clair_clair ... done
root@ip-172-31-31-214:/home/ravitulsianisim#
```

3.8 Check the network in which the microservice is running using the following command:
docker network ls



```
root@ip-172-31-31-214:/home/ravitulsianisim# docker network ls
NETWORK ID     NAME                    DRIVER    SCOPE
55dcf9c8dae8   bridge                  bridge    local
191fc5288616   host                    host      local
7cd80442df55   none                    null      local
78f43a1834a1   ravitulsianisim_default bridge    local
root@ip-172-31-31-214:/home/ravitulsianisim#
```

3.9 Inspect the network using the following command:
docker inspect bridge



```
root@ip-172-31-31-214:/home/ravitulsianisim# docker inspect bridge
[
    {
        "Name": "bridge",
        "Id": "55dcf9c8dae8b83311b778e1d87a73d01174f14cff19847eef0f25fde48ee1bd",
        "Created": "2024-04-25T09:32:53.487425997Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16",
                    "Gateway": "172.17.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
```

By following these steps, you have successfully configured Docker networking for a microservices environment for seamless communication and scalability among microservices within Docker containers.