“Expert Cloud Consulting”

SOP | Multi-Container Application with Flask, Node.js & MySQL using Docker Compose

4 july 2025

—

Contributed by: Akshata
Approved by: Akshay (In Review)
Expert Cloud Consulting
Office #811, Gera Imperium Rise,
Hinjewadi Phase-II Rd, Pune, India – 411057

Multi-Container Application with Flask, Node.js & MySQL using Docker Compose

Expert Cloud Consulting

Enhance Optimise & Scale

ASCP GPUonCLOUD Pvt Ltd

Week 5: Containerization Basics

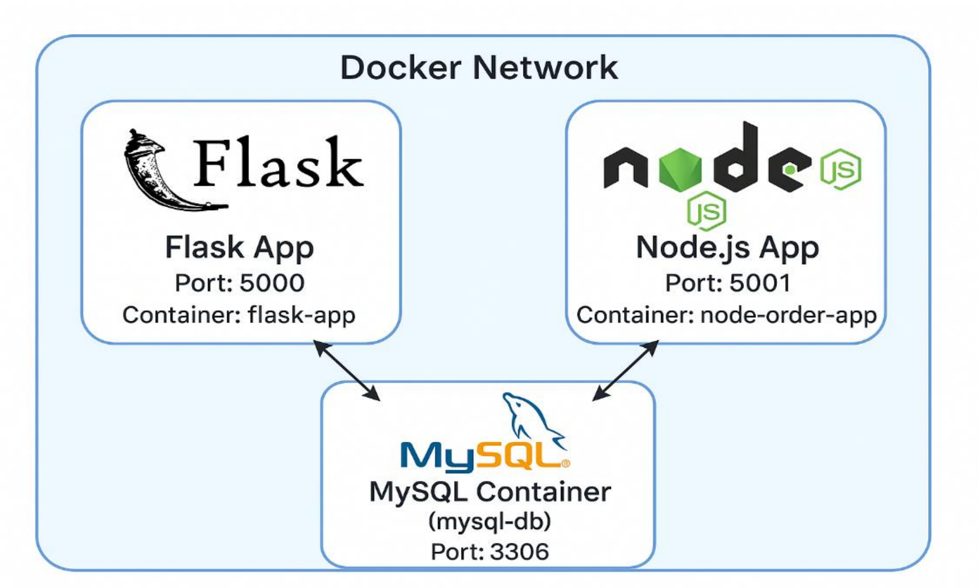
Topics :

- Docker fundamentals: images, containers, volumes, and networks.
- Docker Compose for multi-container applications.

Assignments:

1. Containerize a microservices-based e-commerce application:
- One service for product catalog (Python/Flask).
- Another service for orders (Node.js).
- A shared database container (MySQL).

2. Use Docker Compose to:
- Orchestrate the services.
- Configure persistent storage for the database.



Objective

Deploy a two-tier web application architecture using Docker. The project includes:

- A **Flask application** (running on port 5000)
- A **Node.js application** (running on port 5001)
- A **shared MySQL database**
- Data entered through either app is stored and viewable in MySQL.

Document Overview

This document outlines the setup of a two-tier containerized application using Docker. It includes two services—one built with Node.js and the other with Flask—running on ports 5001 and 5000 respectively. Both services are connected to a shared MySQL database through a dedicated Docker network. Each application has its own Dockerfile and is managed using separate Docker Compose configurations. The architecture ensures clean service separation while enabling database integration

Document References

The following resources were referred to during the creation and execution of this Terraform-based infrastructure setup

Date	Document	Filename / Url
3 July	two-tier-flask-app	https://youtu.be/dXUnAK9_ets?si=ea3yk3uVWdaza3PN
4 July	DockerMySQL+Node.jsApp	https://medium.com/jungletronics/docker-mysql-node-js-app-88f696d837bb

Technology Stack

- Flask **(**Python backend)
- Node.js (JavaScript backend)
- MySQL (Database)
- Docker & Docker Compose (Container orchestration)

Install Docker

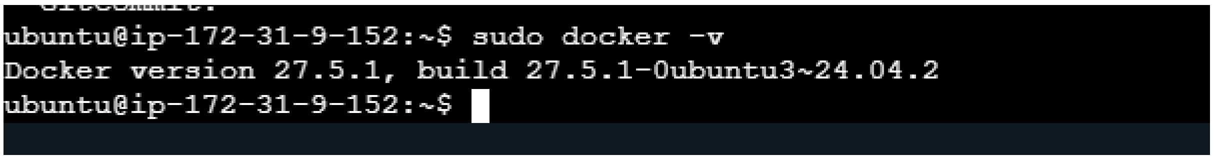
For Ubuntu:

Update existing packages:

```
sudo apt update
sudo apt install docker.io -y
```

Check Docker version:

```
Sudo docker -v
```



```
ubuntu@ip-172-31-9-152:~$ sudo docker -v
Docker version 27.5.1, build 27.5.1-0ubuntu3~24.04.2
ubuntu@ip-172-31-9-152:~$
```

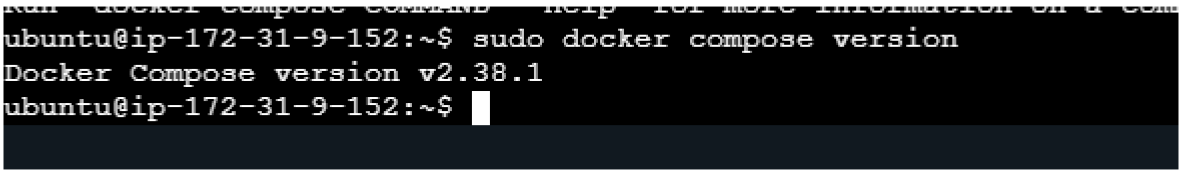
Add your user to the docker group to avoid needing *sudo* for Docker commands:

```
sudo usermod -aG docker $USER
```

Install Docker Compose:

```
sudo mkdir -p /usr/local/lib/docker/cli-plugins

sudo curl -SL "https://github.com/docker/compose/releases/latest/download/docker-compose-linux-
x86_64" \-o /usr/local/lib/docker/cli-plugins/docker-compose

sudo chmod +x /usr/local/lib/docker/cli-plugins/docker-compose
```

docker compose version

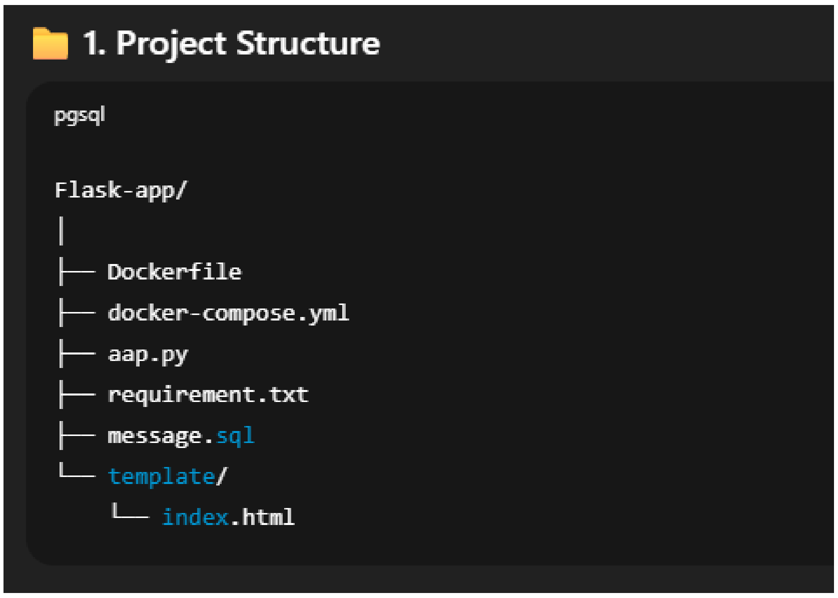


```
ubuntu@ip-172-31-9-152:~$ sudo docker compose version
Docker Compose version v2.38.1
ubuntu@ip-172-31-9-152:~$
```

Clone the Flask application from GitHub

https://github.com/akshataujawane/Flask-app.git

Project Structure



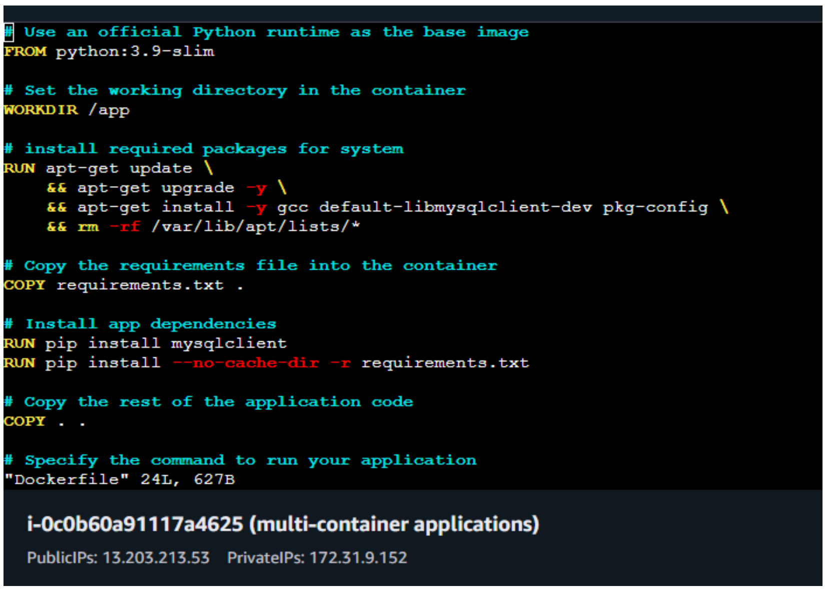
1. Project Structure

```
pgsql

Flask-app/
|
├── Dockerfile
├── docker-compose.yml
├── aap.py
├── requirement.txt
├── message.sql
└── template/
    └── index.html
```

Dockerfile for Flask App:

Sudo vi Dockerfile

This Dockerfile ensures that your Flask application runs consistently in any environment, making it easy to deploy and scale.



```
# Use an official Python runtime as the base image
FROM python:3.9-slim

# Set the working directory in the container
WORKDIR /app

# install required packages for system
RUN apt-get update \
    && apt-get upgrade -y \
    && apt-get install -y gcc default-libmysqlclient-dev pkg-config \
    && rm -rf /var/lib/apt/lists/*

# Copy the requirements file into the container
COPY requirements.txt .

# Install app dependencies
RUN pip install mysqlclient
RUN pip install --no-cache-dir -r requirements.txt

# Copy the rest of the application code
COPY . .

# Specify the command to run your application
"Dockerfile" 24L, 627B
```

i-0c0b60a91117a4625 (multi-container applications)

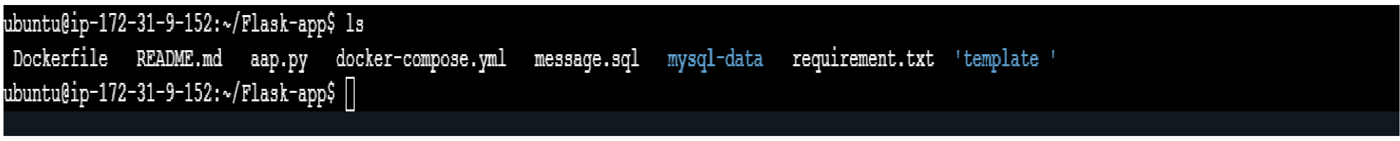
PublicIPs: 13.203.213.53 PrivateIPs: 172.31.9.152

Docker Compose for Flask + MySQL (Two-Tier App) use a docker-compose.yml file to manage multiple services (Flask app + MySQL) together. Docker Compose allows us to define, configure, and run multi-container applications easily

sudo docker-compse.yml

```
version: "3.8"

services:
  mysql:
    user: "${UID}:${GID}"
    image: mysql:5.7
    container_name: mysql
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: devops
      MYSQL_USER: admin
      MYSQL_PASSWORD: admin
    volumes:
      - ./mysql-data:/var/lib/mysql
      - ./message.sql:/docker-entrypoint-initdb.d/message.sql
    networks:
      - twotier
    healthcheck:
      test: ["CMD", "mysqladmin", "ping", "-h", "localhost", "-uroot", "-proot"]
      interval: 10s
      timeout: 5s
      retries: 5
      start_period: 60s

  flask-app:
    image: trainwithshubham/two-tier-flask-app:latest
    container_name: flask-app
    ports:
      - "5000:5000"
    environment:
      MYSQL_HOST: mysql
      MYSQL_USER: root
      MYSQL_PASSWORD: root
      MYSQL_DB: devops
    depends_on:
      - mysql
    networks:
      - twotier
    restart: always
    healthcheck:
      test: ["CMD-SHELL", "curl -f http://localhost:5000/health || exit 1"]
      interval: 10s
      timeout: 5s
      retries: 5
      start_period: 30s

networks:
  twotier:
```

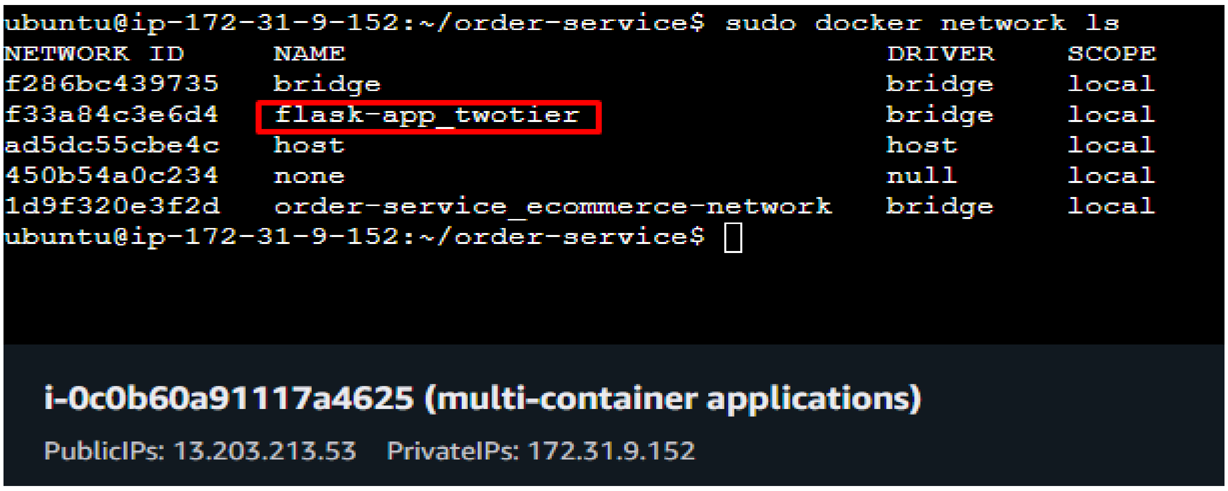
Create the necessary files for your web service and MySQL database



```
ubuntu@ip-172-31-9-152:~/Flask-app$ ls
 Dockerfile   README.md   aap.py   docker-compose.yml   message.sql   mysql-data   requirement.txt  'template '
ubuntu@ip-172-31-9-152:~/Flask-app$ 
```

Set Up Docker Network

Create a custom Docker network that both the MySQL and web containers will shar:
sudo docker network create flask-app_twotier
sudo docker network ls

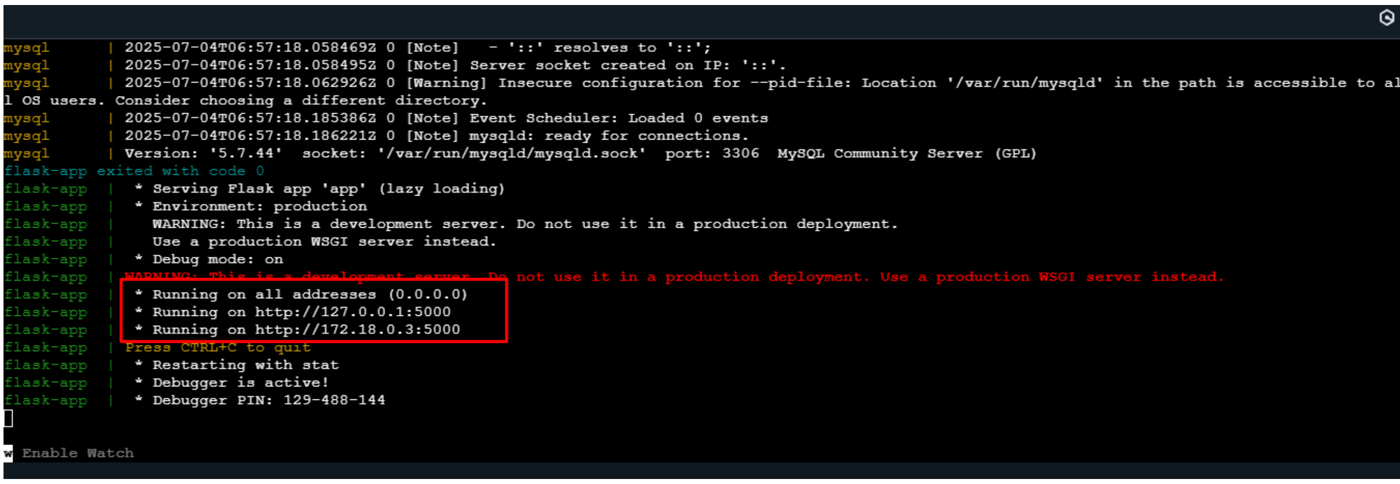


```
ubuntu@ip-172-31-9-152:~/order-service$ sudo docker network ls
NETWORK ID     NAME                             DRIVER    SCOPE
f286bc439735   bridge                           bridge    local
f33a84c3e6d4   flask-app_twotier                bridge    local
ad5dc55cbe4c   host                             host      local
450b54a0c234   none                             null      local
1d9f320e3f2d   order-service_ecommerce-network  bridge    local
ubuntu@ip-172-31-9-152:~/order-service$ 
```

i-0c0b60a91117a4625 (multi-container applications)

PublicIPs: 13.203.213.53   PrivateIPs: 172.31.9.152

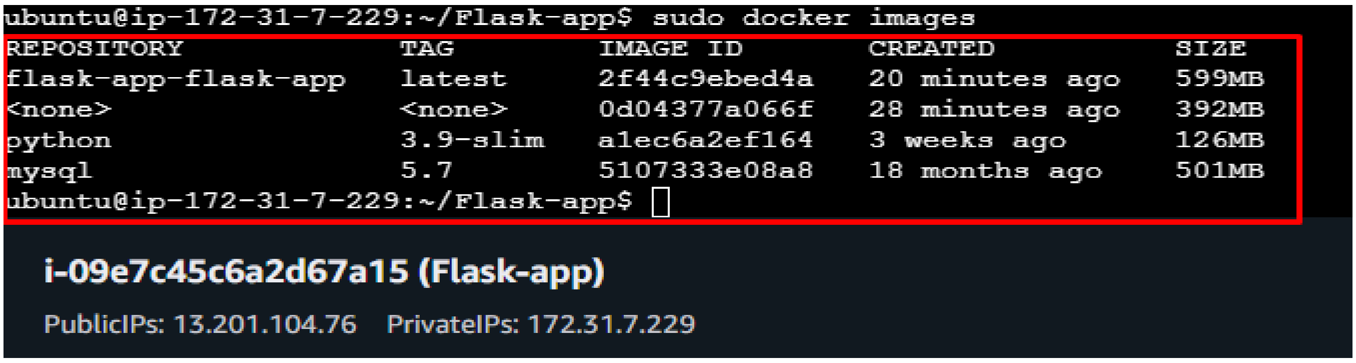
Run Docker Compose:
docker compose up --build



```
mysql        | 2025-07-04T06:57:18.058469Z 0 [Note]   - '::' resolves to '::';
mysql        | 2025-07-04T06:57:18.058495Z 0 [Note] Server socket created on IP: '::'.
mysql        | 2025-07-04T06:57:18.062926Z 0 [Warning] Insecure configuration for --pid-file: Location '/var/run/mysqld' in the path is accessible to al
l OS users. Consider choosing a different directory.
mysql        | 2025-07-04T06:57:18.185386Z 0 [Note] Event Scheduler: Loaded 0 events
mysql        | 2025-07-04T06:57:18.186221Z 0 [Note] mysqld: ready for connections.
mysql        | Version: '5.7.44'  socket: '/var/run/mysqld/mysqld.sock'  port: 3306  MySQL Community Server (GPL)
flask-app exited with code 0
flask-app    |  * Serving Flask app 'app' (lazy loading)
flask-app    |  * Environment: production
flask-app    |    WARNING: This is a development server. Do not use it in a production deployment.
flask-app    |    Use a production WSGI server instead.
flask-app    |  * Debug mode: on
flask-app    | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
flask-app    |  * Running on all addresses (0.0.0.0)
flask-app    |  * Running on http://127.0.0.1:5000
flask-app    |  * Running on http://172.18.0.3:5000
flask-app    | Press CTRL+C to quit
flask-app    |  * Restarting with stat
flask-app    |  * Debugger is active!
flask-app    |  * Debugger PIN: 129-488-144

w Enable Watch
```

Check Docker images and Container : Ensure the container is running:

sudo docker images

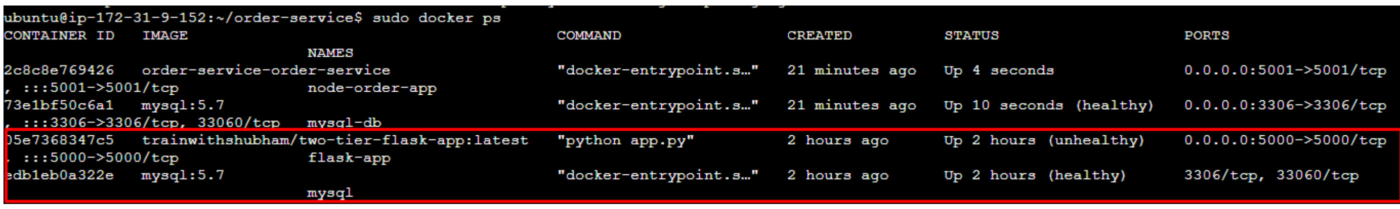


```
ubuntu@ip-172-31-7-229:~/Flask-app$ sudo docker images
REPOSITORY              TAG         IMAGE ID       CREATED           SIZE
flask-app-flask-app     latest      2f44c9ebed4a   20 minutes ago    599MB
<none>                  <none>      0d04377a066f   28 minutes ago    392MB
python                  3.9-slim    a1ec6a2ef164   3 weeks ago       126MB
mysql                   5.7         5107333e08a8   18 months ago     501MB
ubuntu@ip-172-31-7-229:~/Flask-app$
```

i-09e7c45c6a2d67a15 (Flask-app)

PublicIPs: 13.201.104.76   PrivateIPs: 172.31.7.229

Sudo docker ps

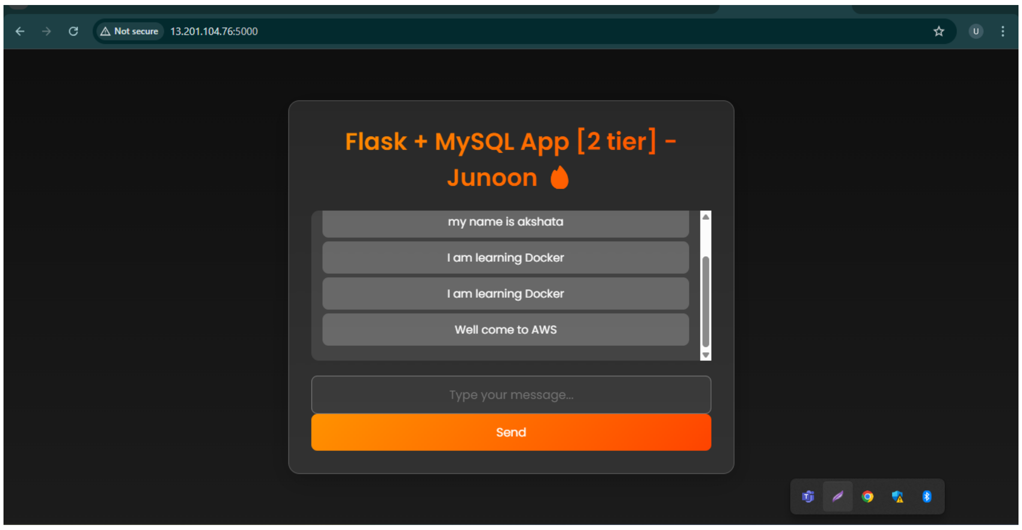


```
ubuntu@ip-172-31-9-152:~/order-service$ sudo docker ps
CONTAINER ID   IMAGE                                       COMMAND                  CREATED          STATUS                     PORTS
                      NAMES
2c8c8e769426   order-service-order-service                 "docker-entrypoint.s…"   21 minutes ago   Up 4 seconds               0.0.0.0:5001->5001/tcp
, :::5001->5001/tcp   node-order-app
73e1bf50c6a1   mysql:5.7                                   "docker-entrypoint.s…"   21 minutes ago   Up 10 seconds (healthy)    0.0.0.0:3306->3306/tcp
, :::3306->3306/tcp, 33060/tcp   mysql-db
)5e7368347c5   trainwithshubham/two-tier-flask-app:latest   "python app.py"          2 hours ago      Up 2 hours (unhealthy)     0.0.0.0:5000->5000/tcp
. :::5000->5000/tcp   flask-app
edb1eb0a322e   mysql:5.7                                   "docker-entrypoint.s…"   2 hours ago      Up 2 hours (healthy)       3306/tcp, 33060/tcp
                      mysql
```

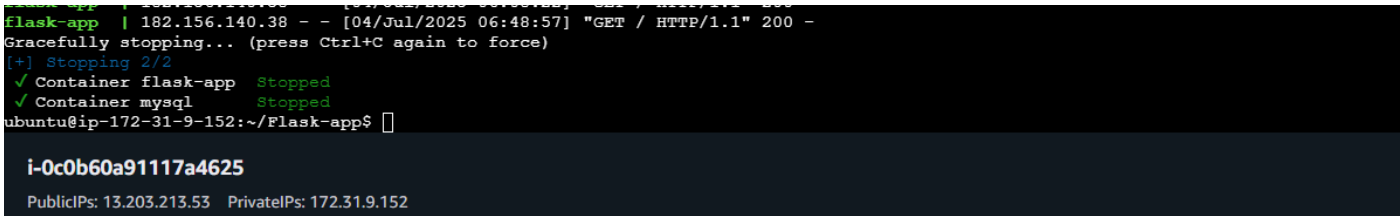
Verify Web Application

Your web application should now be running and accessible at:

http://13.203.213.53:5000/



sudo docker-compose down



```
flask-app  | 182.156.140.38 - - [04/Jul/2025 06:48:57] "GET / HTTP/1.1" 200 -
Gracefully stopping... (press Ctrl+C again to force)
[+] Stopping 2/2
 ✓ Container flask-app  Stopped
 ✓ Container mysql      Stopped
ubuntu@ip-172-31-9-152:~/Flask-app$
```

i-0c0b60a91117a4625

PublicIPs: 13.203.213.53 PrivateIPs: 172.31.9.152

Access the MySQL container to verify if the data has been transferred correctly by executing:

sudo docker exec -it mysql bash

This command allows you to connect to the MySQL database inside the container and inspect tables, rows, and inserted data using SQL queries

Enter the Password

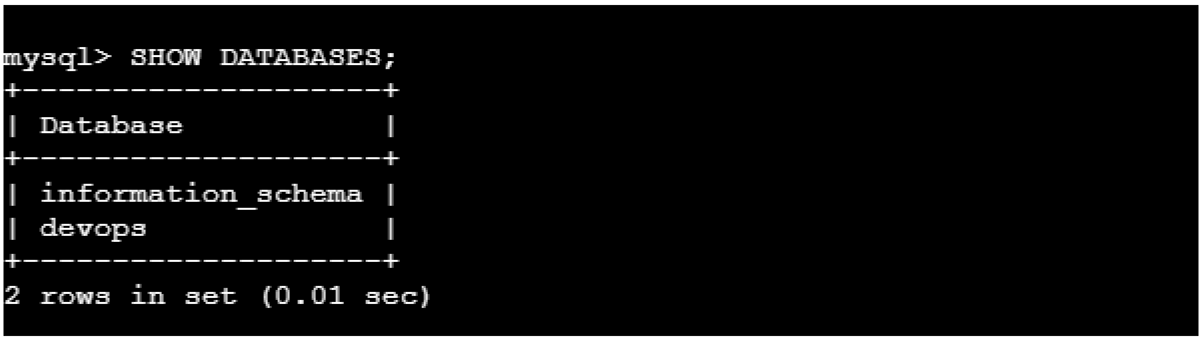
When prompted, enter the MySQL root password you set in your docker-compose.yml.

admin

View Databases

Once inside the MySQL prompt, type:

SHOW DATABASES;

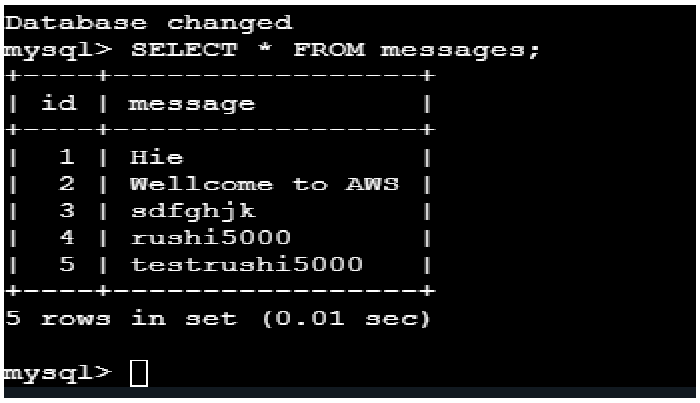


```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| devops             |
+--------------------+
2 rows in set (0.01 sec)
```

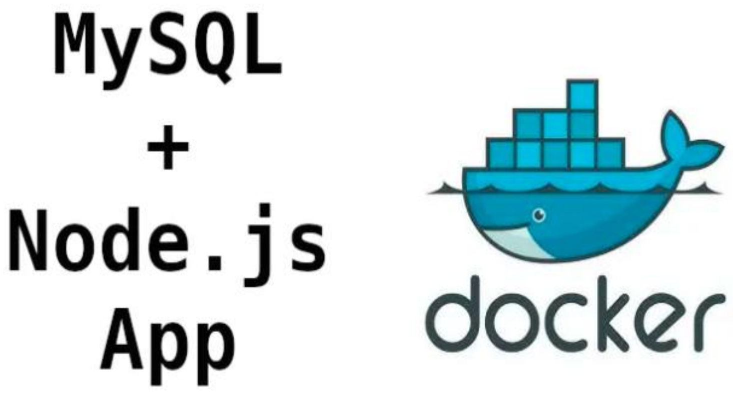
View the Data

Now, to see the data in your

SELECT * FROM orders;



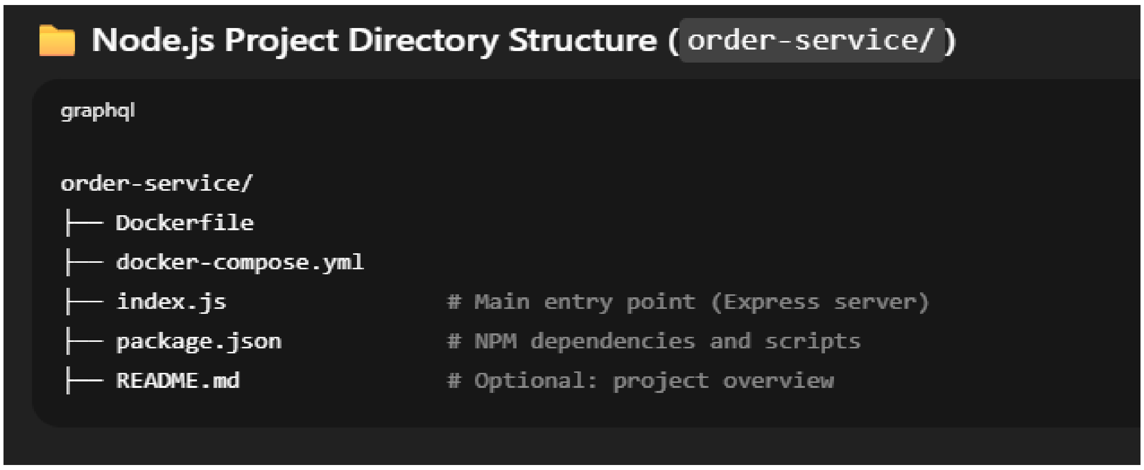
```
Database changed
mysql> SELECT * FROM messages;
+----+------------------+
| id | message          |
+----+------------------+
|  1 | Hie              |
|  2 | Wellcome to AWS  |
|  3 | sdfghjk          |
|  4 | rushi5000        |
|  5 | testrushi5000    |
+----+------------------+
5 rows in set (0.01 sec)

mysql> 
```



To create a multi-tier architecture using Docker, consisting of two applications (Flask and Node.js) connected to a common MySQL database. Each service runs in a separate Docker container and communicates over a custom Docker network.

Clone the node.js application from GitHub

https://github.com/akshataujawane/node.js.git

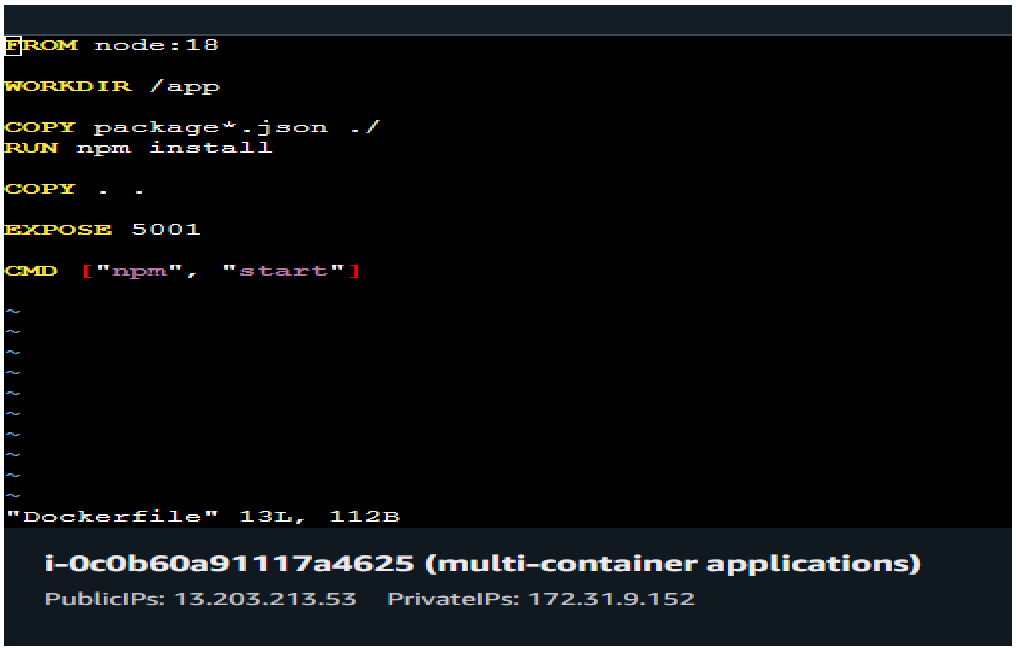


📁 Node.js Project Directory Structure (`order-service/`)

```graphql
order-service/
├── Dockerfile
├── docker-compose.yml
├── index.js                # Main entry point (Express server)
├── package.json            # NPM dependencies and scripts
├── README.md               # Optional: project overview
```

Dockerfile

Builds a container image for your Node.js app.



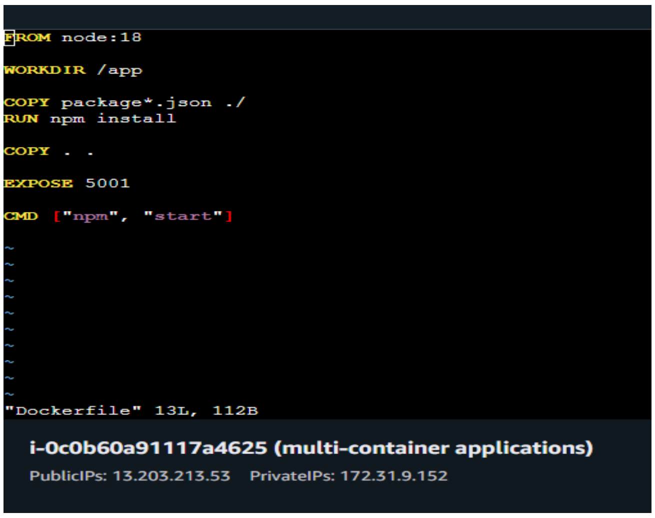
```
FROM node:18
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 5001
CMD ["npm", "start"]
~
~
~
~
~
~
~
~
~
~
"Dockerfile" 13L, 112B
```

i-0c0b60a91117a4625 (multi-container applications)
PublicIPs: 13.203.213.53   PrivateIPs: 172.31.9.152

Expert Cloud Consulting
Enhance Optimise & Scale

Orchestrates your Node.js app and MySQL together.

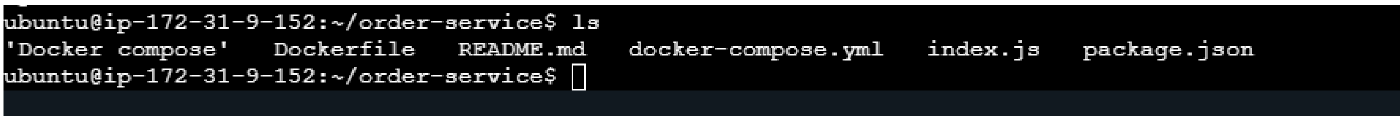
docker-compose.yml



```
FROM node:18

WORKDIR /app

COPY package*.json ./
RUN npm install

COPY . .

EXPOSE 5001

CMD ["npm", "start"]
~
~
~
~
~
~
~
~
~
~
"Dockerfile" 13L, 112B
```

i-0c0b60a91117a4625 (multi-container applications)
PublicIPs: 13.203.213.53 PrivateIPs: 172.31.9.152

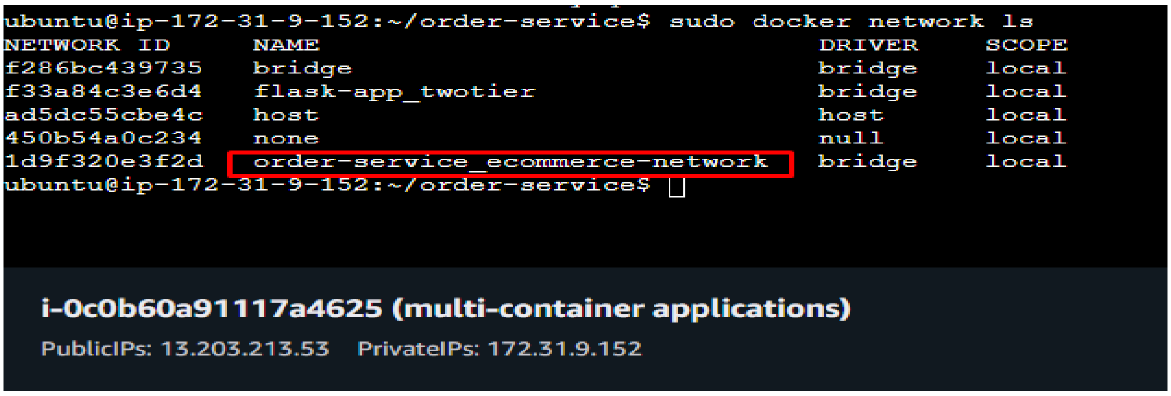
Create the necessary files for your web service and MySQL database



```
ubuntu@ip-172-31-9-152:~/order-service$ ls
'Docker compose'   Dockerfile   README.md   docker-compose.yml   index.js   package.json
ubuntu@ip-172-31-9-152:~/order-service$ 
```

set Up Docker Network

Create a custom Docker network that both the MySQL and web containers will share:
sudo docker network create order-service_ecommerce-network
sudo docker network ls

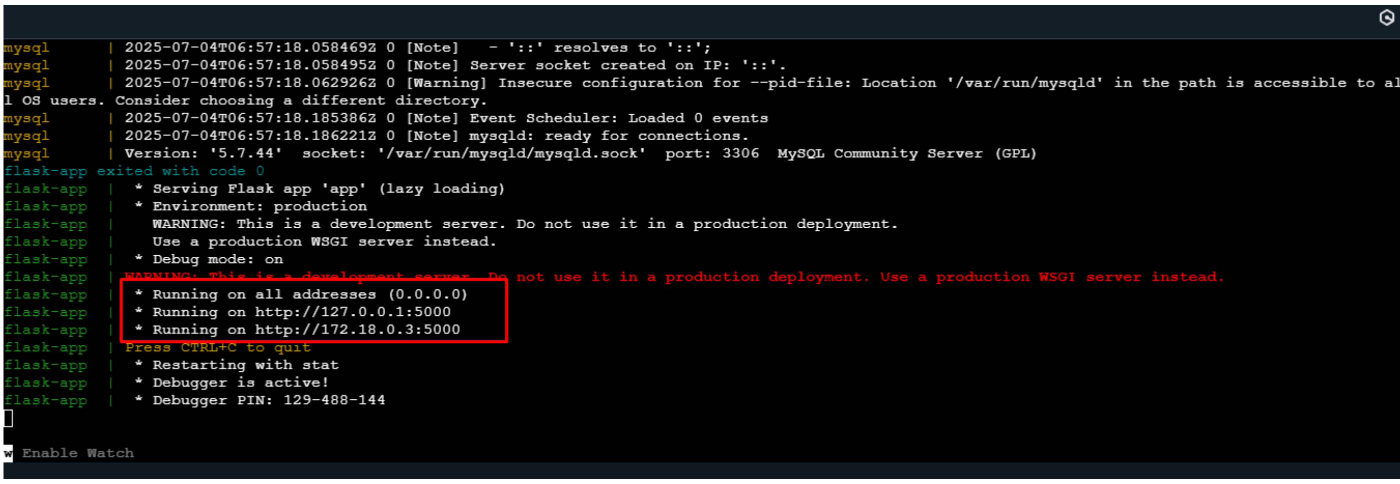


```
ubuntu@ip-172-31-9-152:~/order-service$ sudo docker network ls
NETWORK ID     NAME                                DRIVER    SCOPE
f286bc439735   bridge                              bridge    local
f33a84c3e6d4   flask-app_twotier                   bridge    local
ad5dc55cbe4c   host                                host      local
450b54a0c234   none                                null      local
1d9f320e3f2d   order-service_ecommerce-network     bridge    local
ubuntu@ip-172-31-9-152:~/order-service$ 
```

i-0c0b60a91117a4625 (multi-container applications)
PublicIPs: 13.203.213.53 PrivateIPs: 172.31.9.152

Run Docker Compose:

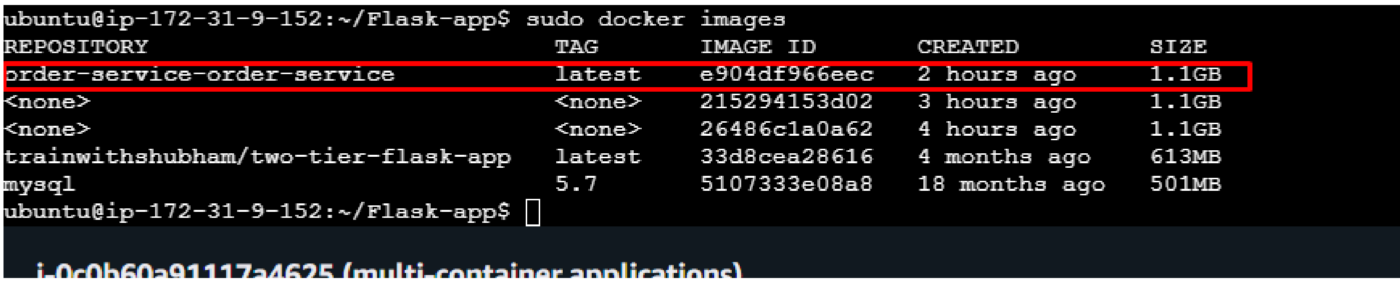
docker compose up –build



```
mysql       | 2025-07-04T06:57:18.058469Z 0 [Note]   - '::' resolves to '::';
mysql       | 2025-07-04T06:57:18.058495Z 0 [Note] Server socket created on IP: '::'.
mysql       | 2025-07-04T06:57:18.062926Z 0 [Warning] Insecure configuration for --pid-file: Location '/var/run/mysqld' in the path is accessible to al
l OS users. Consider choosing a different directory.
mysql       | 2025-07-04T06:57:18.185386Z 0 [Note] Event Scheduler: Loaded 0 events
mysql       | 2025-07-04T06:57:18.186221Z 0 [Note] mysqld: ready for connections.
mysql       | Version: '5.7.44'  socket: '/var/run/mysqld/mysqld.sock'  port: 3306  MySQL Community Server (GPL)
flask-app exited with code 0
flask-app   |  * Serving Flask app 'app' (lazy loading)
flask-app   |  * Environment: production
flask-app   |    WARNING: This is a development server. Do not use it in a production deployment.
flask-app   |    Use a production WSGI server instead.
flask-app   |  * Debug mode: on
flask-app   | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
flask-app   |  * Running on all addresses (0.0.0.0)
flask-app   |  * Running on http://127.0.0.1:5000
flask-app   |  * Running on http://172.18.0.3:5000
flask-app   | Press CTRL+C to quit
flask-app   |  * Restarting with stat
flask-app   |  * Debugger is active!
flask-app   |  * Debugger PIN: 129-488-144

w Enable Watch
```

Check Docker images and Container : Ensure the container is running:

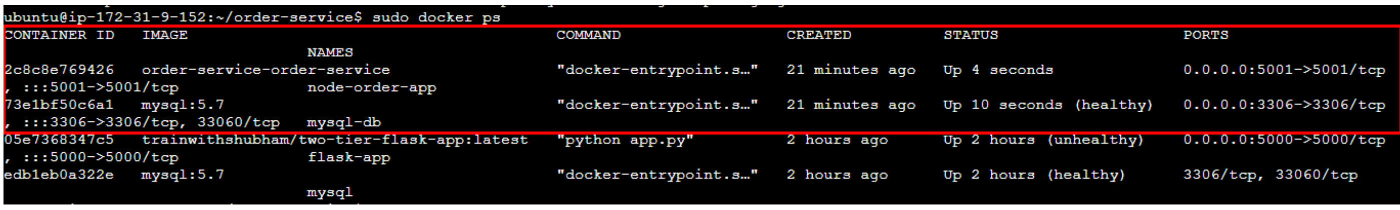
sudo docker images



```
ubuntu@ip-172-31-9-152:~/Flask-app$ sudo docker images
REPOSITORY                         TAG       IMAGE ID       CREATED         SIZE
order-service-order-service        latest    e904df966eec   2 hours ago     1.1GB
<none>                             <none>    215294153d02   3 hours ago     1.1GB
<none>                             <none>    26486c1a0a62   4 hours ago     1.1GB
trainwithshubham/two-tier-flask-app   latest    33d8cea28616   4 months ago    613MB
mysql                              5.7       5107333e08a8   18 months ago   501MB
ubuntu@ip-172-31-9-152:~/Flask-app$
```

i-0c0b60a91117a4625 (multi-container applications)

Sudo docker ps

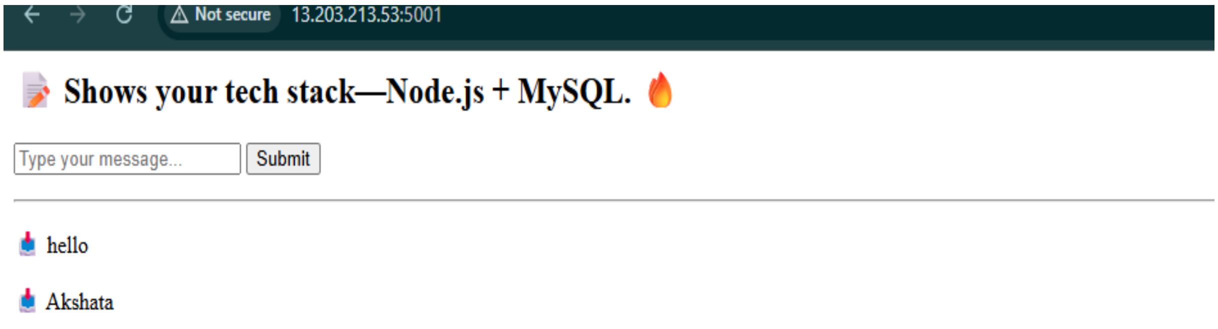


```
ubuntu@ip-172-31-9-152:~/order-service$ sudo docker ps
CONTAINER ID   IMAGE                                      COMMAND                  CREATED          STATUS                    PORTS
                                NAMES
2c8c8e769426   order-service-order-service                "docker-entrypoint.s…"   21 minutes ago   Up 4 seconds              0.0.0.0:5001->5001/tcp
, :::5001->5001/tcp             node-order-app
73e1bf50c6a1   mysql:5.7                                  "docker-entrypoint.s…"   21 minutes ago   Up 10 seconds (healthy)   0.0.0.0:3306->3306/tcp
, :::3306->3306/tcp, 33060/tcp   mysql-db
05e7368347c5   trainwithshubham/two-tier-flask-app:latest "python app.py"          2 hours ago      Up 2 hours (unhealthy)    0.0.0.0:5000->5000/tcp
, :::5000->5000/tcp             flask-app
edb1eb0a322e   mysql:5.7                                  "docker-entrypoint.s…"   2 hours ago      Up 2 hours (healthy)      3306/tcp, 33060/tcp
                                mysql
```

Verify Web Application

Your web application should now be running and accessible at:

http://13.203.213.53:5001/

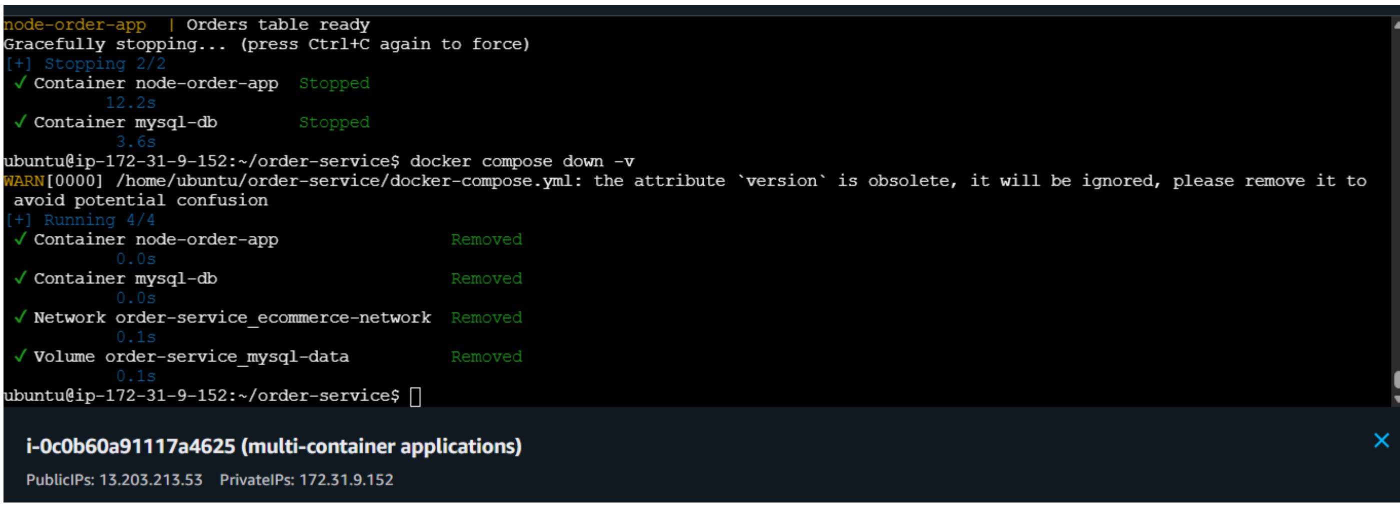


Not secure 13.203.213.53:5001

📝 Shows your tech stack—Node.js + MySQL. 🔥

- hello
- Akshata

sudo docker-compose down



```
node-order-app  | Orders table ready
Gracefully stopping... (press Ctrl+C again to force)
[+] Stopping 2/2
 ✓ Container node-order-app  Stopped
          12.2s
 ✓ Container mysql-db        Stopped
           3.6s
ubuntu@ip-172-31-9-152:~/order-service$ docker compose down -v
WARN[0000] /home/ubuntu/order-service/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to
 avoid potential confusion
[+] Running 4/4
 ✓ Container node-order-app                 Removed
           0.0s
 ✓ Container mysql-db                       Removed
           0.0s
 ✓ Network order-service_ecommerce-network  Removed
           0.1s
 ✓ Volume order-service_mysql-data          Removed
           0.1s
ubuntu@ip-172-31-9-152:~/order-service$
```

i-0c0b60a91117a4625 (multi-container applications)

PublicIPs: 13.203.213.53   PrivateIPs: 172.31.9.152

Access the MySQL container to verify if the data has been transferred correctly by executing:

docker exec -it mysql-db mysql -uroot -p

This command allows you to connect to the MySQL database inside the container and inspect tables, rows, and inserted data using SQL queries

Enter the Password

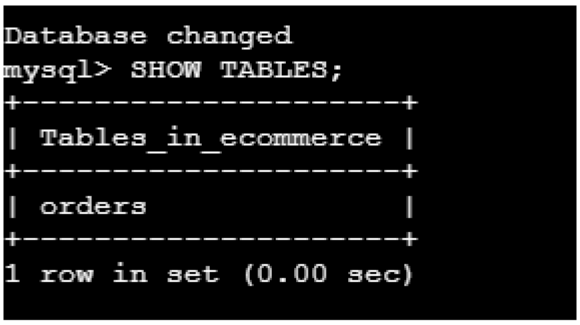
When prompted, enter the MySQL root password you set in your docker-compose.yml.

admin

View Databases

Once inside the MySQL prompt, type:

SHOW DATABASES;

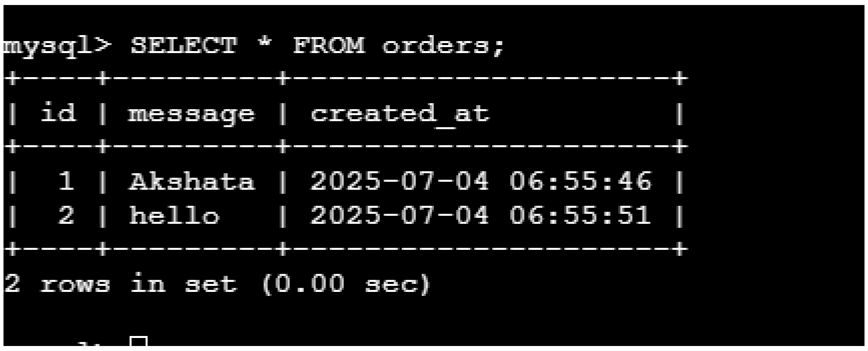


```
Database changed
mysql> SHOW TABLES;
+----------------------+
| Tables_in_ecommerce |
+----------------------+
| orders               |
+----------------------+
1 row in set (0.00 sec)
```

View the Data

Now, to see the data in your

SELECT * FROM orders;



```
mysql> SELECT * FROM orders;
+----+---------+---------------------+
| id | message | created_at          |
+----+---------+---------------------+
|  1 | Akshata | 2025-07-04 06:55:46 |
|  2 | hello   | 2025-07-04 06:55:51 |
+----+---------+---------------------+
2 rows in set (0.00 sec)
```