“Expert Cloud Consulting” -

Containerization Basics [Title,18, Arial]

06.Jan.2025 **[Subtitle,14, Arial]**

version 1.0
—

Contributed by Shraddha Chaudhari **[Normal text,14, Arial]**
Approved by   (In Review)
Expert Cloud Consulting
Office #811, Gera Imperium Rise,
Hinjewadi Phase-II Rd, Pune, India – 411057

“Expert Cloud Consulting”

Introduction to infrastructure as code (IAC) [Title,18, Arial]

1.0 Contents [Heading3, 14, Arial]

2.0 General Information: **[Heading3,14, Arial]**

2.1 Document Jira/ Github Ticket(s) **[Heading4,12, Arial]**

Ticket(s) Name	**Url**
Containerization Basics **[Normal text,10, Arial]**	https://github.com/shaanicha/Weekly_Tasks/tree/main/06Jan-10Jan_Task

2.2 Document Purpose

This manual lays out the processes and guidelines for setting up the Ubuntu linux operating system for the .Net core application on aws EC2 instance. **[Normal text,10, Arial, Justify Alignment]**

2.3 Document Revisions

Date	**Version**	**Contributor (s)**	**Approver(s)**	**Section(s)**	**Change(s)**
10/Jan/2025	1.0	Shraddha Chaudhari	Akshay Shinde	All Sections	New Document Created

2.4 Document References

The following artefacts are referenced within this document. Please refer to the original documents for additional information.

Date	**Document**	**Filename / Url**
2023	volumes	https://docs.docker.com/engine/storage/volumes/

2023	storage	https://docs.docker.com/engine/storage/
2025	Install Docker on Ubuntu	https://docs.docker.com/engine/install/ubuntu/

3.0 Document Overview: -> Docker composed task

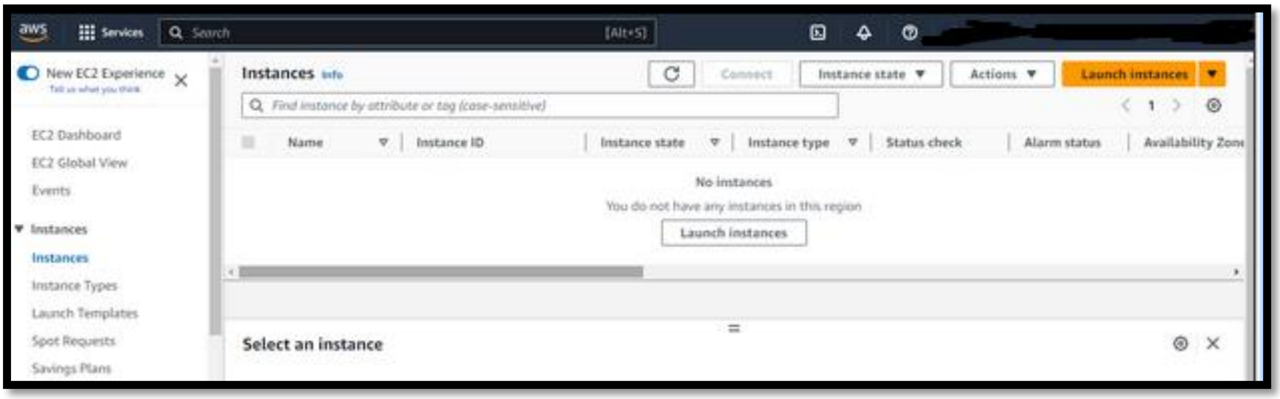
- **Orchestrate Services:** Docker Compose allows you to define and manage multiple interdependent services (like a web app and database) using a single docker-compose.yml file. It simplifies starting, stopping, and scaling services with commands like docker-compose up and down.

- **Persistent Storage for Database:** Using Docker volumes in the Compose file ensures that database data persists even if the container is stopped or removed. For example, you can map a host directory or named volume to the database container's data directory (`/var/lib/post

4.0 Steps / Procedure

4.1 : Launch an EC2 Instance

Log in to the AWS Management Console and navigate to the EC2 dashboard.
Click on the "Launch Instance" button to start the process of launching a new EC2 instance.

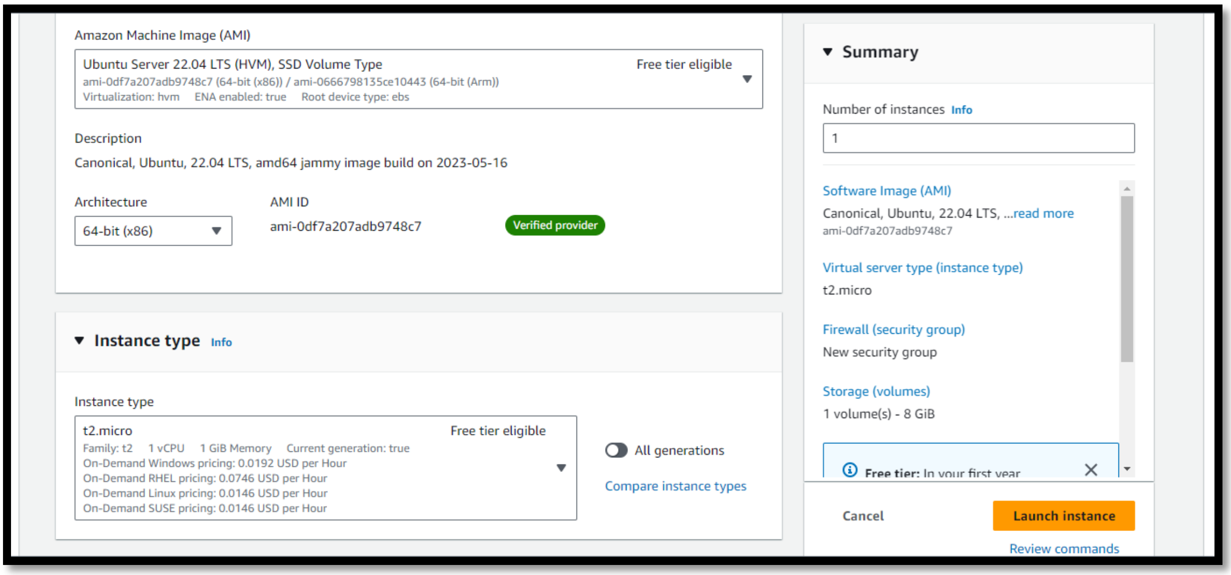


No instances

You do not have any instances in this region

4.2: Choose an Amazon Machine Image (AMI)

Select an instance type, configure your instance details (such as the number of instances, and storage)



Amazon Machine Image (AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type — Free tier eligible
ami-0df7a207adb9748c7 (64-bit (x86)) / ami-0666798135ce10443 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Description
Canonical, Ubuntu, 22.04 LTS, amd64 jammy image build on 2023-05-16

Architecture: 64-bit (x86)
AMI ID: ami-0df7a207adb9748c7 — Verified provider

▼ Instance type Info

Instance type
t2.micro — Free tier eligible
Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Windows pricing: 0.0192 USD per Hour
On-Demand RHEL pricing: 0.0746 USD per Hour
On-Demand Linux pricing: 0.0146 USD per Hour
On-Demand SUSE pricing: 0.0146 USD per Hour

○ All generations
Compare instance types

▼ Summary

Number of instances Info: 1

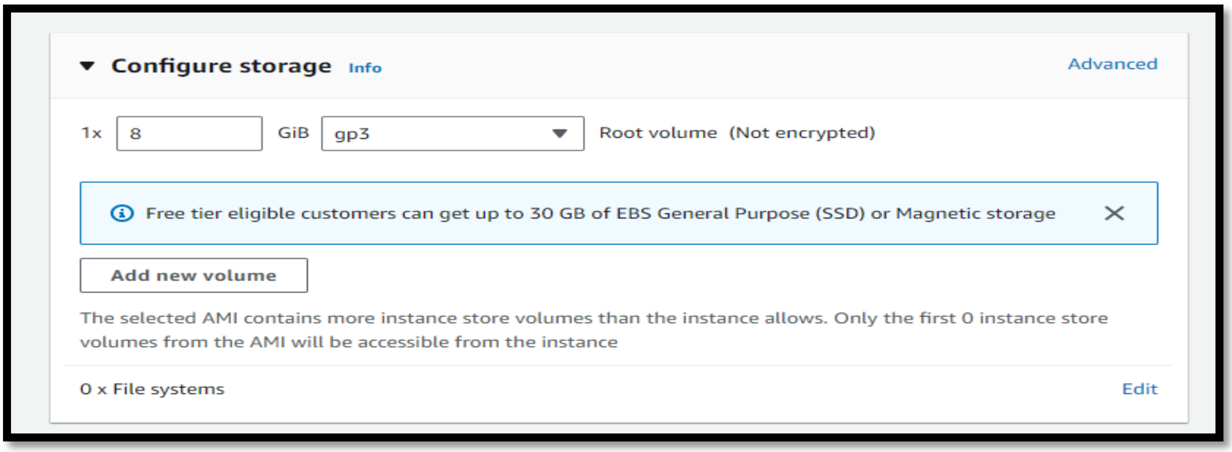
Software Image (AMI)
Canonical, Ubuntu, 22.04 LTS, ...read more
ami-0df7a207adb9748c7

Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

Free tier: In your first year



▼ Configure storage Info — Advanced

1x 8 GiB gp3 Root volume (Not encrypted)

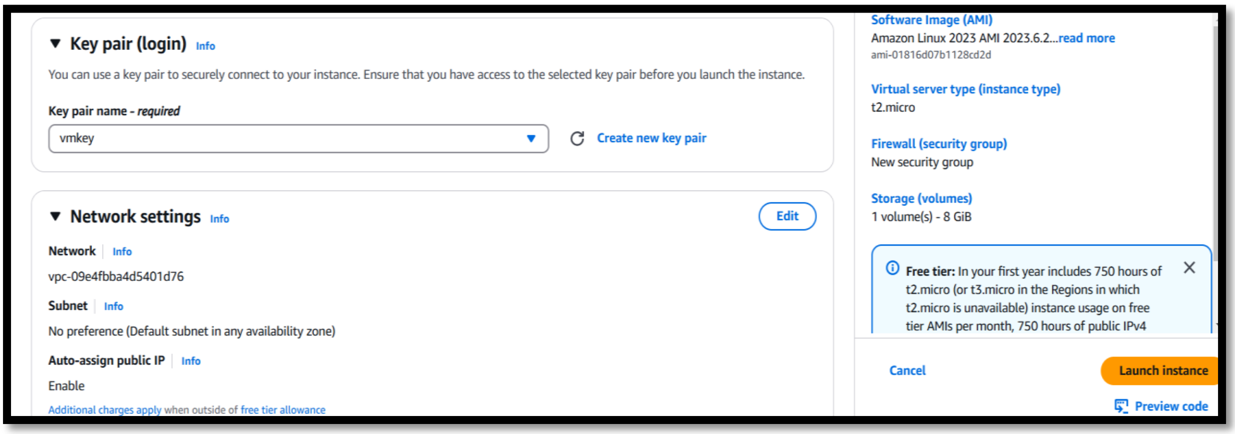
Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage

The selected AMI contains more instance store volumes than the instance allows. Only the first 0 instance store volumes from the AMI will be accessible from the instance

0 x File systems — Edit

4.3: Key-Pair Configuration

Select an instance type and create a new key-pair as name is sandbox-jenkins-keypair.



▼ Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*
vmkey

▼ Network settings Info — Edit

Network Info
vpc-09e4fbba4d5401d76

Subnet Info
No preference (Default subnet in any availability zone)

Auto-assign public IP Info
Enable

Additional charges apply when outside of free tier allowance

Software Image (AMI)
Amazon Linux 2023 AMI 2023.6.2...read more
ami-01816d07b1128cd2d

Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

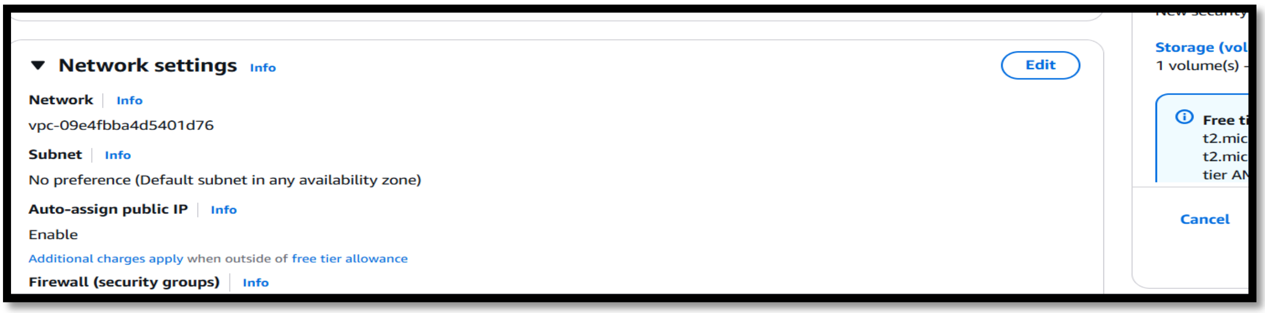
Storage (volumes)
1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4

4.4: Network settings

4.4.1: VPC Configuration:

Select the vpc and subnet for the ec2 instance.



▼ Network settings Info

Edit

Network | Info

vpc-09e4fbba4d5401d76

Subnet | Info

No preference (Default subnet in any availability zone)

Auto-assign public IP | Info

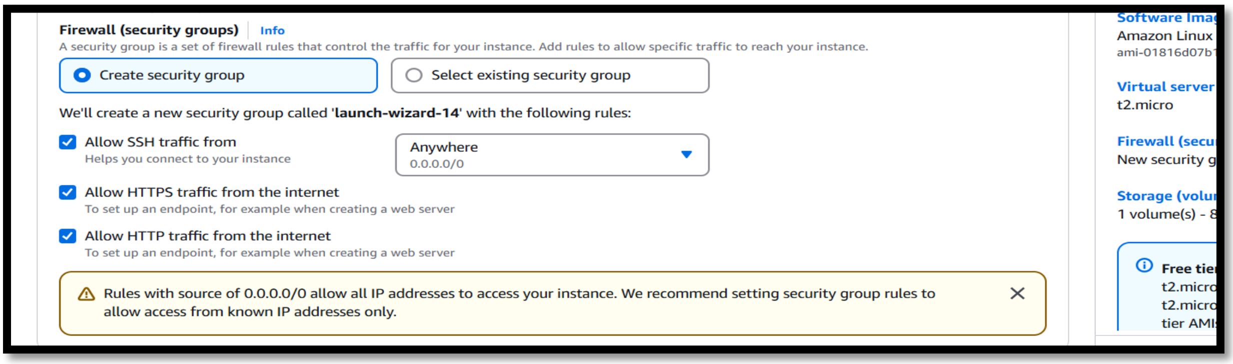
Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups) | Info

4.4.2: Security Group Configuration

Specified Security group rule for this ec2 instances are shown below:



Firewall (security groups) | Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

● Create security group ○ Select existing security group

We'll create a new security group called 'launch-wizard-14' with the following rules:

☑ Allow SSH traffic from
Helps you connect to your instance
Anywhere
0.0.0.0/0

☑ Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server

☑ Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server

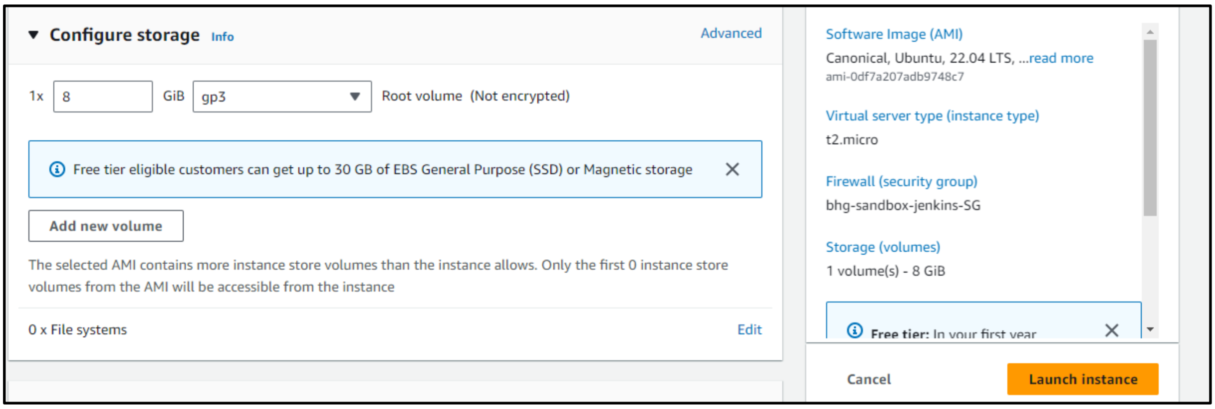
⚠ Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Add a security group to allow:

- Port 22 (SSH).
- Port 5000 (for the app).
- Port 5432 (for PostgreSQL, if needed externally).

4.5: Launch Instance

click on Launch instance



▼ Configure storage Info

1x 8 GiB gp3 ▼ Root volume (Not encrypted)

ⓘ Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage ✕

The selected AMI contains more instance store volumes than the instance allows. Only the first 0 instance store volumes from the AMI will be accessible from the instance

0 x File systems

Software Image (AMI)
Canonical, Ubuntu, 22.04 LTS, ...read more
ami-0df7a207adb9748c7

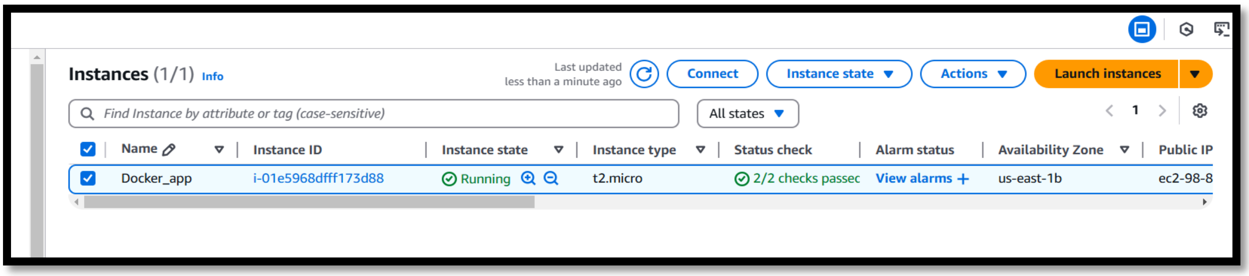
Virtual server type (instance type)
t2.micro

Firewall (security group)
bhg-sandbox-jenkins-SG

Storage (volumes)
1 volume(s) - 8 GiB

ⓘ Free tier: In your first year ✕

Instances are created and they are ready to use.



Instances (1/1) Info

Last updated
less than a minute ago

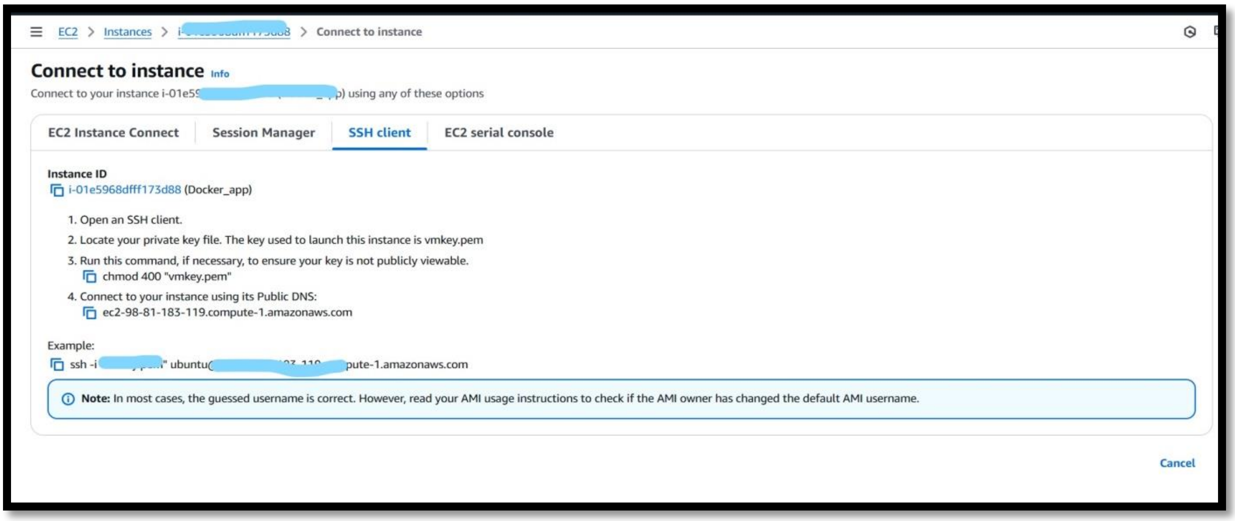
Find Instance by attribute or tag (case-sensitive)

All states ▼

☑	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
☑	Docker_app	i-01e5968dfff173d88	✓ Running	t2.micro	✓ 2/2 checks passec	View alarms +	us-east-1b	ec2-98-8

4.5: SSH Configuration

Log in ec2 instance using SSH client.



Connect to instance Info

Connect to your instance i-01e5[redacted]p) using any of these options

Instance ID
i-01e5968dfff173d88 (Docker_app)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is vmkey.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
   ```
   chmod 400 "vmkey.pem"
   ```
4. Connect to your instance using its Public DNS:
   ```
   ec2-98-81-183-119.compute-1.amazonaws.com
   ```

Example:

```
ssh -i [redacted]" ubuntu[redacted]pute-1.amazonaws.com
```

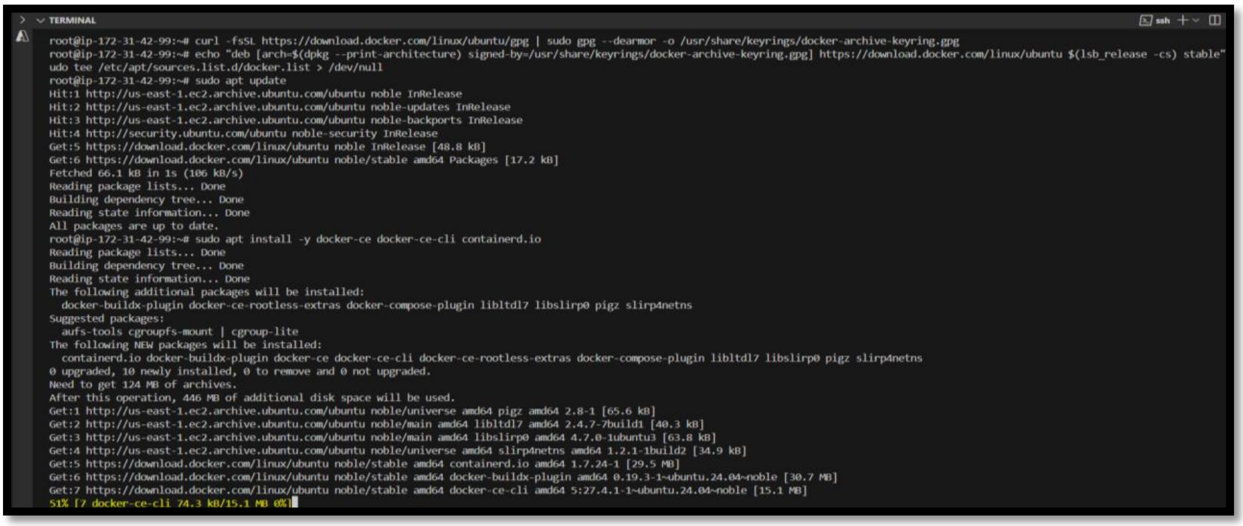
ⓘ **Note:** In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

Successfully able to connect the ec2-Instance by using ssh client.

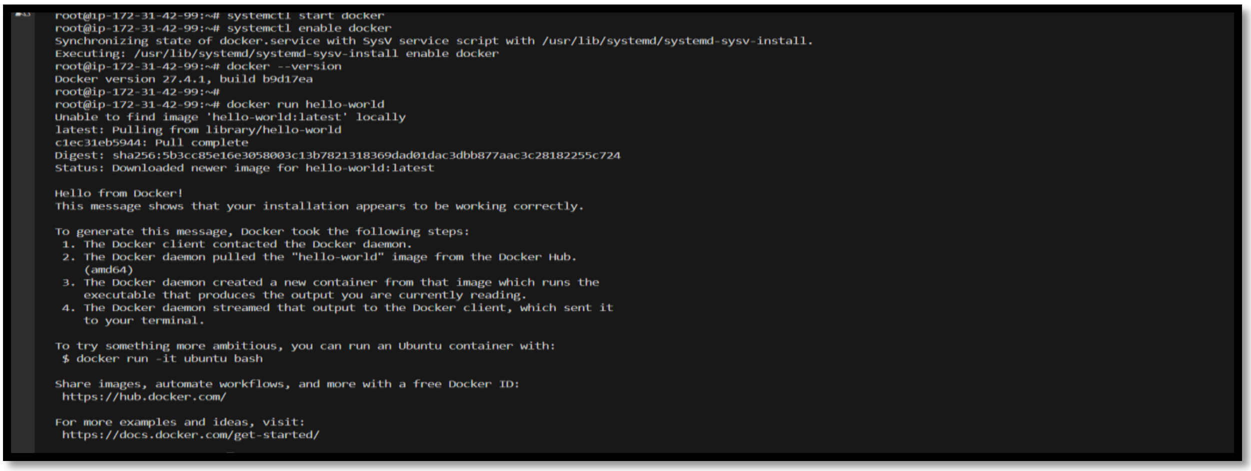
4.6: Install Dockerand Docker-compose on Ubuntu Server

To install Docker and Docker-compose on Ubuntu we need to follow below commands:

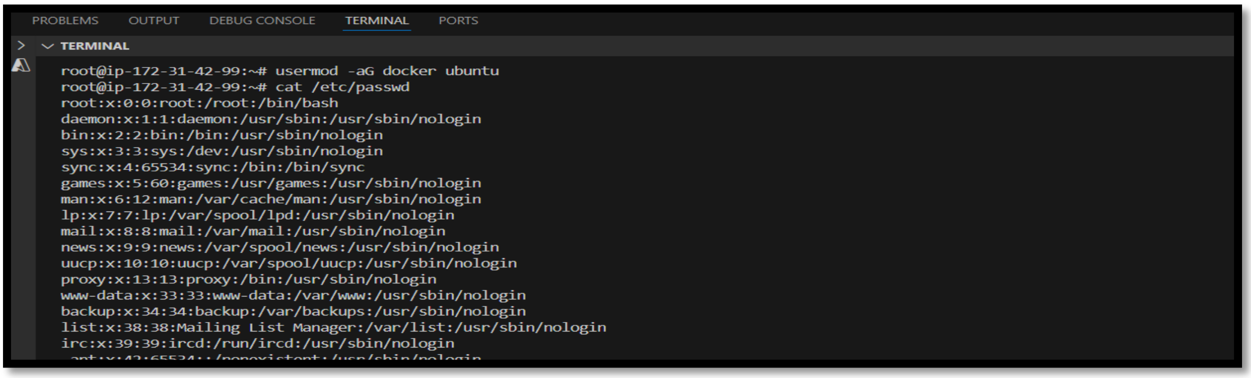
```
# Update system
sudo apt update -y

# Install Docker
sudo apt install -y docker
sudo systemctl start docker
sudo systemctl enable docker
sudo usermod -aG docker ubuntu

# Install Docker Compose
sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-
$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose

# Verify installations
docker --version
docker-compose --version
```



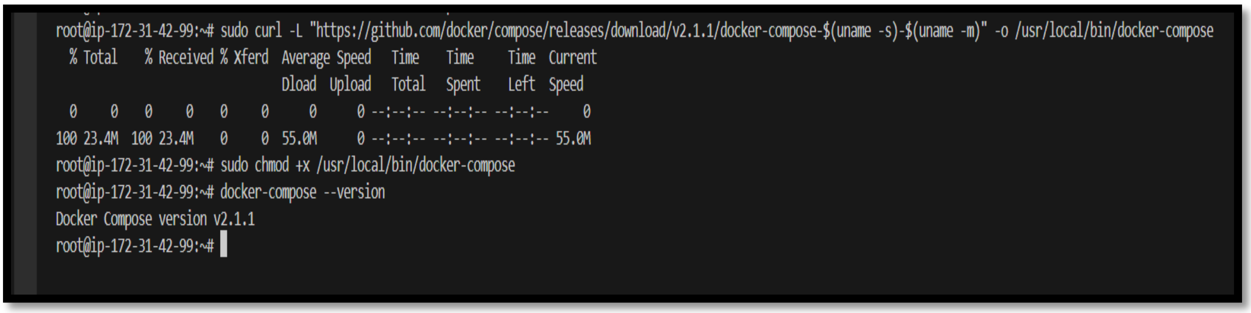
```
root@ip-172-31-42-99:~# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
root@ip-172-31-42-99:~# echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
udo tee /etc/apt/sources.list.d/docker.list > /dev/null
root@ip-172-31-42-99:~# sudo apt update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:5 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]
Get:6 https://download.docker.com/linux/ubuntu noble/stable amd64 Packages [17.2 kB]
Fetched 66.1 kB in 1s (106 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
root@ip-172-31-42-99:~# sudo apt install -y docker-ce docker-ce-cli containerd.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  docker-buildx-plugin docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz slirp4netns
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz slirp4netns
0 upgraded, 10 newly installed, 0 to remove and 0 not upgraded.
Need to get 124 MB of archives.
After this operation, 446 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65.6 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libltdl7 amd64 2.4.7-7build1 [40.3 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libslirp0 amd64 4.7.0-1ubuntu3 [63.8 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 slirp4netns amd64 1.2.1-1build2 [34.9 kB]
Get:5 https://download.docker.com/linux/ubuntu noble/stable amd64 containerd.io amd64 1.7.24-1 [29.5 MB]
Get:6 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-buildx-plugin amd64 0.19.3-1~ubuntu.24.04~noble [30.7 MB]
Get:7 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-ce-cli amd64 5:27.4.1-1~ubuntu.24.04~noble [15.1 MB]
51% [7 docker-ce-cli 74.3 kB/15.1 MB 0%]
```



```
root@ip-172-31-42-99:~# systemctl start docker
root@ip-172-31-42-99:~# systemctl enable docker
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
root@ip-172-31-42-99:~# docker --version
Docker version 27.4.1, build b9d17ea
root@ip-172-31-42-99:~#
root@ip-172-31-42-99:~# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:5b3cc85e16e3058003c13b7821318369dad01dac3dbb877aac3c28182255c724
Status: Downloaded newer image for hello-world:latest

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 it
    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/
```



```
root@ip-172-31-42-99:~# usermod -aG docker ubuntu
root@ip-172-31-42-99:~# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
_apt:x:42:65534::/nonexistent:/usr/sbin/nologin
```



```
root@ip-172-31-42-99:~# sudo curl -L "https://github.com/docker/compose/releases/download/v2.1.1/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 23.4M  100 23.4M    0     0  55.0M      0 --:--:-- --:--:-- --:--:-- 55.0M
root@ip-172-31-42-99:~# sudo chmod +x /usr/local/bin/docker-compose
root@ip-172-31-42-99:~# docker-compose --version
Docker Compose version v2.1.1
root@ip-172-31-42-99:~# 
```

5.0: Set Up Your Project

5.1: Create a Directory**:**

mkdir my-docker-compose-project
cd my-docker-compose-project

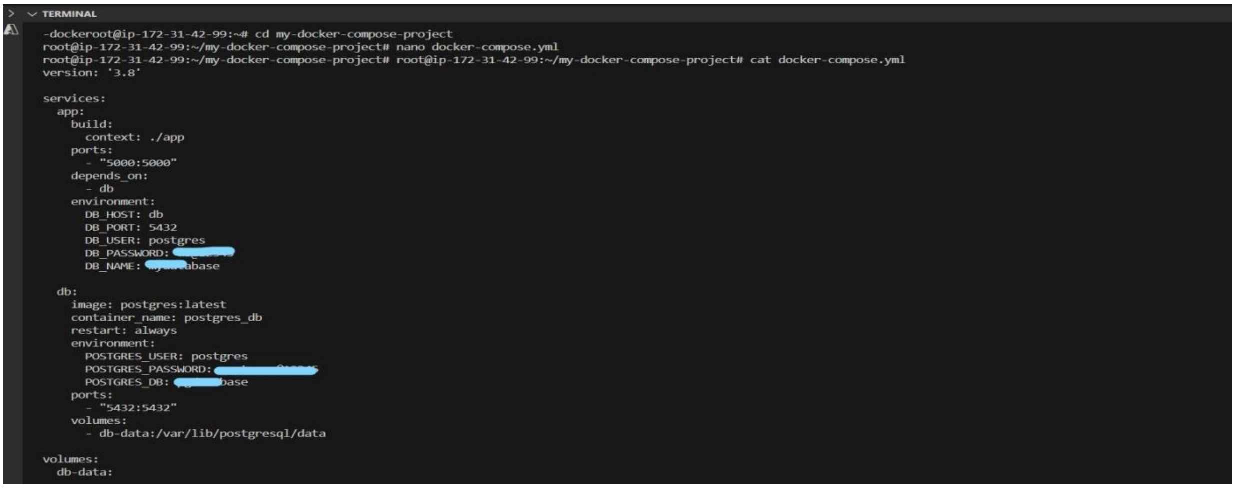
5.2: Create the **docker-compose.yml** File**:**

Used of docker-composed file for:

 Manage Multi-Container Applications: Simplifies the setup and coordination of multiple services (e.g., web app, database).

 Declarative Configuration: Defines services, networks, and volumes in a clean, reusable format.

```
nano docker-compose.yml
```



```
-dockeroot@ip-172-31-42-99:~# cd my-docker-compose-project
root@ip-172-31-42-99:~/my-docker-compose-project# nano docker-compose.yml
root@ip-172-31-42-99:~/my-docker-compose-project# root@ip-172-31-42-99:~/my-docker-compose-project# cat docker-compose.yml
version: '3.8'

services:
  app:
    build:
      context: ./app
    ports:
      - "5000:5000"
    depends_on:
      - db
    environment:
      DB_HOST: db
      DB_PORT: 5432
      DB_USER: postgres
      DB_PASSWORD: [redacted]
      DB_NAME: [redacted]abase

  db:
    image: postgres:latest
    container_name: postgres_db
    restart: always
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: [redacted]
      POSTGRES_DB: [redacted]base
    ports:
      - "5432:5432"
    volumes:
      - db-data:/var/lib/postgresql/data

volumes:
  db-data:
```

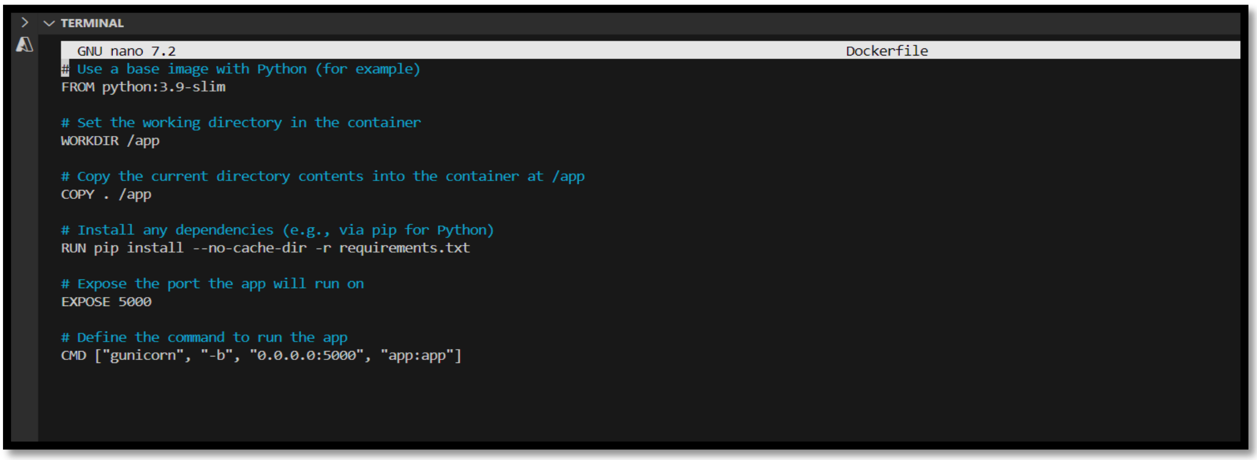
5.3: Create the Application Directory**:**

```
mkdir app
```

5.4: Create the Application Dockerfile (**app/Dockerfile**)**:**

A **Dockerfile** is used to automate the creation of Docker images by defining all the steps, configurations, and dependencies required to set up a containerized application.

```
nano Dockerfile
```



```
  GNU nano 7.2                                                   Dockerfile
# Use a base image with Python (for example)
FROM python:3.9-slim

# Set the working directory in the container
WORKDIR /app

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

# Install any dependencies (e.g., via pip for Python)
RUN pip install --no-cache-dir -r requirements.txt

# Expose the port the app will run on
EXPOSE 5000

# Define the command to run the app
CMD ["gunicorn", "-b", "0.0.0.0:5000", "app:app"]
```

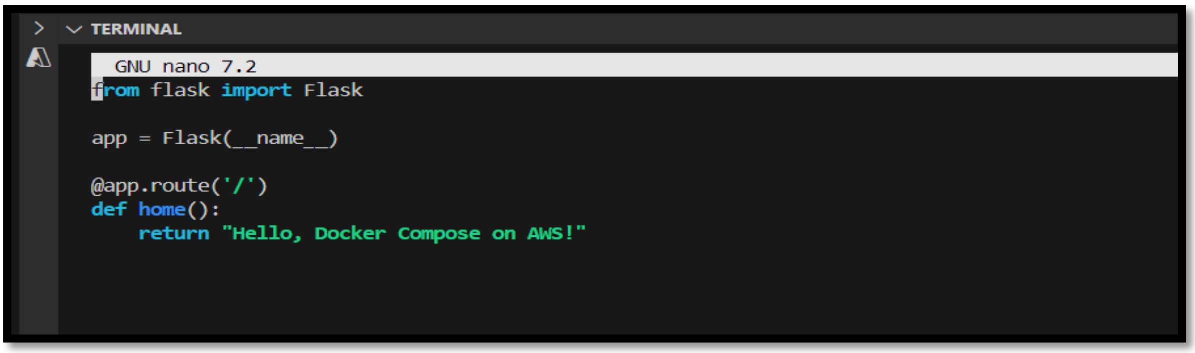
5.5: Create the Application Code (**app/app.py**)**:**

The **app.py** file is typically the entry point for a Python application. It serves as the main script that initializes and runs your application.

Key Benefits:

1. **Application Logic**: Contains the core code for the app, such as routing, logic, or processing.
2. **Framework Integration**: Often used to define and run frameworks like Flask or FastAPI.
3. **Entry Point**: Acts as the script that starts the app when executed (python app.py).
4. **Customizable Behavior**: Handles configuration, middleware, and service integration.

nano app.py

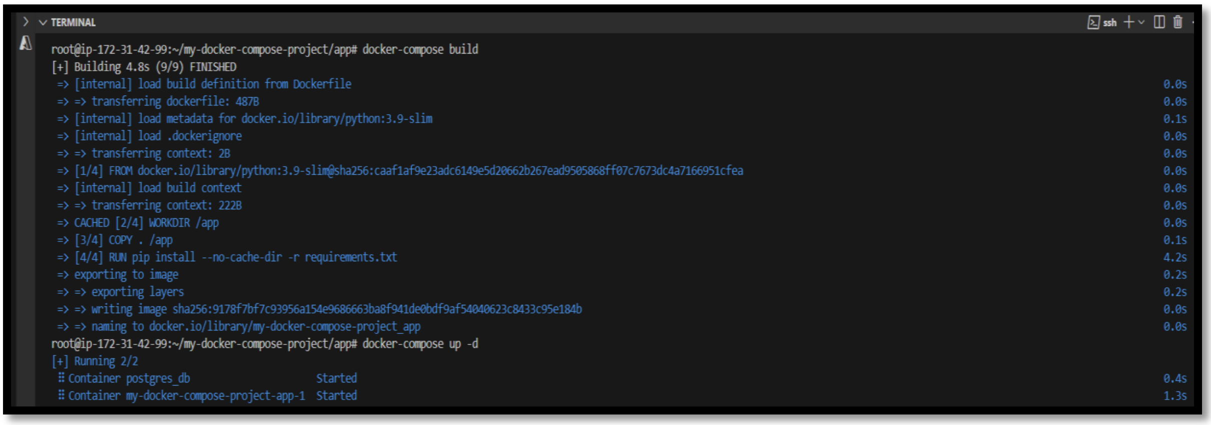


```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def home():
    return "Hello, Docker Compose on AWS!"
```

6.0: Deploy the Application

6.1: Run Docker Compose:

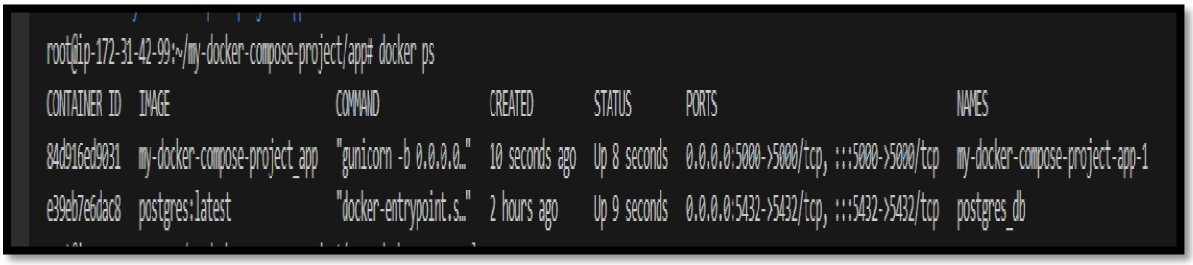
docker-compose build
docker-compose up –d



```
root@ip-172-31-42-99:~/my-docker-compose-project/app# docker-compose build
[+] Building 4.8s (9/9) FINISHED
 => [internal] load build definition from Dockerfile                                            0.0s
 => => transferring dockerfile: 487B                                                            0.0s
 => [internal] load metadata for docker.io/library/python:3.9-slim                              0.1s
 => [internal] load .dockerignore                                                               0.0s
 => => transferring context: 2B                                                                 0.0s
 => [1/4] FROM docker.io/library/python:3.9-slim@sha256:caaf1af9e23adc6149e5d20662b267ead9505868ff07c7673dc4a7166951cfea  0.0s
 => [internal] load build context                                                               0.0s
 => => transferring context: 222B                                                               0.0s
 => CACHED [2/4] WORKDIR /app                                                                   0.0s
 => [3/4] COPY . /app                                                                           0.1s
 => [4/4] RUN pip install --no-cache-dir -r requirements.txt                                    4.2s
 => exporting to image                                                                          0.2s
 => => exporting layers                                                                         0.2s
 => => writing image sha256:9178f7bf7c93956a154e9686663ba8f941de0bdf9af54040623c8433c95e184b    0.0s
 => => naming to docker.io/library/my-docker-compose-project_app                                0.0s
root@ip-172-31-42-99:~/my-docker-compose-project/app# docker-compose up -d
[+] Running 2/2
 ⠿ Container postgres_db                  Started                                               0.4s
 ⠿ Container my-docker-compose-project-app-1  Started                                           1.3s
```

6.2: Verify the Containers:

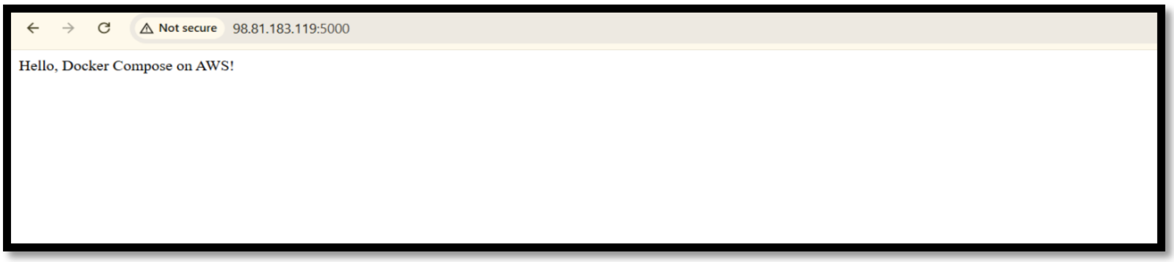
It will show status if our container and databse running or not docker-compose ps



```
root@ip-172-31-42-99:~/my-docker-compose-project/app# docker ps
CONTAINER ID   IMAGE                      COMMAND                  CREATED          STATUS         PORTS                                         NAMES
84d916ed9031   my-docker-compose-project_app   "gunicorn -b 0.0.0.0…"   10 seconds ago   Up 8 seconds   0.0.0.0:5000->5000/tcp, :::5000->5000/tcp   my-docker-compose-project-app-1
e39eb7e6dac8   postgres:latest            "docker-entrypoint.s…"   2 hours ago      Up 9 seconds   0.0.0.0:5432->5432/tcp, :::5432->5432/tcp   postgres_db
```

6.3: Test the Application:

- Visit http://localhost:5000 in your browser to see the Flask app running.



Hello, Docker Compose on AWS!

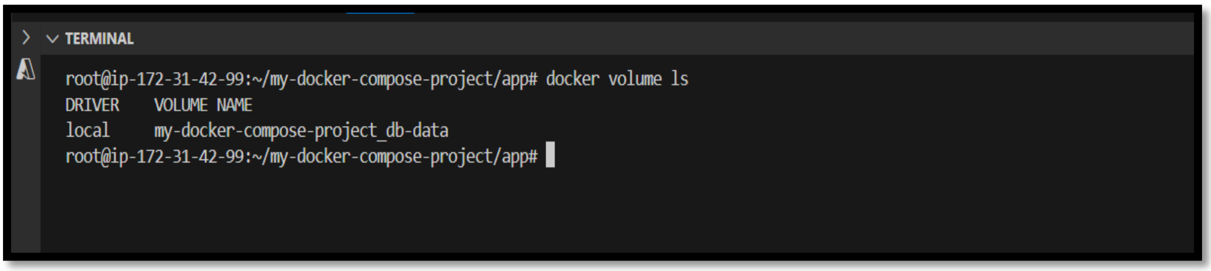
7.0: Persistent Storage:

Persistent storage refers to storage that retains data even after the application or system using it is stopped or restarted. It ensures that critical data is saved and can be retrieved later.
In Containers: Achieved using volumes.

The database data will persist in the Docker volume db-data.

7.1: Verify the volume:

docker volume ls

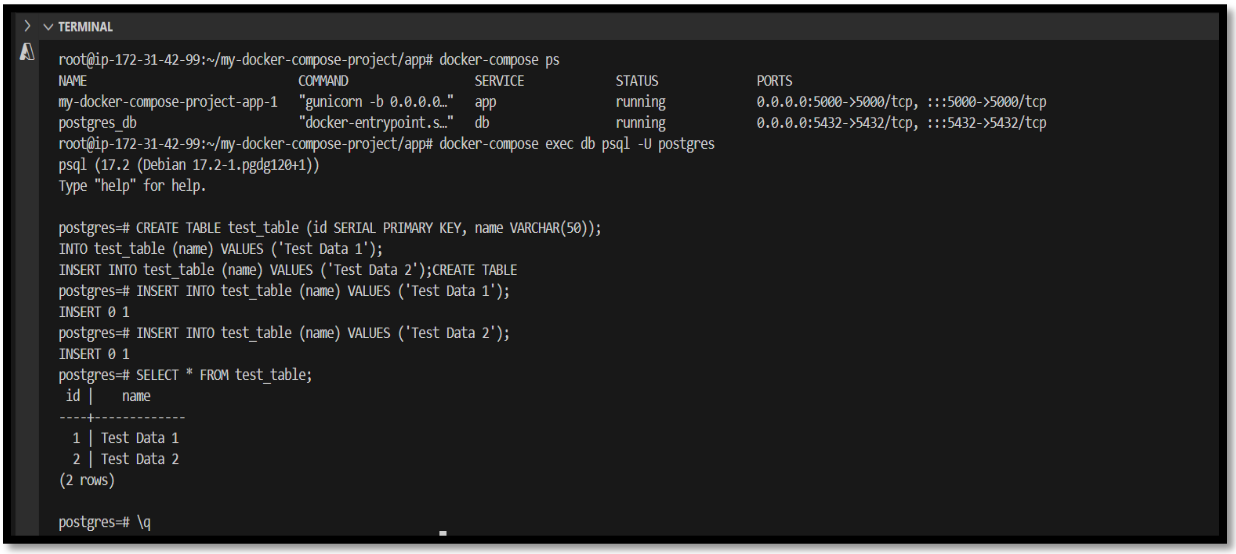


```
root@ip-172-31-42-99:~/my-docker-compose-project/app# docker volume ls
DRIVER    VOLUME NAME
local     my-docker-compose-project_db-data
root@ip-172-31-42-99:~/my-docker-compose-project/app# 
```

7.2: Persistent Data Test

To verify storage persistence:
First we need to add some data to our database

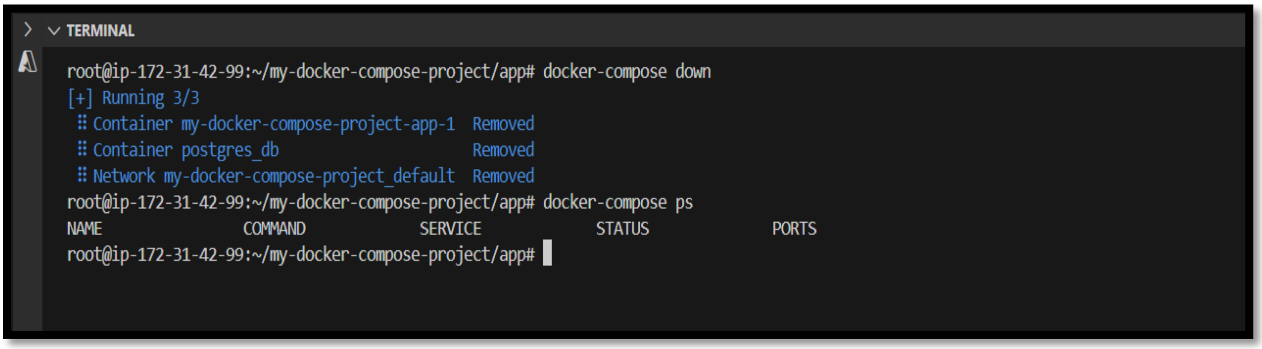
```
# Connect to your PostgreSQL container
docker-compose exec db psql -U postgres

# Inside PostgreSQL, create a test table and add data
CREATE TABLE test_table (id SERIAL PRIMARY KEY, name VARCHAR(50));
INSERT INTO test_table (name) VALUES ('Test Data 1');
INSERT INTO test_table (name) VALUES ('Test Data 2');

# Verify data is there
SELECT * FROM test_table;

# Exit PostgreSQL
\q
```



```
root@ip-172-31-42-99:~/my-docker-compose-project/app# docker-compose ps
NAME                             COMMAND                  SERVICE             STATUS              PORTS
my-docker-compose-project-app-1  "gunicorn -b 0.0.0.0…"   app                 running             0.0.0.0:5000->5000/tcp, :::5000->5000/tcp
postgres_db                      "docker-entrypoint.s…"   db                  running             0.0.0.0:5432->5432/tcp, :::5432->5432/tcp
root@ip-172-31-42-99:~/my-docker-compose-project/app# docker-compose exec db psql -U postgres
psql (17.2 (Debian 17.2-1.pgdg120+1))
Type "help" for help.

postgres=# CREATE TABLE test_table (id SERIAL PRIMARY KEY, name VARCHAR(50));
INTO test_table (name) VALUES ('Test Data 1');
INSERT INTO test_table (name) VALUES ('Test Data 2');CREATE TABLE
postgres=# INSERT INTO test_table (name) VALUES ('Test Data 1');
INSERT 0 1
postgres=# INSERT INTO test_table (name) VALUES ('Test Data 2');
INSERT 0 1
postgres=# SELECT * FROM test_table;
 id |    name
----+-------------
  1 | Test Data 1
  2 | Test Data 2
(2 rows)

postgres=# \q
```

7.3: Stop all containers**:**

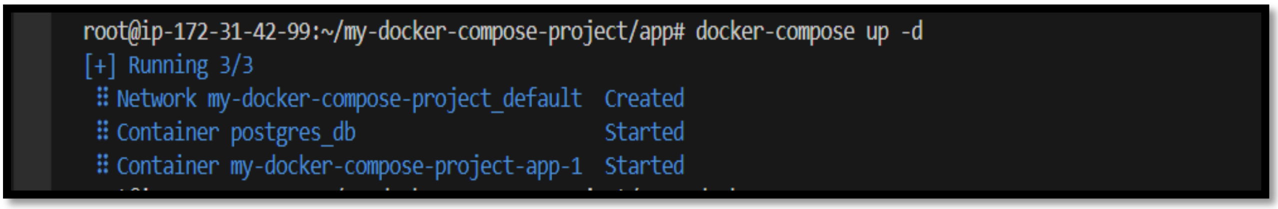
```
docker-compose down
```



```
root@ip-172-31-42-99:~/my-docker-compose-project/app# docker-compose down
[+] Running 3/3
 ⠿ Container my-docker-compose-project-app-1  Removed
 ⠿ Container postgres_db                      Removed
 ⠿ Network my-docker-compose-project_default  Removed
root@ip-172-31-42-99:~/my-docker-compose-project/app# docker-compose ps
NAME                COMMAND             SERVICE             STATUS              PORTS
root@ip-172-31-42-99:~/my-docker-compose-project/app#
```

7.4: Start services again**:**

docker-compose up -d



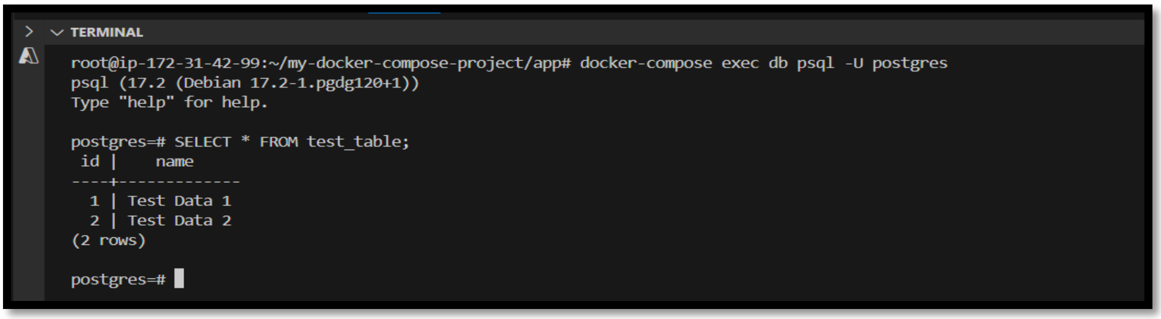
```
root@ip-172-31-42-99:~/my-docker-compose-project/app# docker-compose up -d
[+] Running 3/3
 ⠿ Network my-docker-compose-project_default  Created
 ⠿ Container postgres_db                      Started
 ⠿ Container my-docker-compose-project-app-1  Started
```

7.5: Verify data persisted:

```
# Connect to database again
docker-compose exec db psql -U postgres

# Check if data is still there
SELECT * FROM test_table;
```

You should see:

```
id |    name
----+-------------
  1 | Test Data 1
  2 | Test Data 2
```



```
root@ip-172-31-42-99:~/my-docker-compose-project/app# docker-compose exec db psql -U postgres
psql (17.2 (Debian 17.2-1.pgdg120+1))
Type "help" for help.

postgres=# SELECT * FROM test_table;
 id |    name
----+-------------
  1 | Test Data 1
  2 | Test Data 2
(2 rows)

postgres=# 
```

If you see your test data after restarting the containers, it confirms that:

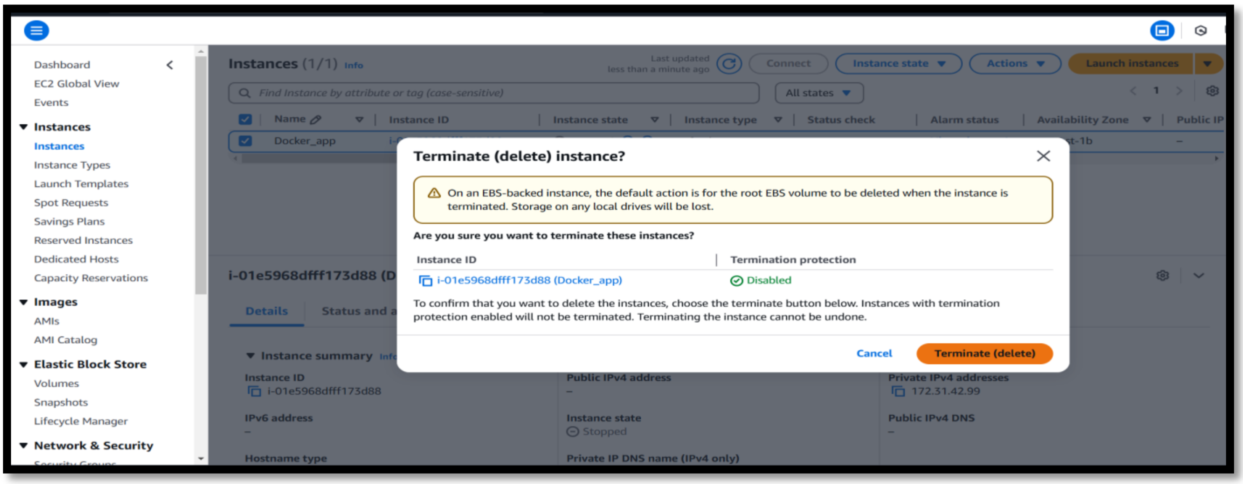
- Your volume is properly configured
- Data is being stored persistently
- Storage survives container restarts

8.0: Cleanup:

If we want to Stop and remove containers, networks, and volumes:

docker-compose down --volumes

Terminate the EC2 instance if no longer needed.



Terminate (delete) instance?

On an EBS-backed instance, the default action is for the root EBS volume to be deleted when the instance is terminated. Storage on any local drives will be lost.

Are you sure you want to terminate these instances?

Instance ID	Termination protection
i-01e5968dfff173d88 (Docker_app)	Disabled

To confirm that you want to delete the instances, choose the terminate button below. Instances with termination protection enabled will not be terminated. Terminating the instance cannot be undone.

Above mentioned Github link defines all files which need to complete all workflow .

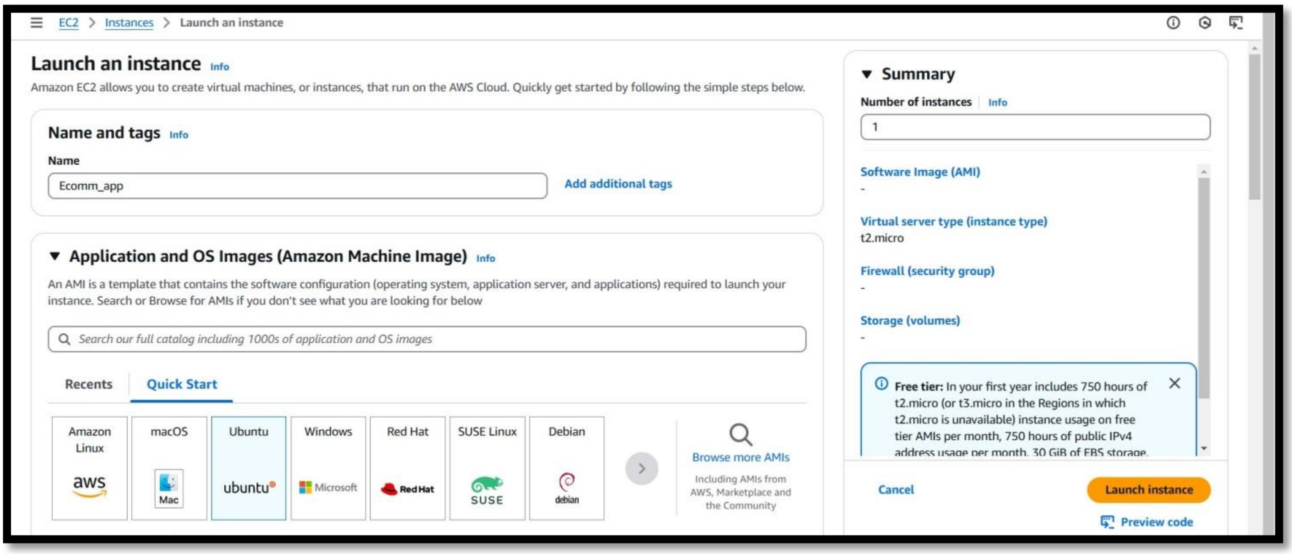
9.0 Document Overview: -> E-commerce application

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)

9.1 Launch ec2 intance:

Log in to the AWS Management Console and navigate to the EC2 dashboard.
Click on the "Launch Instance" button to start the process of launching a new EC2 instance.



EC2 > Instances > Launch an instance

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name: Ecomm_app

Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Summary

Number of instances Info: 1

Software Image (AMI): -

Virtual server type (instance type): t2.micro

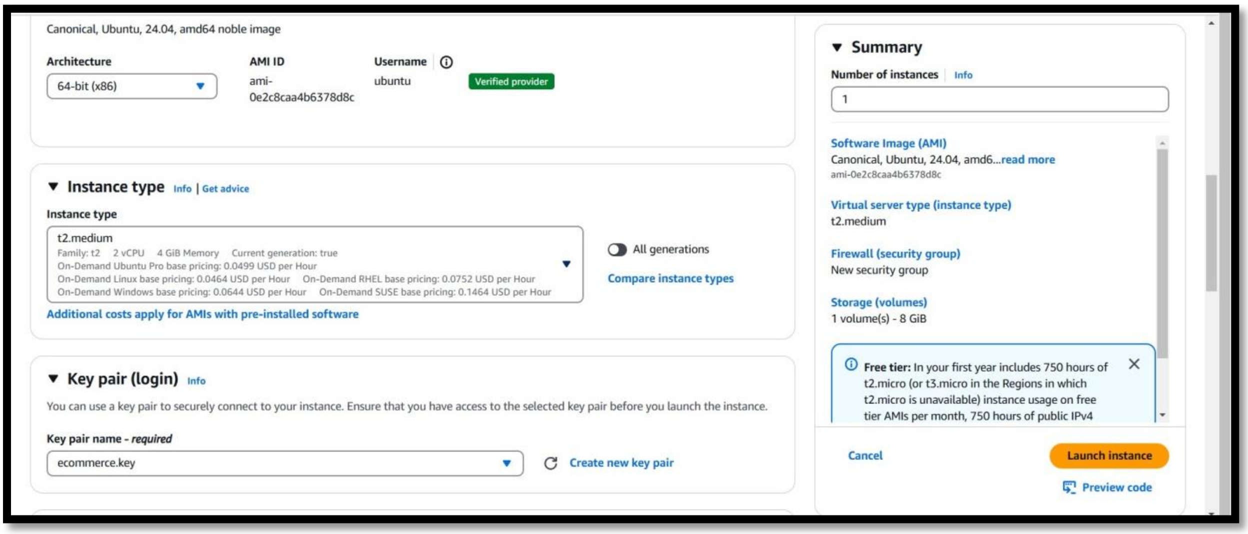
Firewall (security group): -

Storage (volumes): -

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 30 GiB of EBS storage,

9.2: Choose an Amazon Machine Image (AMI)

Select an instance type, configure your instance details (such as the number of instances, and storage) t2.medium.



Canonical, Ubuntu, 24.04, amd64 noble image

Architecture: 64-bit (x86)

AMI ID: ami-0e2c8caa4b6378d8c

Username: ubuntu

Verified provider

▼ Instance type Info | Get advice

Instance type

t2.medium
Family: t2 2 vCPU 4 GiB Memory Current generation: true
On-Demand Ubuntu Pro base pricing: 0.0499 USD per Hour
On-Demand Linux base pricing: 0.0464 USD per Hour On-Demand RHEL base pricing: 0.0752 USD per Hour
On-Demand Windows base pricing: 0.0644 USD per Hour On-Demand SUSE base pricing: 0.1464 USD per Hour

Additional costs apply for AMIs with pre-installed software

All generations

Compare instance types

▼ Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

ecommerce.key

Create new key pair

▼ Summary

Number of instances Info

1

Software Image (AMI)
Canonical, Ubuntu, 24.04, amd6...read more
ami-0e2c8caa4b6378d8c

Virtual server type (instance type)
t2.medium

Firewall (security group)
New security group

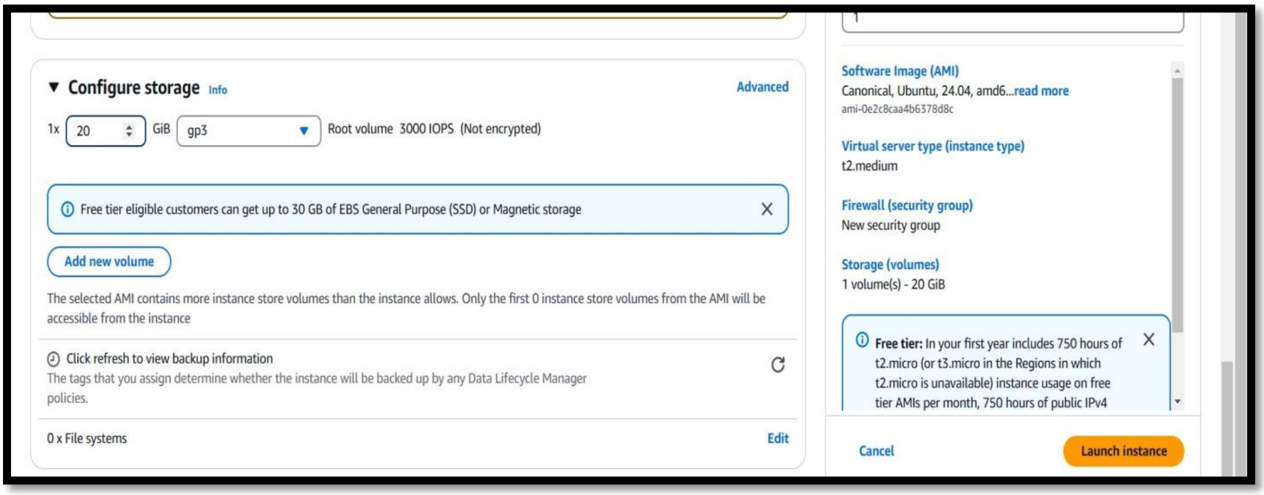
Storage (volumes)
1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4

Cancel

Launch instance

Preview code



▼ Configure storage Info

Advanced

1x 20 GiB gp3 Root volume 3000 IOPS (Not encrypted)

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage

Add new volume

The selected AMI contains more instance store volumes than the instance allows. Only the first 0 instance store volumes from the AMI will be accessible from the instance

Click refresh to view backup information
The tags that you assign determine whether the instance will be backed up by any Data Lifecycle Manager policies.

0 x File systems

Edit

1

Software Image (AMI)
Canonical, Ubuntu, 24.04, amd6...read more
ami-0e2c8caa4b6378d8c

Virtual server type (instance type)
t2.medium

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 20 GiB

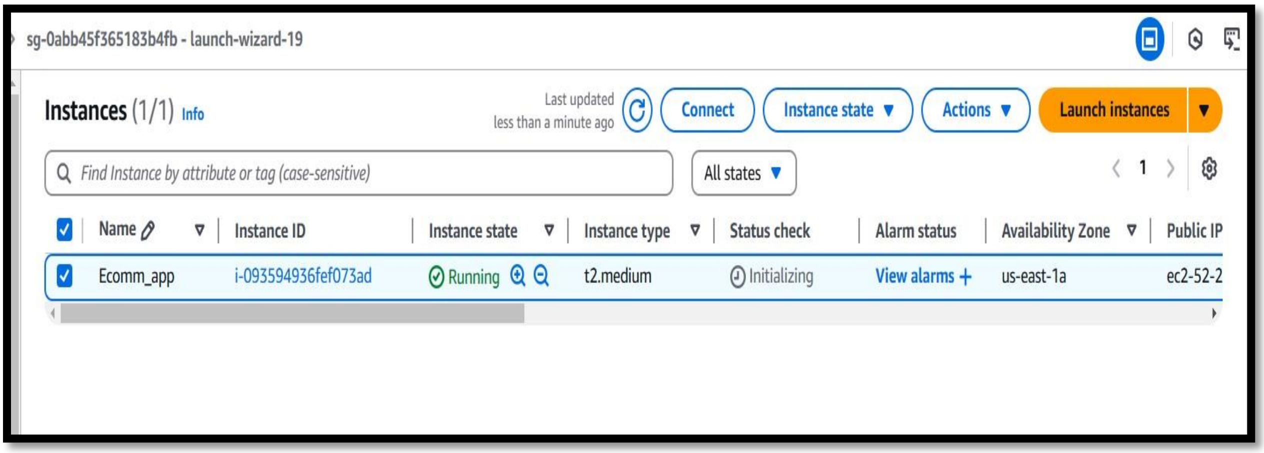
Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4

Cancel

Launch instance

9.3: Launch Instance

click on Launch instance



sg-0abb45f365183b4fb - launch-wizard-19

Instances (1/1) Info

Last updated less than a minute ago

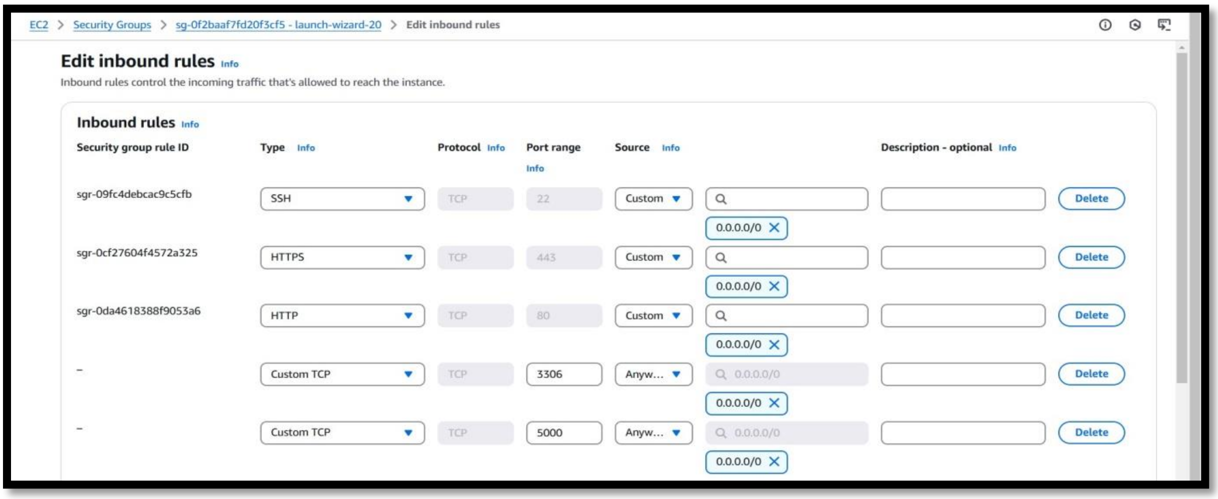
☑	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
☑	Ecomm_app	i-093594936fef073ad	Running	t2.medium	Initializing	View alarms +	us-east-1a	ec2-52-2

9.4: Expose port (3306,5000):

Go to security and then click to edit inbound rule and then add rule for

-> 3306 Used for database connections between applications and a MySQL server.

-> 5000 Used to run and serve web applications during development.



EC2 > Security Groups > sg-0f2baaf7fd20f3cf5 - launch-wizard-20 > Edit inbound rules

Edit inbound rules Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

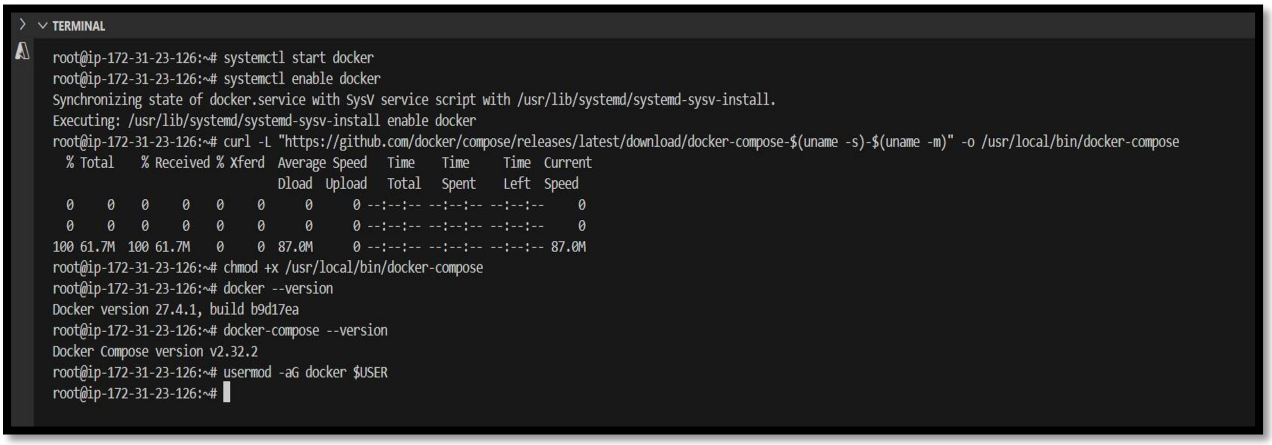
Inbound rules Info

Security group rule ID	Type	Protocol	Port range	Source		Description - optional	
sgr-09fc4debcac9c5cfb	SSH	TCP	22	Custom	0.0.0.0/0		Delete
sgr-0cf27604f4572a325	HTTPS	TCP	443	Custom	0.0.0.0/0		Delete
sgr-0da4618388f9053a6	HTTP	TCP	80	Custom	0.0.0.0/0		Delete
–	Custom TCP	TCP	3306	Anyw...	0.0.0.0/0		Delete
–	Custom TCP	TCP	5000	Anyw...	0.0.0.0/0		Delete

Allows necessary network access to your application while maintaining security.

9.5: update package and install docker and docker-compose:

```
# Update package database
sudo apt-get update

# Install required dependencies
sudo apt-get install -y apt-transport-https ca-certificates curl software-properties-common

# Add Docker's official GPG key
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

# Add Docker repository
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
$(lsb_release -cs) stable"

# Update package database again
sudo apt-get update

# Install Docker CE (Community Edition)
sudo apt-get install -y docker-ce

# Verify Docker installation
sudo docker --version
```



```
root@ip-172-31-23-126:~# systemctl start docker
root@ip-172-31-23-126:~# systemctl enable docker
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
root@ip-172-31-23-126:~# curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 61.7M  100 61.7M    0     0  87.0M      0 --:--:-- --:--:-- --:--:-- 87.0M
root@ip-172-31-23-126:~# chmod +x /usr/local/bin/docker-compose
root@ip-172-31-23-126:~# docker --version
Docker version 27.4.1, build b9d17ea
root@ip-172-31-23-126:~# docker-compose --version
Docker Compose version v2.32.2
root@ip-172-31-23-126:~# usermod -aG docker $USER
root@ip-172-31-23-126:~#
```

9.6: project structure:

```
ecommerce-app/
├── app.py
├── database/
├── static/
│   ├── css/
│   │   └── style.css
│   ├── js/
│   │   ├── main.js
│   │   └── orders.js
│   └── images/
├── templates/
│   ├── index.html
│   └── orders.html
├── requirements.txt
├── Dockerfile
├── docker-compose.yml
├── .gitignore
├── .env
└── README.md
```

9.7: Create Ecommerce_app directory:

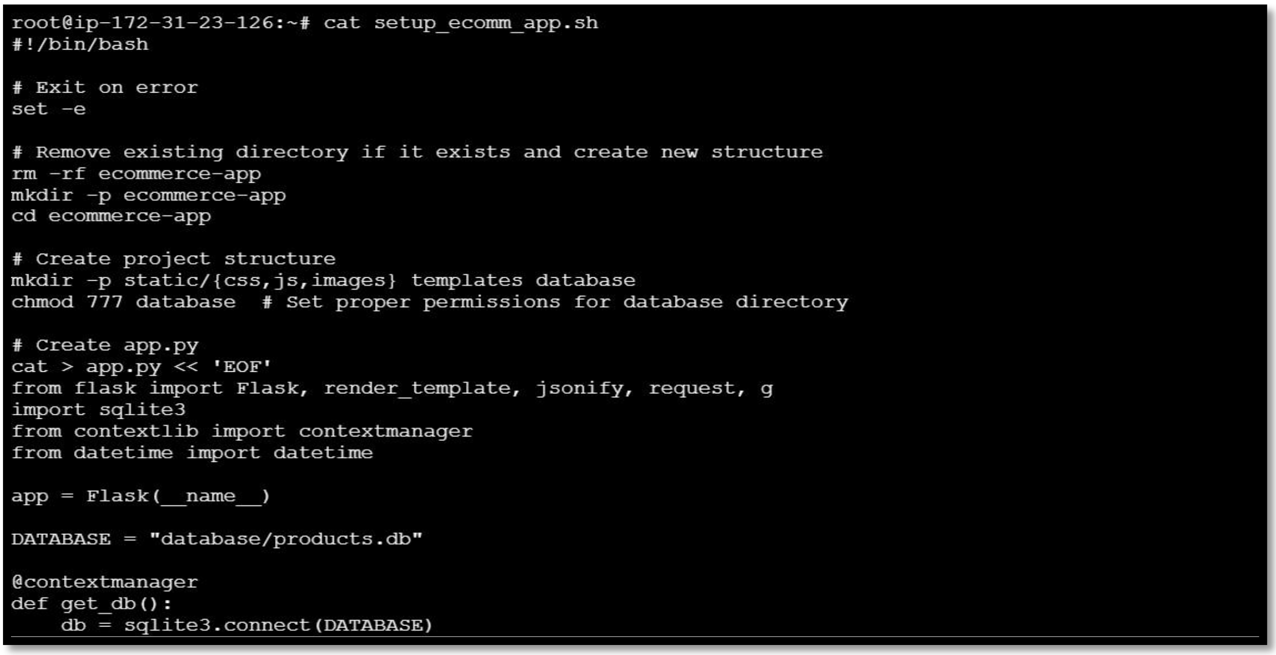
Creates a dedicated workspace for the application and navigate into that directory

```
Mkdir ecommerce_app
Cd ecommerce_app
```

9.8: Create Automation Script:

Create Automation script “setup_ecomm_app.sh” which includes all files which need to run ecommerce application.

```
nano setup_ecomm_app.sh
```



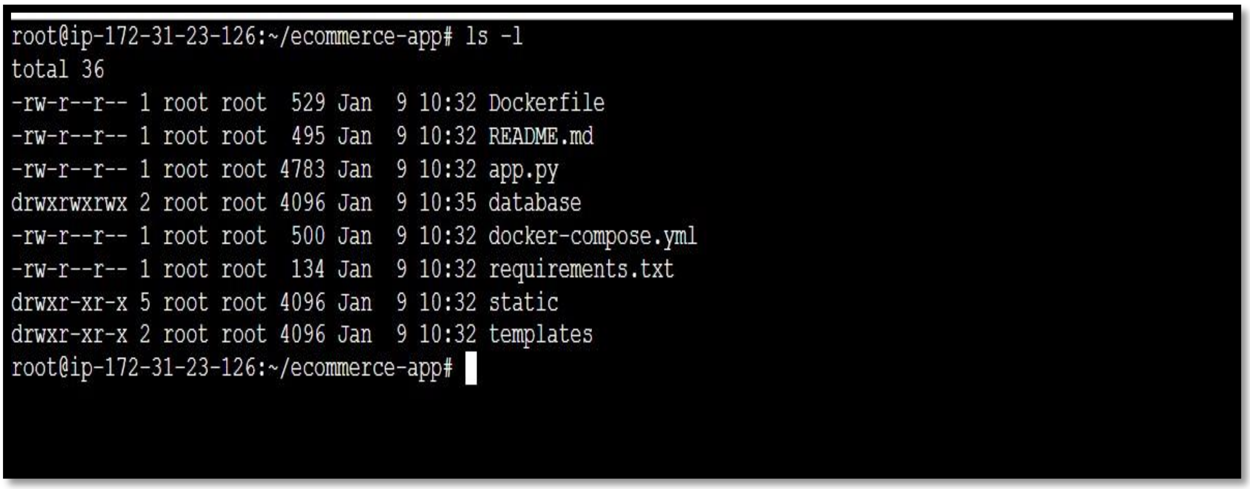
```
root@ip-172-31-23-126:~# cat setup_ecomm_app.sh
#!/bin/bash

# Exit on error
set -e

# Remove existing directory if it exists and create new structure
rm -rf ecommerce-app
mkdir -p ecommerce-app
cd ecommerce-app

# Create project structure
mkdir -p static/{css,js,images} templates database
chmod 777 database  # Set proper permissions for database directory

# Create app.py
cat > app.py << 'EOF'
from flask import Flask, render_template, jsonify, request, g
import sqlite3
from contextlib import contextmanager
from datetime import datetime

app = Flask(__name__)

DATABASE = "database/products.db"

@contextmanager
def get_db():
    db = sqlite3.connect(DATABASE)
```

Copy the entire script content from the provided file on Github into setup_ecomm_app.sh

# Make the script executable
chmod +x setup_ecomm_app.sh

 # Run the script
./setup_ecomm_app.sh

Prepares and executes the automation script that creates the application structure.
Once Run this setup_ecomm_app.sh script it will create all files in ecommerce_app folder
Once we do ls –l we can see all files.



```
root@ip-172-31-23-126:~/ecommerce-app# ls -l
total 36
-rw-r--r-- 1 root root  529 Jan  9 10:32 Dockerfile
-rw-r--r-- 1 root root  495 Jan  9 10:32 README.md
-rw-r--r-- 1 root root 4783 Jan  9 10:32 app.py
drwxrwxrwx 2 root root 4096 Jan  9 10:35 database
-rw-r--r-- 1 root root  500 Jan  9 10:32 docker-compose.yml
-rw-r--r-- 1 root root  134 Jan  9 10:32 requirements.txt
drwxr-xr-x 5 root root 4096 Jan  9 10:32 static
drwxr-xr-x 2 root root 4096 Jan  9 10:32 templates
root@ip-172-31-23-126:~/ecommerce-app#
```

All files and provided folders on Github. Follow above mentioned Github link.

10: Application Deployment:

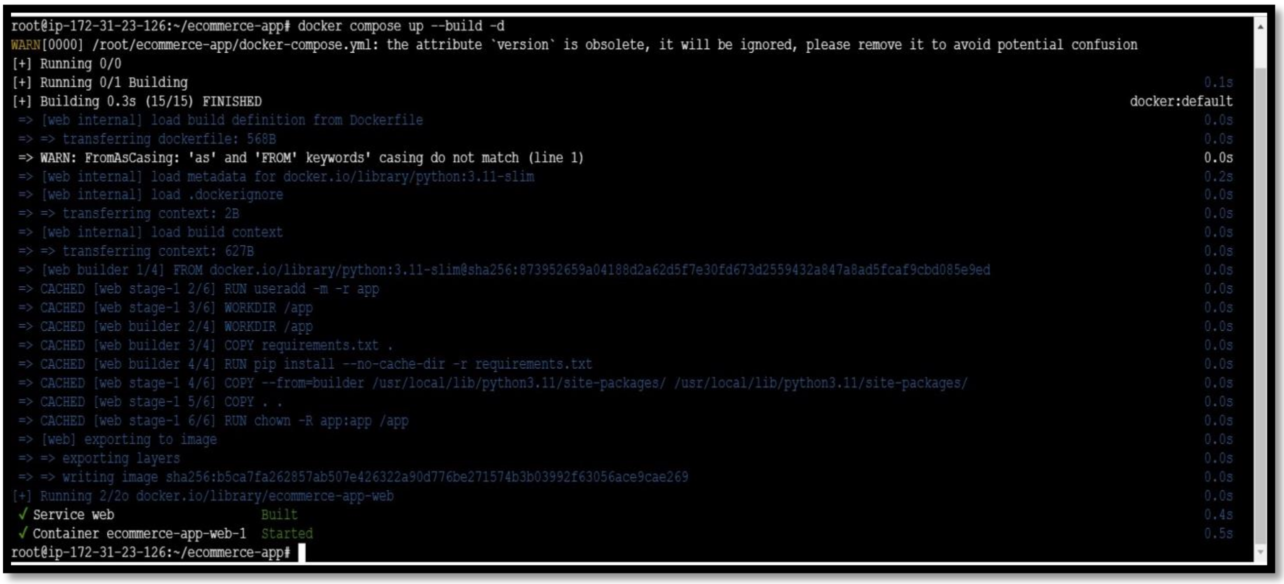
10.1. Build and Start Application

Navigate to ecommerce_App directory

```
cd ~/ecommerce-app
```

Build and start containers

```
docker-compose up --build -d
```



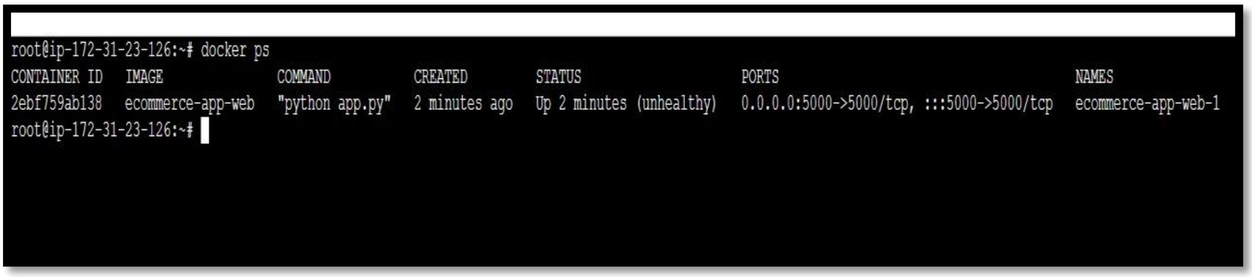
```
root@ip-172-31-23-126:~/ecommerce-app# docker compose up --build -d
WARN[0000] /root/ecommerce-app/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Running 0/0
[+] Running 0/1 Building                                                                                                                   0.1s
[+] Building 0.3s (15/15) FINISHED                                                                                               docker:default
 => [web internal] load build definition from Dockerfile                                                                                   0.0s
 => => transferring dockerfile: 568B                                                                                                       0.0s
 => WARN: FromAsCasing: 'as' and 'FROM' keywords' casing do not match (line 1)                                                            0.0s
 => [web internal] load metadata for docker.io/library/python:3.11-slim                                                                   0.2s
 => [web internal] load .dockerignore                                                                                                     0.0s
 => => transferring context: 2B                                                                                                           0.0s
 => [web internal] load build context                                                                                                     0.0s
 => => transferring context: 627B                                                                                                         0.0s
 => [web builder 1/4] FROM docker.io/library/python:3.11-slim@sha256:873952659a04188d2a62d5f7e30fd673d2559432a847a8ad5fcaf9cbd085e9ed      0.0s
 => CACHED [web stage-1 2/6] RUN useradd -m -r app                                                                                        0.0s
 => CACHED [web stage-1 3/6] WORKDIR /app                                                                                                 0.0s
 => CACHED [web builder 2/4] WORKDIR /app                                                                                                 0.0s
 => CACHED [web builder 3/4] COPY requirements.txt .                                                                                      0.0s
 => CACHED [web builder 4/4] RUN pip install --no-cache-dir -r requirements.txt                                                           0.0s
 => CACHED [web stage-1 4/6] COPY --from=builder /usr/local/lib/python3.11/site-packages/ /usr/local/lib/python3.11/site-packages/         0.0s
 => CACHED [web stage-1 5/6] COPY . .                                                                                                     0.0s
 => CACHED [web stage-1 6/6] RUN chown -R app:app /app                                                                                    0.0s
 => [web] exporting to image                                                                                                              0.0s
 => => exporting layers                                                                                                                   0.0s
 => => writing image sha256:b5ca7fa262857ab507e426322a90d776be271574b3b03992f63056ace9cae269                                              0.0s
[+] Running 2/2o docker.io/library/ecommerce-app-web                                                                                      0.0s
 ✓ Service web                      Built                                                                                                 0.4s
 ✓ Container ecommerce-app-web-1    Started                                                                                               0.5s
root@ip-172-31-23-126:~/ecommerce-app#
```

Deploys the application in detached mode using Docker containers.

5.2. Verify Deployment

Check container status

```
docker-compose ps
```



```
root@ip-172-31-23-126:~# docker ps
CONTAINER ID   IMAGE                COMMAND           CREATED         STATUS                       PORTS                                        NAMES
2ebf759ab138   ecommerce-app-web    "python app.py"   2 minutes ago   Up 2 minutes (unhealthy)     0.0.0.0:5000->5000/tcp, :::5000->5000/tcp    ecommerce-app-web-1
root@ip-172-31-23-126:~#
```

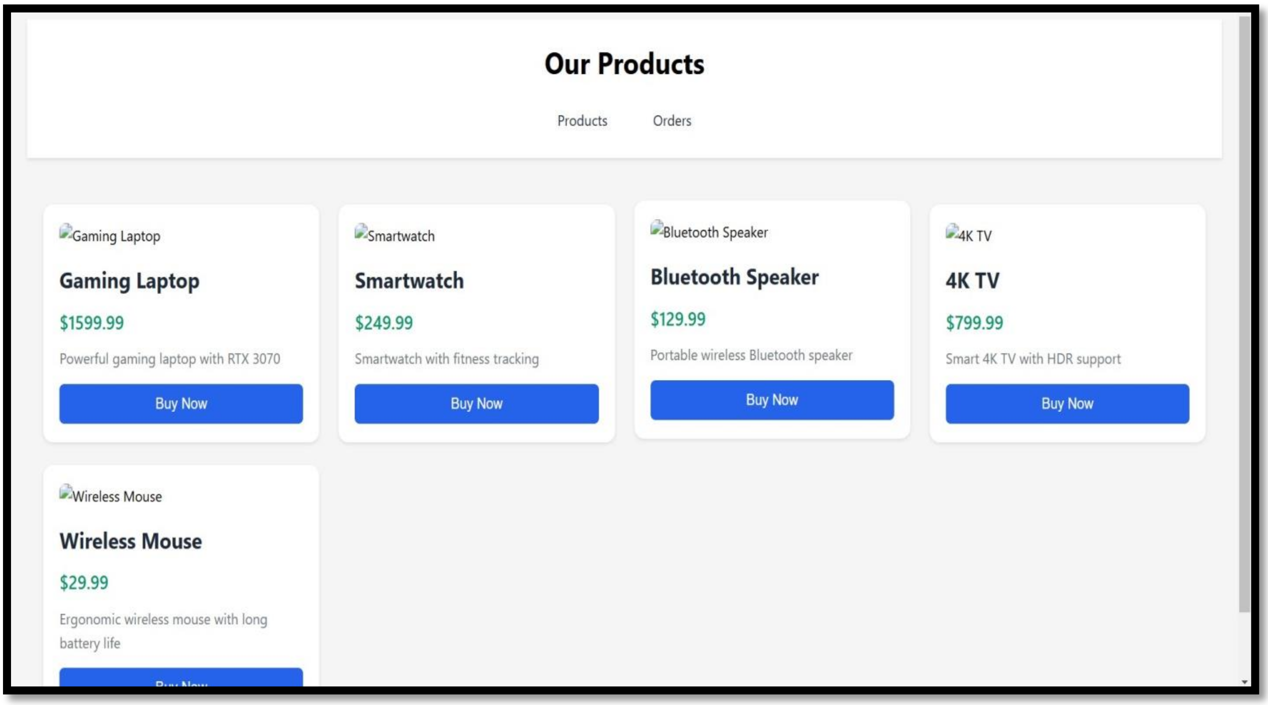
Confirms successful deployment and allows monitoring.

11: Accessing the application:

Through Ip address we can access our ecommerce application.

Via IP: http://your-server-ip:5000

It will showing product page



Our Products

Gaming Laptop

Gaming Laptop

$1599.99

Powerful gaming laptop with RTX 3070

Smartwatch

Smartwatch

$249.99

Smartwatch with fitness tracking

Bluetooth Speaker

Bluetooth Speaker

$129.99

Portable wireless Bluetooth speaker

4K TV

4K TV

$799.99

Smart 4K TV with HDR support

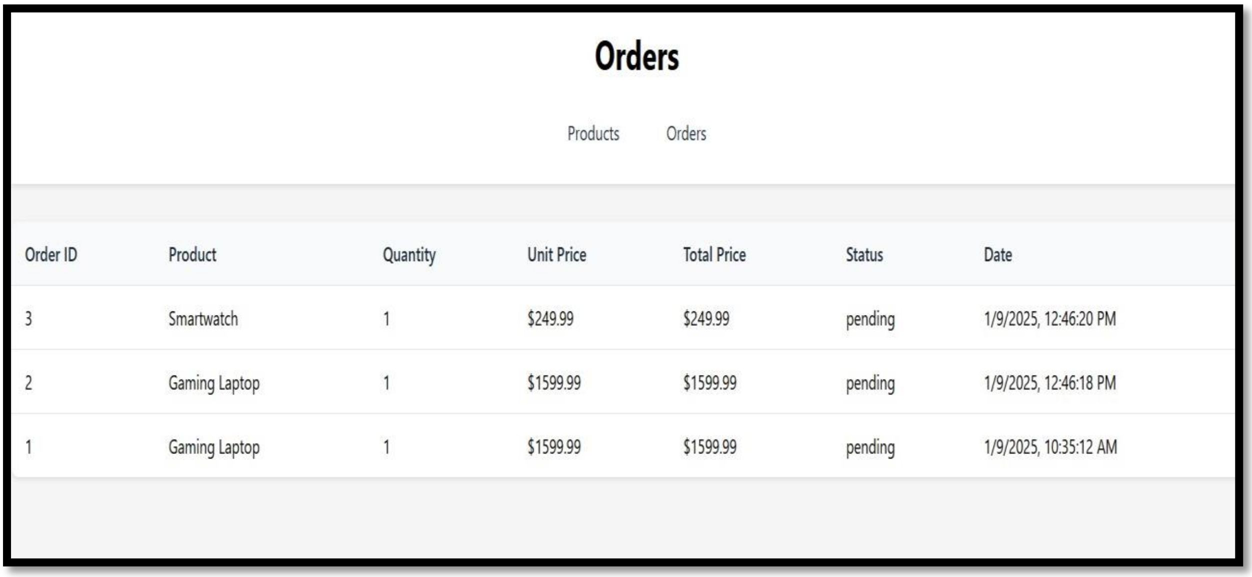
Wireless Mouse

Wireless Mouse

$29.99

Ergonomic wireless mouse with long battery life

If we click on buy now then it will navigate to order page. It will showing all entries which users clicked on buy now.



Orders

Order ID	Product	Quantity	Unit Price	Total Price	Status	Date
3	Smartwatch	1	$249.99	$249.99	pending	1/9/2025, 12:46:20 PM
2	Gaming Laptop	1	$1599.99	$1599.99	pending	1/9/2025, 12:46:18 PM
1	Gaming Laptop	1	$1599.99	$1599.99	pending	1/9/2025, 10:35:12 AM

This is complete setup process for deploying the e-commerce application on an AWS Ubuntu server, including security configurations, monitoring, and maintenance procedures.

12: Troubleshooting:

Common Issues and Solutions:

1. Application not accessible

Check if containers are running

```
docker-compose ps
```

2. Docker issues:

Restart Docker

```
sudo systemctl restart docker
```

# Rebuild application

```
docker-compose down && docker-compose up --build -d
```