



School of Computer Science Engineering and Application

BCA TY SEM VI

Subject Name: Container and Orchestration Practical

Assignment No. 8

Aim: Build Image with two dependencies (Flask, Redis) and create container with 5 replicas with docker stack.

Submitted By

Name: Jayesh Bhangale

PRN: 20210801024

Date: 16th April, 2024

Technology Used: Docker, AWS, Swarm, Docker Hub.

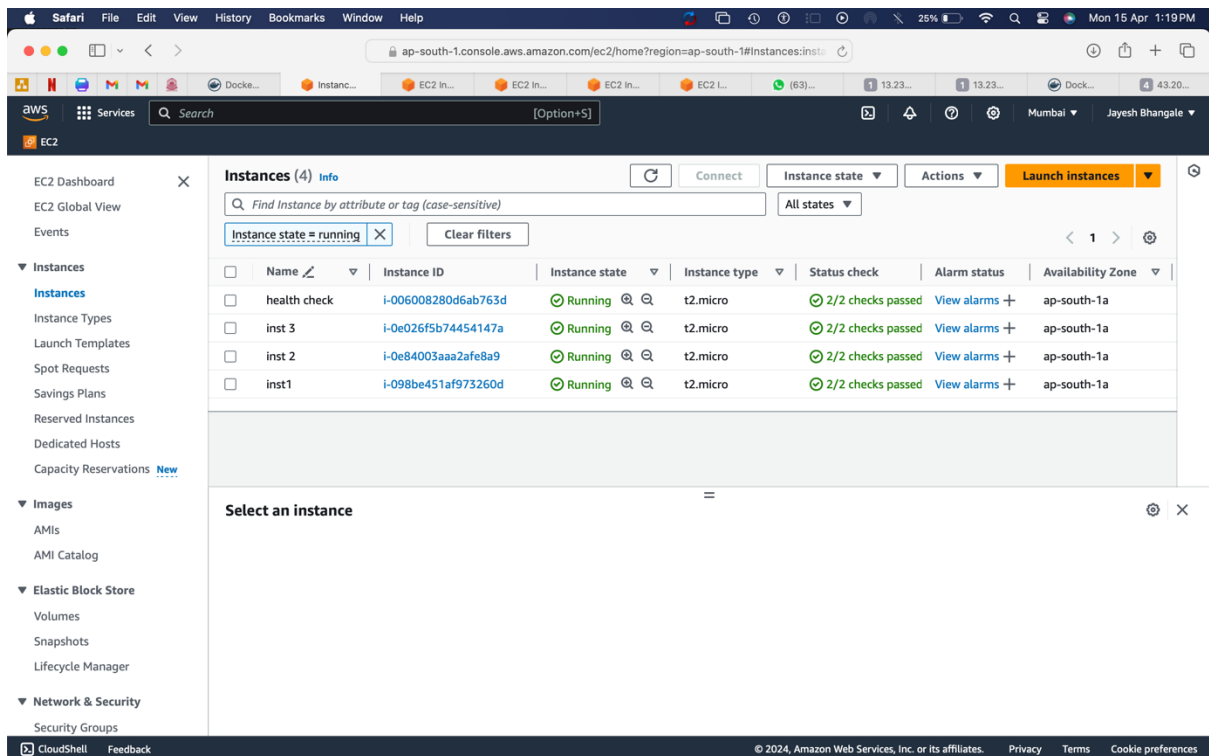
STEP -1: Create 3 instances, connect to it and install docker.

Sudo su

apt update -y

curl -fsSL <https://get.docker.com> -o get-docker.sh

sh get-docker.sh



STEP-2: Now Activate Docker swarm.

docker swarm init

```

aws
Services
Search
[Option+S]
Mumbai
Jayesh Bhangale
EC2
docker.service - Docker Application Container Engine
  Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
  Active: active (running) since Mon 2024-04-15 06:52:30 UTC; 2min 41s ago
  TriggeredBy: ● docker.socket
  Docs: https://docs.docker.com
  Main PID: 3259 (dockerd)
  Tasks: 8
  Memory: 103.5M
  CPU: 382ms
  CGroup: /system.slice/docker.service
          └─3259 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Apr 15 06:52:29 ip-172-31-44-47 systemd[1]: Starting Docker Application Container Engine...
Apr 15 06:52:30 ip-172-31-44-47 dockerd[3259]: time="2024-04-15T06:52:30.174111396Z" level=info msg="Starting up"
Apr 15 06:52:30 ip-172-31-44-47 dockerd[3259]: time="2024-04-15T06:52:30.184413138Z" level=info msg="detected 127.0.0.53 nameserver, assuming systemd-resolved"
Apr 15 06:52:30 ip-172-31-44-47 dockerd[3259]: time="2024-04-15T06:52:30.400641847Z" level=info msg="Loading containers: start."
Apr 15 06:52:30 ip-172-31-44-47 dockerd[3259]: time="2024-04-15T06:52:30.725067408Z" level=info msg="Loading containers: done."
Apr 15 06:52:30 ip-172-31-44-47 dockerd[3259]: time="2024-04-15T06:52:30.755193550Z" level=info msg="Docker daemon" commit=60b9add containerd-snapshotter=1.5.0-rc2
Apr 15 06:52:30 ip-172-31-44-47 dockerd[3259]: time="2024-04-15T06:52:30.756790001Z" level=info msg="Daemon has completed initialization"
Apr 15 06:52:30 ip-172-31-44-47 dockerd[3259]: time="2024-04-15T06:52:30.822594714Z" level=info msg="API listen on /run/docker.sock"
Apr 15 06:52:30 ip-172-31-44-47 systemd[1]: Started Docker Application Container Engine.
lines 1-21/21 (END)

root@ip-172-31-44-47:~# docker swarm init
Swarm initialized: current node (07d7v4l07raltzcgjw085r0h) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-4uhw6fu0zwmzbzqfkbw68s25p6tzeak3goubge9tv18x39razs-8rolhpuaa0r9lrmv2gy9av6bj 172.31.44.47:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

root@ip-172-31-44-47:~#

i-098be451af973260d (inst1)
PublicIPs: 13.235.42.231 PrivateIPs: 172.31.44.47
CloudShell Feedback
© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

```

STEP-3: Now create a directory and upload the required files.

```
mkdir webapp
```

```
nano app.py
```

```
from flask import Flask
```

```
from redis import Redis, RedisError
```

```
import os
```

```
import socket
```

```
# Connect to Redis
```

```
redis = Redis(host="redis", db=0, socket_connect_timeout=2, socket_timeout=2)
```

```
app = Flask(__name__)
```

```
@app.route("/")
```

```
def hello():
```

```
    try:
```

```
        visits = redis.incr("counter")
```

```
    except RedisError:
```

```
        visits = "<i>cannot connect to Redis, counter disabled</i>"
```

```
    html = "<h3>Hello {name}</h3>" \
```

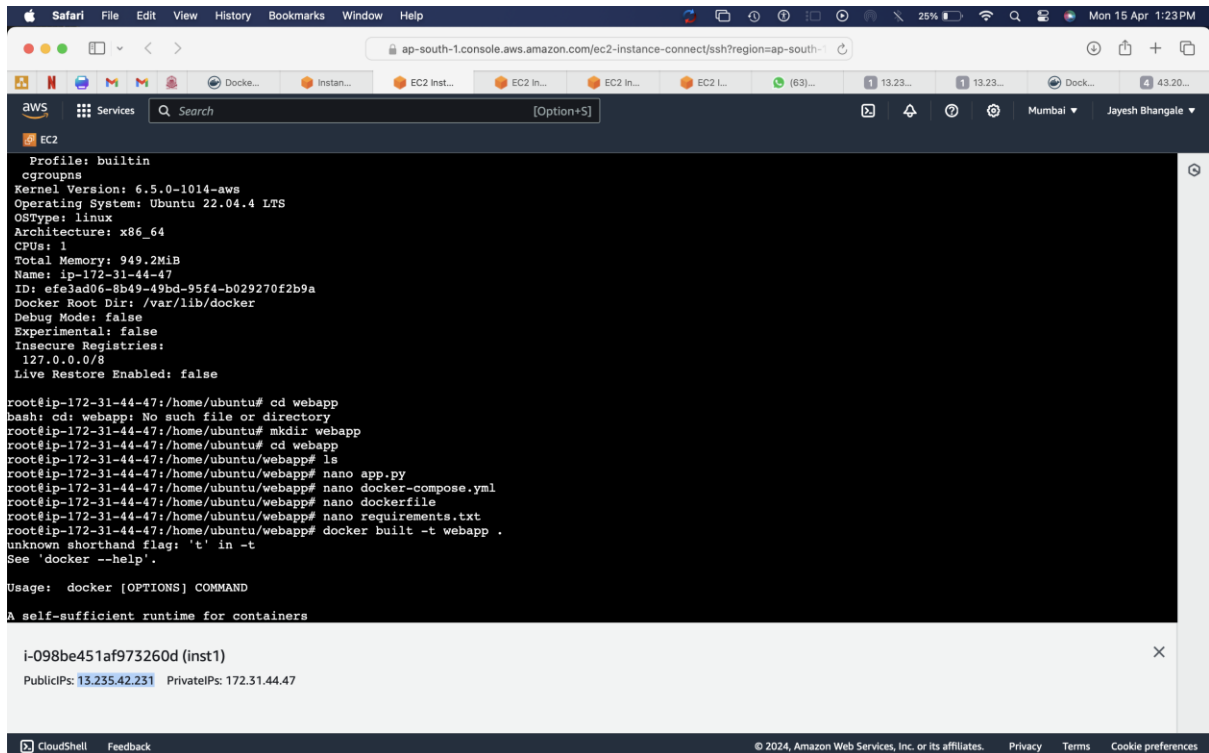
```
           "<b>Hostname:</b> {hostname}<br/>" \
```

```
           "<b>Visits:</b> {visits}"
```

```
return html.format(name=os.getenv("NAME", "world"),
hostname=socket.gethostname(), visits=visits)
```

```
if __name__ == "__main__":
```

```
    app.run(host='0.0.0.0', port=80)
```



nano docker-compose.yml

version: "3"

services:

Service Name Defined as web

web:

Pull the Image from Repository.

replace username/repo:tag with your name and image details

image: username/repo:tag

Command used to deploy the Service

deploy:

Run 5 instances of that image as a service called web

replicas: 5

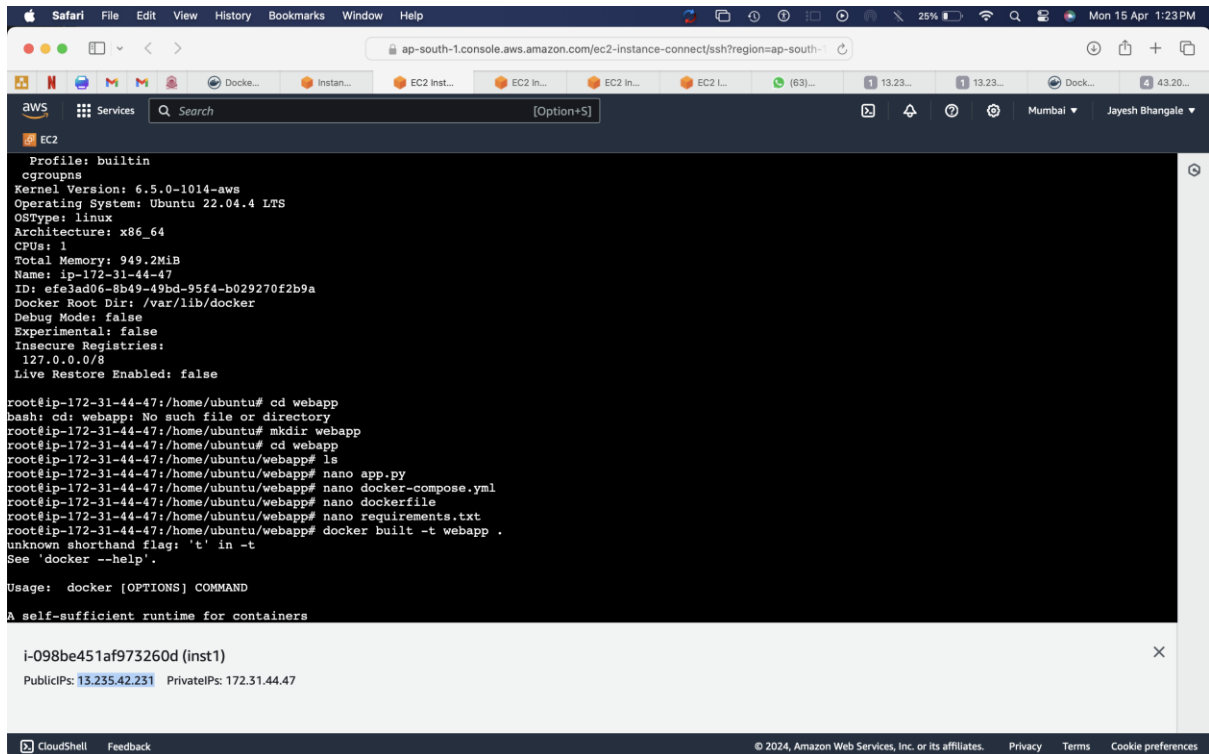
resources:

Limiting each one to use, at most, 10% of a single core of CPU time and 50MB of RAM.

```
limits:
  cpus: "0.1"
  memory: 50M
# Immediately restart containers if one fails.
restart_policy:
  condition: on-failure
# Map port 4000 on the host to web's port 80.
ports:
  - "4000:80"
# Define default network
networks:
  - webnet

redis:
  image: redis:latest
  ports:
    - "6379:6379"
  volumes:
    - "/app/redis_data:/data"
  deploy:
    placement:
      constraints: [node.hostname == manager2]
  command: redis-server --appendonly yes
  networks:
    - webnet

networks:
  webnet:
```



nano dockerfile

Use an official Python runtime as a parent image

FROM python:3.12-slim

Set the working directory to /app

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 --trusted-host pypi.python.org -r requirements.txt

Make port 80 available to the world outside this container

EXPOSE 80

Define environment variable

ENV NAME World

Run app.py when the container launches

CMD ["python", "app.py"]

nano requirements.txt

Flask

Redis

The screenshot shows a terminal window within the AWS CloudShell interface. The terminal output displays the EC2 instance profile, including kernel version (6.5.0-1014-aws), operating system (Ubuntu 22.04.4 LTS), and architecture (x86_64). The user is in the root shell at the IP address 172.31.44.47. They attempt to change to a directory named 'webapp', which does not exist. They then create the directory with 'mkdir webapp' and navigate into it. Subsequently, they create a Python application file 'app.py' and a Docker Compose file 'docker-compose.yml' using the 'nano' editor. They also create a 'requirements.txt' file. Finally, they attempt to build a Docker image named 'webapp' using 'docker build -t webapp', but receive an error: 'unknown shorthand flag: 't' in -t'. The terminal also shows the Docker usage instructions and a notification for the instance ID i-098be451af973260d (inst1) with its public and private IP addresses.

```

Profile: builtin
cgroups
Kernel Version: 6.5.0-1014-aws
Operating System: Ubuntu 22.04.4 LTS
OSType: linux
Architecture: x86_64
CPUs: 1
Total Memory: 949.2MiB
Name: ip-172-31-44-47
ID: efe3ad06-8b49-49bd-95f4-b029270f2b9a
Docker Root Dir: /var/lib/docker
Debug Mode: false
Experimental: false
Insecure Registries:
  127.0.0.0/8
Live Restore Enabled: false

root@ip-172-31-44-47:/home/ubuntu# cd webapp
bash: cd: webapp: No such file or directory
root@ip-172-31-44-47:/home/ubuntu# mkdir webapp
root@ip-172-31-44-47:/home/ubuntu# cd webapp
root@ip-172-31-44-47:/home/ubuntu/webapp# ls
root@ip-172-31-44-47:/home/ubuntu/webapp# nano app.py
root@ip-172-31-44-47:/home/ubuntu/webapp# nano docker-compose.yml
root@ip-172-31-44-47:/home/ubuntu/webapp# nano dockerfile
root@ip-172-31-44-47:/home/ubuntu/webapp# nano requirements.txt
root@ip-172-31-44-47:/home/ubuntu/webapp# docker build -t webapp .
unknown shorthand flag: 't' in -t
See 'docker --help'.

Usage: docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

i-098be451af973260d (inst1)
PublicIPs: 13.235.42.231 PrivateIPs: 172.31.44.47

```

STEP-4: Now build the image and upload it to your Docker Hub.

docker build -t webapp .

docker login

docker tag webapp aniket0724/webapp:1.0

docker push aniket0724/webapp:1.0

```

root@ip-172-31-44-47:/home/ubuntu/webapp# docker build -t webapp .
[+] Building 12.6s (9/9) FINISHED
=> [internal] load build definition from dockerfile
=> => transferring dockerfile: 553B
=> [internal] load metadata for docker.io/library/python:3.12-slim
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/4] FROM docker.io/library/python:3.12-slim@sha256:541d45d3d675fb8197f534525a671e2f8d66c882b89491f9dda271f4f94dcd06
=> => resolve docker.io/library/python:3.12-slim@sha256:541d45d3d675fb8197f534525a671e2f8d66c882b89491f9dda271f4f94dcd06
=> => sha256:78bef5c7424f06f0cb3d8d2ec97df139d1a617e758d07be642987bea4e18e337 12.00MB / 12.00MB
=> => sha256:541d45d3d675fb8197f534525a671e2f8d66c882b89491f9dda271f4f94dcd06 1.65kB / 1.65kB
=> => sha256:47fc40de7ad3d9690c3c4ebc0f93230d70c489b53e053e00e8abd327ac1137d4 1.37kB / 1.37kB
=> => sha256:0e42464fe231d538e4fb7a2c5047bd3721a9128553b6e7e8500fa06cf9b42d70 6.69kB / 6.69kB
=> => sha256:13808c22b207b066ef43572e57e4fb8c6172e887dd9a918c089a174a19371b7a 29.13MB / 29.13MB
=> => sha256:6c9a484475c10b31eadca58e66b24d9babf508955f52c40080a00595c55cc6c1 3.51MB / 3.51MB
=> => sha256:42f0d54f5caaaaf735fc348a42888969743393bf7b71563c5cbca3e40e1906f 245B / 245B
=> => extracting sha256:13808c22b207b066ef43572e57e4fb8c6172e887dd9a918c089a174a19371b7a
=> => sha256:1723cff2f16b53cb2ec17c6984f723b9f44a83a51946201ab455dd9678a322f2 2.98MB / 2.98MB
=> => extracting sha256:6c9a484475c10b31eadca58e66b24d9babf508955f52c40080a00595c55cc6c1
=> => extracting sha256:78bef5c7424f06f0cb3d8d2ec97df139d1a617e758d07be642987bea4e18e337
=> => extracting sha256:42f0d54f5caaaaf735fc348a42888969743393bf7b71563c5cbca3e40e1906f
=> => extracting sha256:1723cff2f16b53cb2ec17c6984f723b9f44a83a51946201ab455dd9678a322f2
=> [internal] load build context
=> => transferring context: 2.41kB
=> [2/4] WORKDIR /app
=> [3/4] COPY . /app
=> [4/4] RUN pip install --trusted-host pypi.python.org -r requirements.txt
=> exporting to image
=> exporting layers
=> writing image sha256:3b2dbad589778b370b33390859e35642f545c67e3fd3e26496b19716e97ac51f
=> naming to docker.io/library/webapp
root@ip-172-31-44-47:/home/ubuntu/webapp# docker login

i-098be451af973260d (inst1)
PublicIPs: 13.235.42.231 PrivateIPs: 172.31.44.47

```

STEP -5 : Now deploy the Stack.

docker stack deploy -c docker-compose.yml webapp

```

root@ip-172-31-44-47:/home/ubuntu/webapp# docker login
Log in with your Docker ID or email address to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com/ to create one.
You can log in with your password or a Personal Access Token (PAT). Using a limited-scope PAT grants better security and is required for organizations using SSO. Learn more at https://docs.docker.com/go/access-tokens/
Username: jayeshbhargale
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
root@ip-172-31-44-47:/home/ubuntu/webapp# docker tag webapp jayeshbhargale/webapp:1.0
root@ip-172-31-44-47:/home/ubuntu/webapp# docker push jayeshbhargale/webapp:1.0
The push refers to repository [docker.io/jayeshbhargale/webapp]
5a95cfd1d3dcb: Pushed
a10cc0dfdc6e: Pushed
13a5ffbc3b8: Pushed
6258d28909ff: Mounted from library/python
cfa35e1cecf: Mounted from library/python
c6bc67c64ae2: Mounted from library/python
bfc9081d1eb2: Mounted from library/python
1f00ff201478: Mounted from library/python
1.0: digest: sha256:f8ef30739994564258e6cc0eaa864ffa8495127fc0fbbe6c059bc6957d815f8 size: 1995
root@ip-172-31-44-47:/home/ubuntu/webapp# docker stack deploy -c docker-compose.yml webapp
Since --detach=false was not specified, tasks will be created in the background.
In a future release, --detach=false will become the default.
Creating network webapp_webnet
Creating service webapp_web
Creating service webapp_redis
root@ip-172-31-44-47:/home/ubuntu/webapp#

```

STEP-6: Now connect the other nodes to the manager node.

docker swarm join-token worker

(copy the token and paste it on the worker nodes)

The screenshot displays the AWS CloudShell interface with a terminal window showing the status of the Docker service and the initialization of a Docker Swarm.

Terminal Output:

```

EC2
docker.service - Docker Application Container Engine
  Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
  Active: active (running) since Mon 2024-04-15 06:52:30 UTC; 2min 41s ago
  TriggeredBy: ● docker.socket
  Docs: https://docs.docker.com
  Main PID: 3259 (dockerd)
  Tasks: 8
  Memory: 103.5M
  CPU: 382ms
  CGroup: /system.slice/docker.service
          └─3259 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Apr 15 06:52:29 ip-172-31-44-47 systemd[1]: Starting Docker Application Container Engine...
Apr 15 06:52:30 ip-172-31-44-47 dockerd[3259]: time="2024-04-15T06:52:30.174111396Z" level=info msg="Starting up"
Apr 15 06:52:30 ip-172-31-44-47 dockerd[3259]: time="2024-04-15T06:52:30.184413138Z" level=info msg="detected 127.0.0.53 nameserver, assuming systemd-resolved"
Apr 15 06:52:30 ip-172-31-44-47 dockerd[3259]: time="2024-04-15T06:52:30.400641847Z" level=info msg="Loading containers: start."
Apr 15 06:52:30 ip-172-31-44-47 dockerd[3259]: time="2024-04-15T06:52:30.725067408Z" level=info msg="Loading containers: done."
Apr 15 06:52:30 ip-172-31-44-47 dockerd[3259]: time="2024-04-15T06:52:30.755193550Z" level=info msg="Docker daemon" commit=60b9add containerd-snapshotter=
Apr 15 06:52:30 ip-172-31-44-47 dockerd[3259]: time="2024-04-15T06:52:30.756790001Z" level=info msg="Daemon has completed initialization"
Apr 15 06:52:30 ip-172-31-44-47 dockerd[3259]: time="2024-04-15T06:52:30.822594714Z" level=info msg="API listen on /run/docker.sock"
Apr 15 06:52:30 ip-172-31-44-47 systemd[1]: Started Docker Application Container Engine.
lines 1-21/21 (END)

root@ip-172-31-44-47:~# docker swarm init
Swarm initialized: current node (07d7v4l07raltzcgjw085r0h) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-4uhw6fu0zwmzbzqfkgbw68s25p6tzeak3goubge9tv18x39razs-8rolhpuaa0r9lrmv2gy9av6bj 172.31.44.47:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

root@ip-172-31-44-47:~#
  
```

Instance Details:

- Instance ID: i-098be451af973260d (inst1)
- Public IP: 13.235.42.231 Private IP: 172.31.44.47

CloudShell Interface:

documentation for details: <https://docs.docker.com/go/attack-surface/>

```

root@ip-172-31-46-94:/home/ubuntu# systemctl status docker
● docker.service - Docker Application Container Engine
  Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
  Active: active (running) since Mon 2024-04-15 06:53:37 UTC; 24s ago
  TriggeredBy: ● docker.socket
  Docs: https://docs.docker.com
  Main PID: 3391 (dockerd)
  Tasks: 7
  Memory: 37.3M
  CGroup: /system.slice/docker.service
          └─3391 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Apr 15 06:53:36 ip-172-31-46-94 systemd[1]: Starting Docker Application Container Engine...
Apr 15 06:53:36 ip-172-31-46-94 dockerd[3391]: time="2024-04-15T06:53:36.600909474Z" level=info msg="Starting up"
Apr 15 06:53:36 ip-172-31-46-94 dockerd[3391]: time="2024-04-15T06:53:36.602443194Z" level=info msg="detected 127.0.0.53 nameserver, assuming systemd-resolved"
Apr 15 06:53:36 ip-172-31-46-94 dockerd[3391]: time="2024-04-15T06:53:36.813864873Z" level=info msg="Loading containers: start."
Apr 15 06:53:37 ip-172-31-46-94 dockerd[3391]: time="2024-04-15T06:53:37.138487042Z" level=info msg="Loading containers: done."
Apr 15 06:53:37 ip-172-31-46-94 dockerd[3391]: time="2024-04-15T06:53:37.175400849Z" level=info msg="Docker daemon" commit=60b9add containerd-snapshotter=
Apr 15 06:53:37 ip-172-31-46-94 dockerd[3391]: time="2024-04-15T06:53:37.175780662Z" level=info msg="Daemon has completed initialization"
Apr 15 06:53:37 ip-172-31-46-94 dockerd[3391]: time="2024-04-15T06:53:37.218681957Z" level=info msg="API listen on /run/docker.sock"
Apr 15 06:53:37 ip-172-31-46-94 systemd[1]: Started Docker Application Container Engine.
lines 1-20/20 (END)

root@ip-172-31-46-94:/home/ubuntu# docker swarm join --token SWMTKN-1-4uhw6fu0zwmzbzqfkgbw68s25p6tzeak3goubge9tv18x39razs-8rolhpuaa0r9lrmv2gy9av6bj 172.31.44.47:2377
This node joined a swarm as a worker.
root@ip-172-31-46-94:/home/ubuntu#
  
```

Instance Details:

- Instance ID: i-0e84003aaa2afe8a9 (inst 2)
- Public IP: 13.234.232.131 Private IP: 172.31.46.94

The screenshot shows a terminal window within the AWS CloudShell interface. The user is logged in as root on an Ubuntu instance with IP 172.31.33.247. The terminal output shows the following steps:

- Running `systemctl status docker` to check the status of the Docker service. The output indicates that Docker is loaded and active (running).
- Running `docker swarm join --token SWMTKN-1-4uhw6fu0zwmzbzqfkgbw68s25p6tzeak3goubqe9tv18x39razs-8rolhpuua0r9lmr2gy9av6bj 172.31.44.47:2377` to join the node to a Docker Swarm.
- The terminal output shows the Docker daemon starting up, including messages like "Starting Docker Application Container Engine...", "detected l27.0.0.53 nameserver, assuming systemd...", "Loading containers: start.", "Loading containers: done.", "Docker daemon commit=60b9add containerd-snapshot...", "Daemon has completed initialization", and "API listen on /run/docker.sock".
- The terminal output also shows the Docker daemon's PID (3026) and its group (/system.slice/docker.service).
- The terminal output shows the Docker daemon's status as "Stopped" and the Docker service as "Active: active (running) since Mon 2024-04-15 06:53:50 UTC; 28s ago".
- The terminal output shows the Docker daemon's status as "Stopped" and the Docker service as "Active: active (running) since Mon 2024-04-15 06:53:50 UTC; 28s ago".

The terminal output is as follows:

```
documentation for details: https://docs.docker.com/go/attack-surface/
=====
root@ip-172-31-33-247:/home/ubuntu# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2024-04-15 06:53:50 UTC; 28s ago
 TriggeredBy: ● docker.socket
   Docs: https://docs.docker.com
  Main PID: 3026 (dockerd)
    Tasks: 7
   Memory: 37.3M
   CGroup: /system.slice/docker.service
           └─3026 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

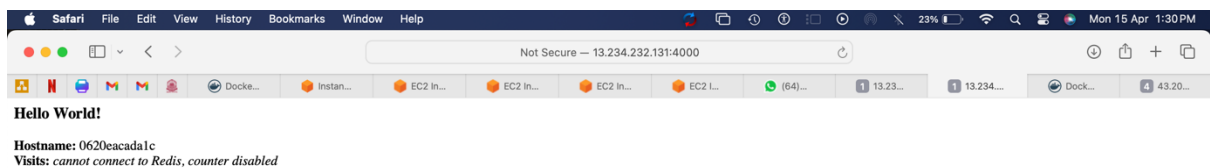
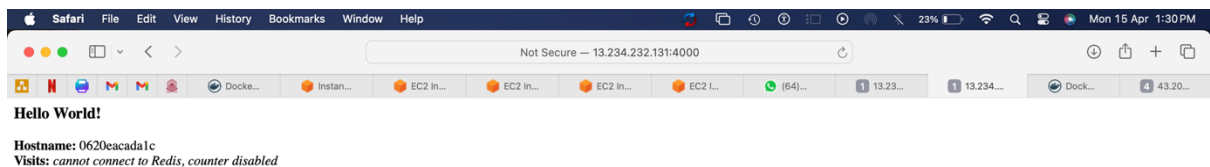
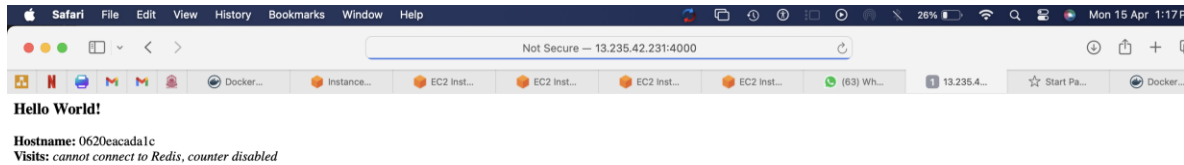
Apr 15 06:53:50 ip-172-31-33-247 systemd[1]: Starting Docker Application Container Engine...
Apr 15 06:53:50 ip-172-31-33-247 dockerd[3026]: time="2024-04-15T06:53:50.396065454Z" level=info msg="Starting up"
Apr 15 06:53:50 ip-172-31-33-247 dockerd[3026]: time="2024-04-15T06:53:50.397537755Z" level=info msg="detected l27.0.0.53 nameserver, assuming systemd..."
Apr 15 06:53:50 ip-172-31-33-247 dockerd[3026]: time="2024-04-15T06:53:50.576850702Z" level=info msg="Loading containers: start."
Apr 15 06:53:50 ip-172-31-33-247 dockerd[3026]: time="2024-04-15T06:53:50.848440601Z" level=info msg="Loading containers: done."
Apr 15 06:53:50 ip-172-31-33-247 dockerd[3026]: time="2024-04-15T06:53:50.880252478Z" level=info msg="Docker daemon commit=60b9add containerd-snapshot..."
Apr 15 06:53:50 ip-172-31-33-247 dockerd[3026]: time="2024-04-15T06:53:50.880637616Z" level=info msg="Daemon has completed initialization"
Apr 15 06:53:50 ip-172-31-33-247 dockerd[3026]: time="2024-04-15T06:53:50.924348561Z" level=info msg="API listen on /run/docker.sock"
Apr 15 06:53:50 ip-172-31-33-247 systemd[1]: Started Docker Application Container Engine.
lines 1-20/20 (END)
[!]+ Stopped                                systemctl status docker

root@ip-172-31-33-247:/home/ubuntu# docker swarm join --token SWMTKN-1-4uhw6fu0zwmzbzqfkgbw68s25p6tzeak3goubqe9tv18x39razs-8rolhpuua0r9lmr2gy9av6bj 172.31.44.47:2377
This node joined a swarm as a worker.
root@ip-172-31-33-247:/home/ubuntu#
```

The terminal output is as follows:

```
i-Oe026f5b74454147a (inst 3)
PublicIPs: 43.205.198.186 PrivateIPs: 172.31.33.247
```

Step 7: Now connect to the IP address with the port 4000 of all the nodes.



End Of Practical

Sign:

Subject In charge:

Dr. Swapnil D. Waghmare