

DOCKER NETWORKS & DOCKER COMPOSE

DOCKER NETWORKS

Types of Docker Networks

There are four types of docker networks

1. Bridge Network
2. Host Network
3. None Network
4. Overlay Network

Bridge Network:

In Bridge Network containers are communicate with each other in

the same host and the default network is bridge network

Host Network:

In Host Network the host IP and container IP is same

None Network:

None Network is not exposing the container because it can't provide

the IP for the container

Overlay Network:

Overlay Network is used to create communication between one server container to another server container

Steps to create Bridge Network

1. First create one network in terminal by using command
`#docker network create network name`
2. To list the Networks there is command
`#docker network ls`

As shown in below figure

The screenshot shows an AWS Management Console terminal window for an EC2 instance. The terminal displays the following content:

```
Tasks: 12
Memory: 112.0M
CPU: 375ms
CGROUP: /system.slice/docker.service
└─2698 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nfile=32768:65536

Nov 20 06:09:17 ip-172-31-33-103.ec2.internal systemd[1]: Starting docker.service - Docker Application Container Engine...
Nov 20 06:09:18 ip-172-31-33-103.ec2.internal dockerd[2698]: time="2024-11-20T06:09:18.007289251Z" level=info msg="Starting up"
Nov 20 06:09:18 ip-172-31-33-103.ec2.internal dockerd[2698]: time="2024-11-20T06:09:18.245463296Z" level=info msg="[graphdriver]
using prior storage driver: overlay2"
Nov 20 06:09:18 ip-172-31-33-103.ec2.internal dockerd[2698]: time="2024-11-20T06:09:18.272218957Z" level=info msg="Loading contain
ers: start."
Nov 20 06:09:18 ip-172-31-33-103.ec2.internal dockerd[2698]: time="2024-11-20T06:09:18.915954758Z" level=info msg="Default bridge
(docker0) is assigned with an IP address 172.17.0.0/16. Daemon option --bip
Nov 20 06:09:18 ip-172-31-33-103.ec2.internal dockerd[2698]: time="2024-11-20T06:09:18.995538591Z" level=info msg="Loading contain
ers: done."
Nov 20 06:09:19 ip-172-31-33-103.ec2.internal dockerd[2698]: time="2024-11-20T06:09:19.030332509Z" level=info msg="Docker daemon"
commit=b08a51f containerd-snapshotter=false storage-driver=overlay2 version=
Nov 20 06:09:19 ip-172-31-33-103.ec2.internal dockerd[2698]: time="2024-11-20T06:09:19.031889141Z" level=info msg="Daemon has com
pleted initialization"
Nov 20 06:09:19 ip-172-31-33-103.ec2.internal dockerd[2698]: time="2024-11-20T06:09:19.092742317Z" level=info msg="API listen on
/run/docker.sock"
Nov 20 06:09:19 ip-172-31-33-103.ec2.internal systemd[1]: Started docker.service - Docker Application Container Engine.

[root@ip-172-31-33-103 ~]# docker network create devops
3b0bc37eadb332bf23557f354bc1fe2af33d7aab32724fc7ba9239fccaa58624

[root@ip-172-31-33-103 ~]# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
f711b3192255        bridge              bridge              local
3b0bc37eadb3        devops              bridge              local
0ea19e16145a        host                host                local
3a52789427b8        none                null                local

[root@ip-172-31-33-103 ~]#
```

Below the terminal window, the instance details are shown:

i-01160b200f21cbe28 (Docker)
PublicIPs: 54.86.186.203 PrivateIPs: 172.31.33.103

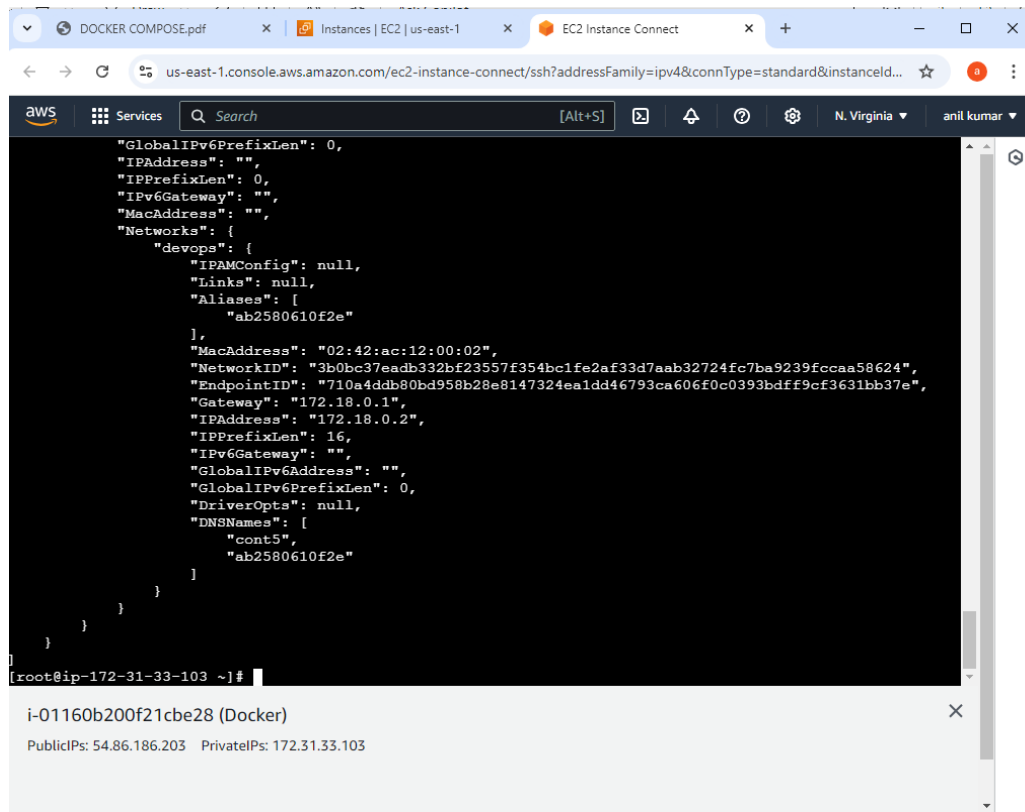
3. Create the network for the container by the command is

#docker run -it --name new cont name --network network name image name

4. After creating the container with created network just inspect that container to see the information by the command

#Docker inspect container name

As shown in below figure



The screenshot shows a terminal window within the AWS Management Console. The terminal displays a JSON configuration for a network interface. The configuration includes details for a network interface (devops) and a network card (cont5). The network interface is associated with a specific network ID and has a private IP address of 172.18.0.1. The network card is associated with a specific endpoint ID and has a private IP address of 172.18.0.2. The terminal prompt is [root@ip-172-31-33-103 ~]#.

```
"GlobalIPv6PrefixLen": 0,
"IPAddress": "",
"IPPrefixLen": 0,
"IPv6Gateway": "",
"MacAddress": "",
"Networks": {
  "devops": {
    "IPAMConfig": null,
    "Links": null,
    "Aliases": [
      "ab2580610f2e"
    ],
    "MacAddress": "02:42:ac:12:00:02",
    "NetworkID": "3b0bc37eadb332bf23557f354bc1fe2af33d7aab32724fc7ba9239fcca58624",
    "EndpointID": "710a4ddb80bd958b28e8147324ea1dd46793ca606f0c0393bdf9cf3631bb37e",
    "Gateway": "172.18.0.1",
    "IPAddress": "172.18.0.2",
    "IPPrefixLen": 16,
    "IPv6Gateway": "",
    "GlobalIPv6Address": "",
    "GlobalIPv6PrefixLen": 0,
    "DriverOpts": null,
    "DNSNames": [
      "cont5",
      "ab2580610f2e"
    ]
  }
}
```

[root@ip-172-31-33-103 ~]#

i-01160b200f21cbe28 (Docker)
PublicIPs: 54.86.186.203 PrivateIPs: 172.31.33.103

5. Switch to the container by using command

#docker attach cont name

#apt update -y

#apt install iputils-ping

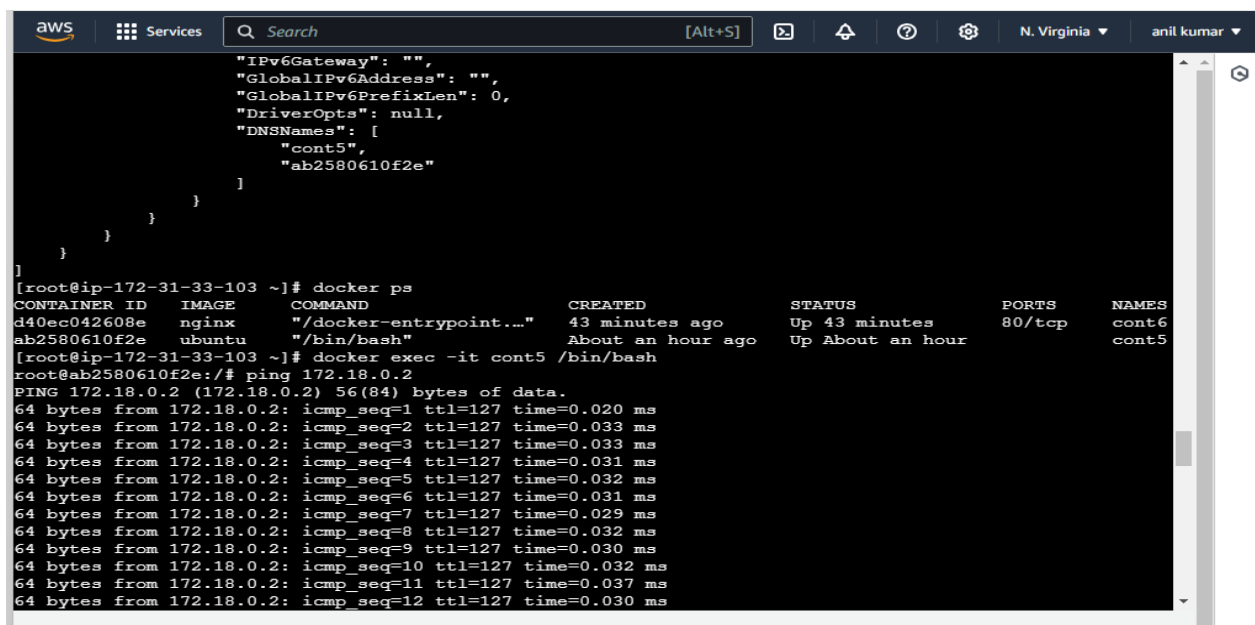
6. Check the running containers

#docker ps

#docker inspect containerName ID

Copy the IP in container1

7. Switch into container2 Type ping paste the copied IP in cont2 – check the connection if the connection is providing then it is working as shown in below figure



```
aws
Services
Search [Alt+S]
N. Virginia
anil kumar

"IPv6Gateway": "",
"GlobalIPv6Address": "",
"GlobalIPv6PrefixLen": 0,
"DriverOpts": null,
"DNSNames": [
  "cont5",
  "ab2580610f2e"
]
}
}
}
}
]

[root@ip-172-31-33-103 ~]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
d40ec042608e   nginx    "/docker-entrypoint..." 43 minutes ago Up 43 minutes 80/tcp       cont6
ab2580610f2e   ubuntu   "/bin/bash"              About an hour ago Up About an hour

[root@ip-172-31-33-103 ~]# docker exec -it cont5 /bin/bash
root@ab2580610f2e:/# ping 172.18.0.2
PING 172.18.0.2 (172.18.0.2) 56(84) bytes of data.
64 bytes from 172.18.0.2: icmp_seq=1 ttl=127 time=0.020 ms
64 bytes from 172.18.0.2: icmp_seq=2 ttl=127 time=0.033 ms
64 bytes from 172.18.0.2: icmp_seq=3 ttl=127 time=0.033 ms
64 bytes from 172.18.0.2: icmp_seq=4 ttl=127 time=0.031 ms
64 bytes from 172.18.0.2: icmp_seq=5 ttl=127 time=0.032 ms
64 bytes from 172.18.0.2: icmp_seq=6 ttl=127 time=0.031 ms
64 bytes from 172.18.0.2: icmp_seq=7 ttl=127 time=0.029 ms
64 bytes from 172.18.0.2: icmp_seq=8 ttl=127 time=0.032 ms
64 bytes from 172.18.0.2: icmp_seq=9 ttl=127 time=0.030 ms
64 bytes from 172.18.0.2: icmp_seq=10 ttl=127 time=0.032 ms
64 bytes from 172.18.0.2: icmp_seq=11 ttl=127 time=0.037 ms
64 bytes from 172.18.0.2: icmp_seq=12 ttl=127 time=0.030 ms
```

DOCKER COMPOSE

What is docker compose

Docker compose is a tool in docker it is used to create multiple services with the single yaml file

Steps to create docker compose

1. First create one server
2. Install the docker packages and start the docker service
3. We have to install the docker compose packages by the command

```
sudo curl -L  
"https://github.com/docker/compose/releases/download/v  
2.17.3/docker-compose-$(uname -s)-$(uname -m)" -o  
/usr/local/bin/docker-compose
```

4. Provide the executable permissions by the command is

```
sudo chmod +x /usr/local/bin/docker-compose
```

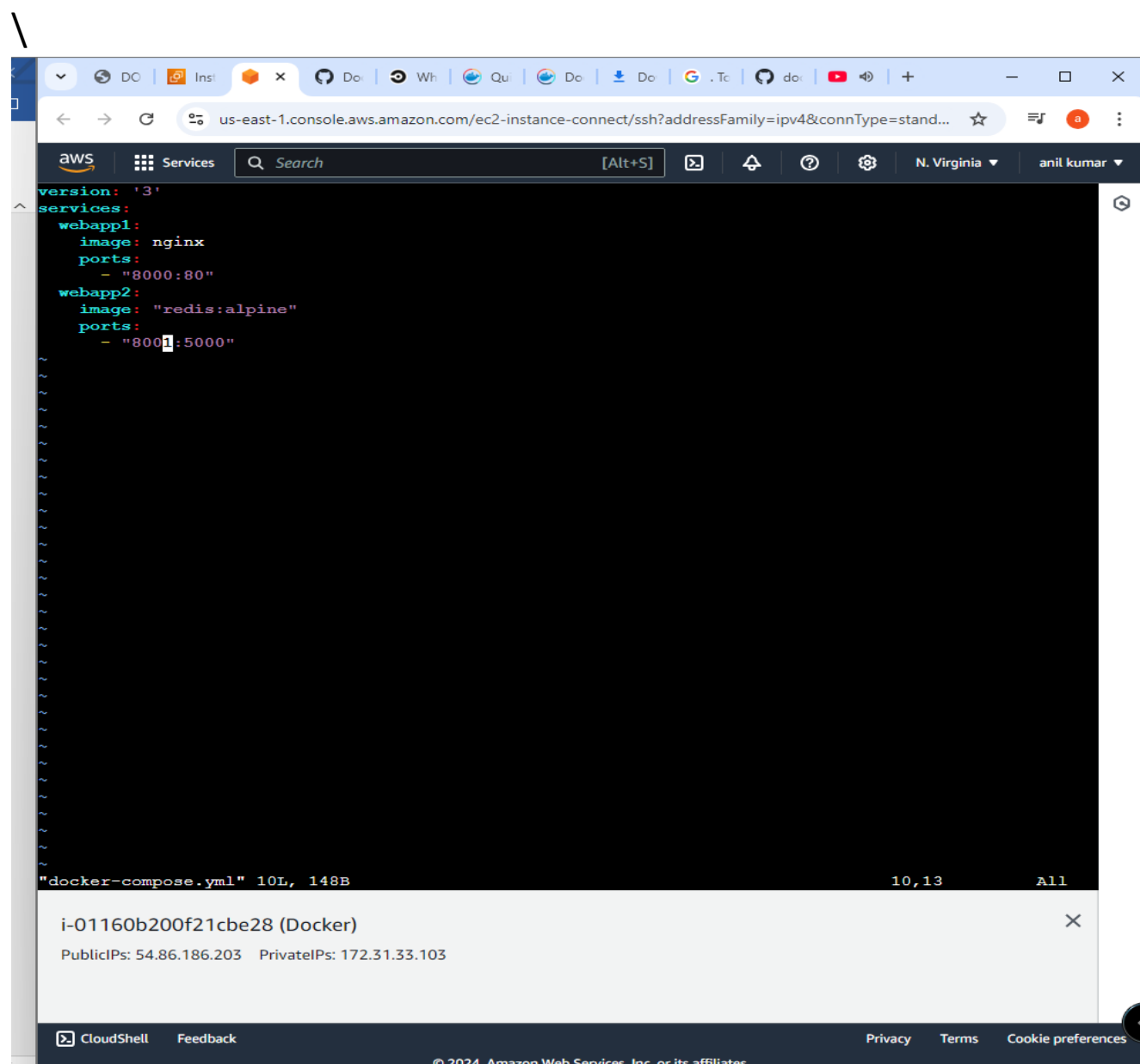
5. To check the version of docker by the command

```
docker-compose --version
```

6. We can add multiple services and we can create multiple containers by using only one file that is yaml file by the command is

vi docker-compose.yml

After opening the editor we need to give some instructions as shown in below figure



The screenshot shows an AWS CloudShell terminal window. The terminal displays the content of a file named `docker-compose.yml`. The YAML content is as follows:

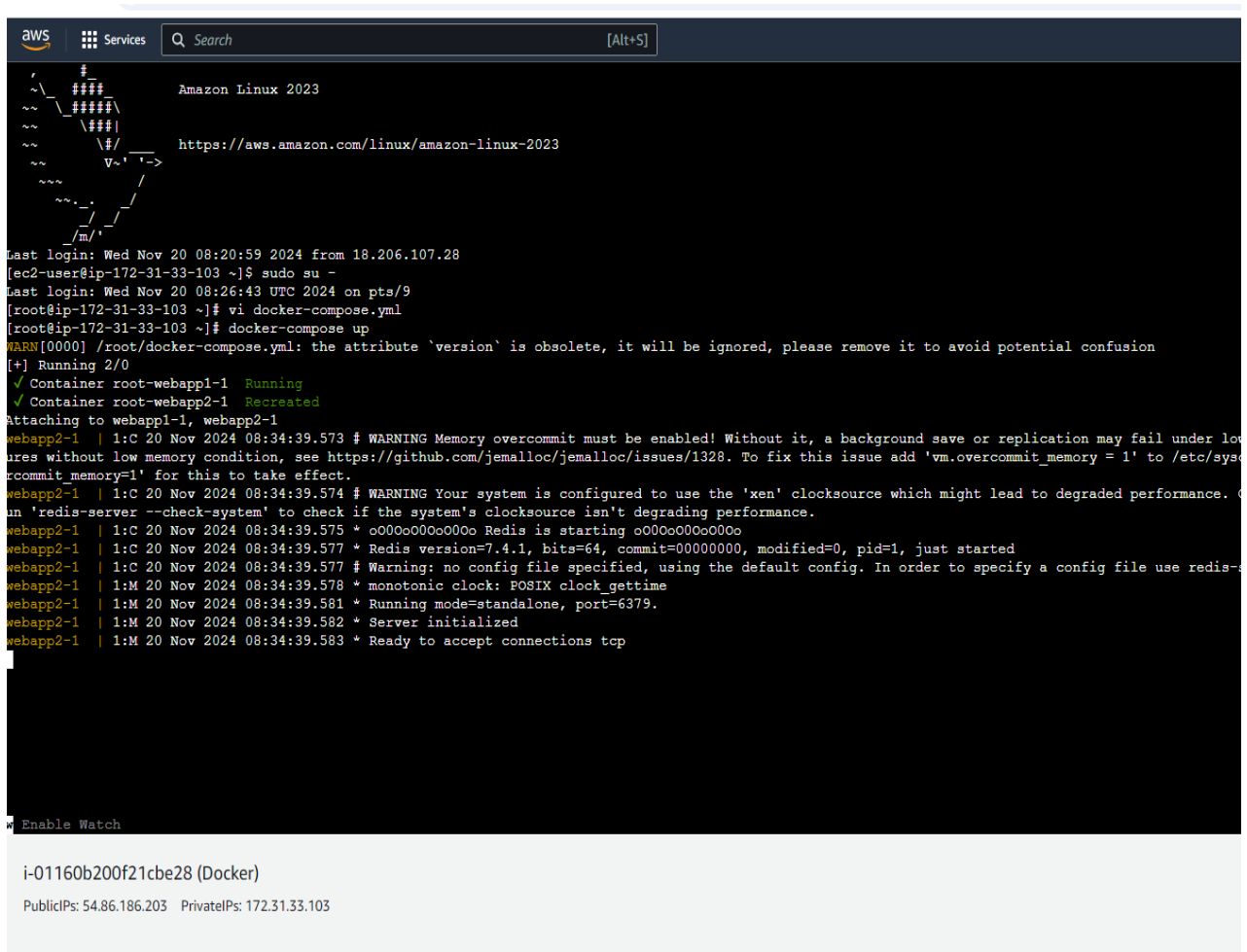
```
version: '3'
services:
  webapp1:
    image: nginx
    ports:
      - "8000:80"
  webapp2:
    image: "redis:alpine"
    ports:
      - "8001:5000"
```

Below the terminal output, there is a summary box for the instance `i-01160b200f21cbe28 (Docker)`. It lists the PublicIPs as `54.86.186.203` and PrivateIPs as `172.31.33.103`. The bottom of the screen shows the CloudShell interface with a feedback link and copyright information: `© 2024 Amazon Web Services, Inc. or its affiliates.`

7. To start the service inside of the compose file the command is

docker-compose up

As shown in below figure



```
aws Services Search [Alt+S]

Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023

Last login: Wed Nov 20 08:20:59 2024 from 18.206.107.28
[ec2-user@ip-172-31-33-103 ~]$ sudo su -
Last login: Wed Nov 20 08:26:43 UTC 2024 on pts/9
[root@ip-172-31-33-103 ~]# vi docker-compose.yml
[root@ip-172-31-33-103 ~]# docker-compose up
WARN[0000] /root/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Running 2/0
  ✓ Container root-webapp1-1 Running
  ✓ Container root-webapp2-1 Recreated
Attaching to webapp1-1, webapp2-1
webapp2-1 | 1:C 20 Nov 2024 08:34:39.573 # WARNING Memory overcommit must be enabled! Without it, a background save or replication may fail under low
webapp2-1 | 1:C 20 Nov 2024 08:34:39.574 # WARNING Your system is configured to use the 'xen' clocksource which might lead to degraded performance.
webapp2-1 | 1:C 20 Nov 2024 08:34:39.575 * oOoOoOoOoOoOo Redis is starting oOoOoOoOoOoOo
webapp2-1 | 1:C 20 Nov 2024 08:34:39.577 * Redis version=7.4.1, bits=64, commit=00000000, modified=0, pid=1, just started
webapp2-1 | 1:C 20 Nov 2024 08:34:39.577 # Warning: no config file specified, using the default config. In order to specify a config file use redis-
webapp2-1 | 1:M 20 Nov 2024 08:34:39.578 * monotonic clock: POSIX clock_gettime
webapp2-1 | 1:M 20 Nov 2024 08:34:39.581 * Running mode=standalone, port=6379.
webapp2-1 | 1:M 20 Nov 2024 08:34:39.582 * Server initialized
webapp2-1 | 1:M 20 Nov 2024 08:34:39.583 * Ready to accept connections tcp

i-01160b200f21cbe28 (Docker)
PublicIPs: 54.86.186.203 PrivateIPs: 172.31.33.103
```

8. Check the service is working or not with the port number in web page as shown in below figure

9. Run the container in detach mode by the command is
`docker-compose up -d`

10.To stop the container we use the command is
`docker- compose down`