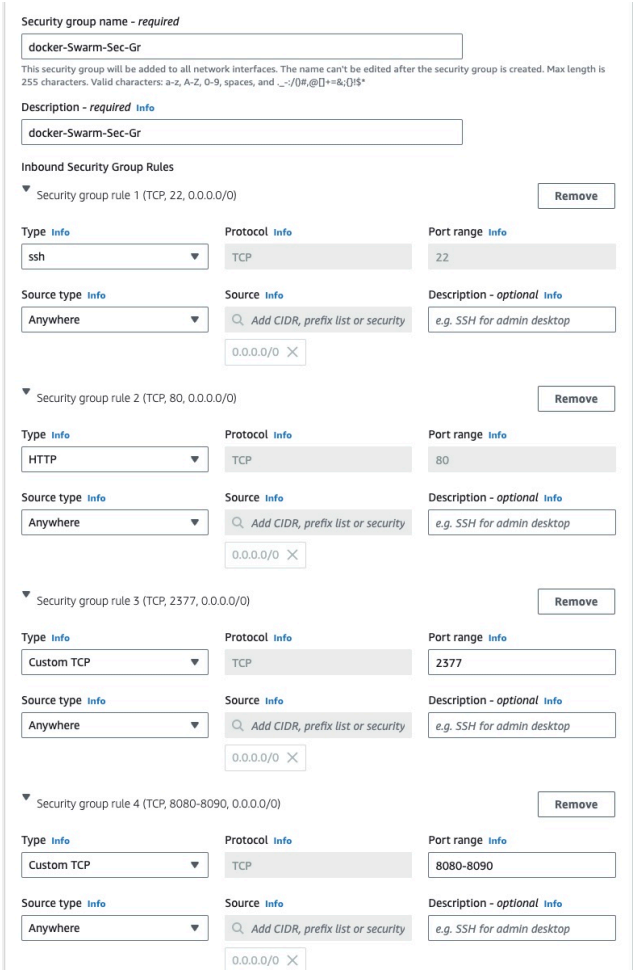
Docker Swarm

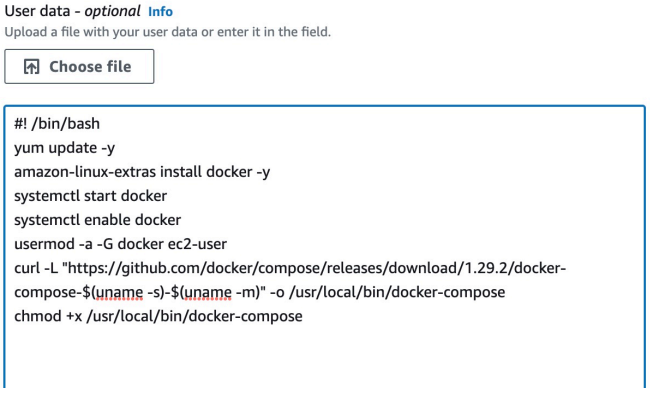
Part 1

Launch Docker Machine Instances and Connect with SSH

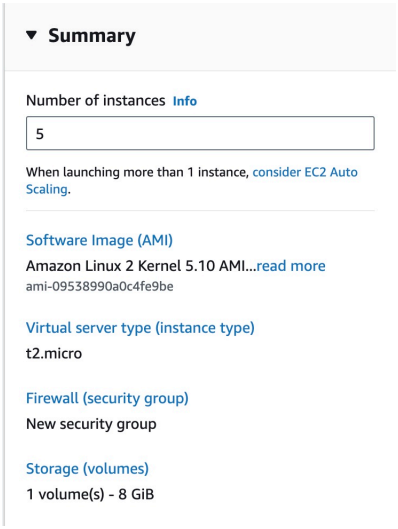
I launch 5 different instances from AWS. The instances AMI is Amazon Linux 2 AMI, t2 micro chip, I created a security group named ‘docker-Swarm-Sec-Gr’, default VPC, ssh port 22 is anywhere, http port 80 is anywhere, custom TCP port 2377 is anywhere, another custom TCP port range is 8080-8090 is anywhere.



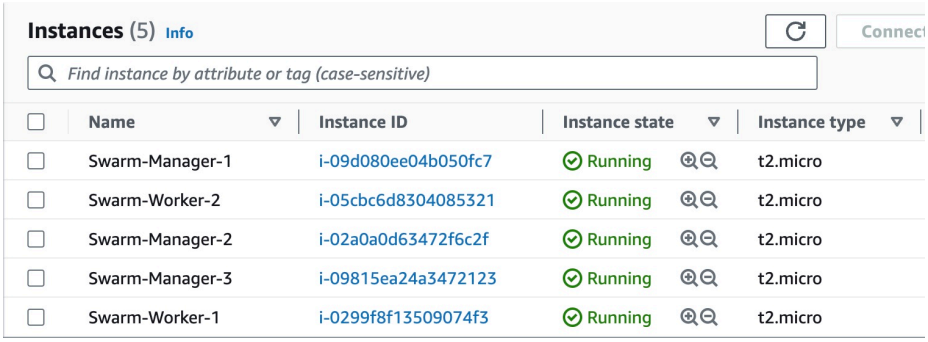
Under advanced details, I paste the code below under the user data.



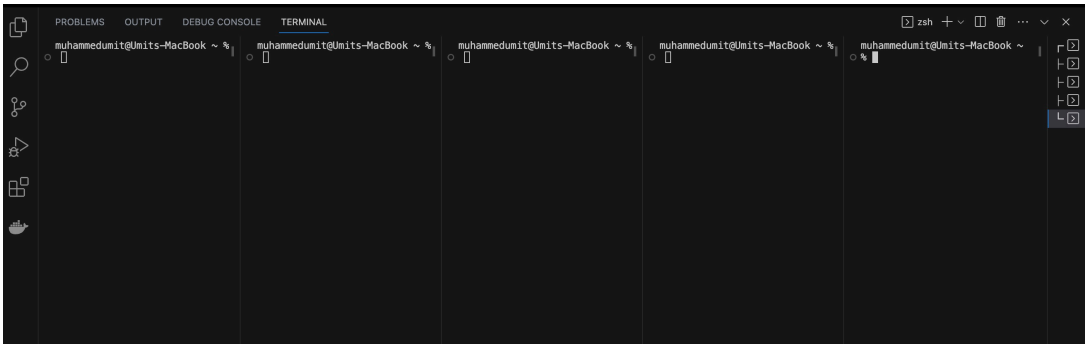
The number of instances selected 5.



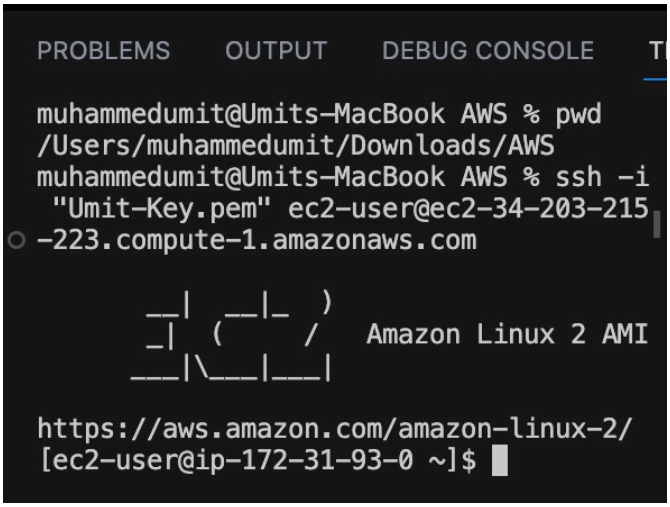
I changed the instances names as ‘Swarm-Manager-1’, ‘Swarm-Manager-2’, Swarm-Manager-3’, ‘Swarm-Worker-1’, ‘Swarm-Worker-2’.



I split the VS-Code terminal window into 5.



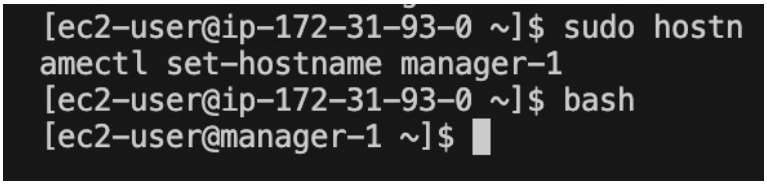
I connect to instances using ssh client.



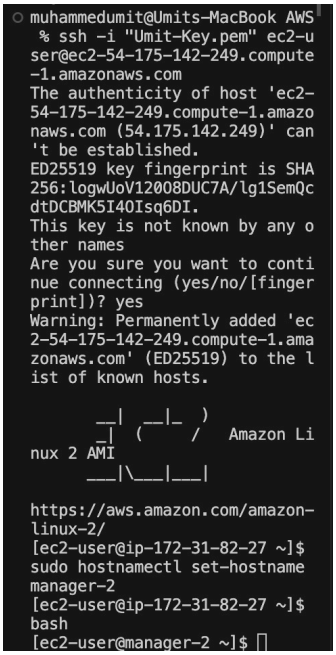
Part 2

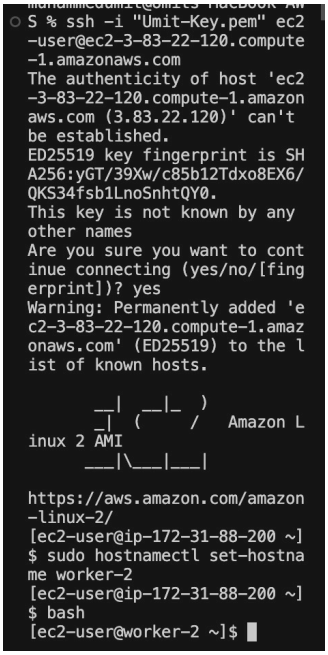
Set Up a Swarm Cluster with Manager and Worker Nodes

I change the hostname of the nodes, so I can discern the roles of each nodes. For example, I can rename the nodes(instances) like master, worker-1 etc. by using ‘sudo hostnamectl set-hostname <node name manager or worker> ’ and ‘bash’ command.

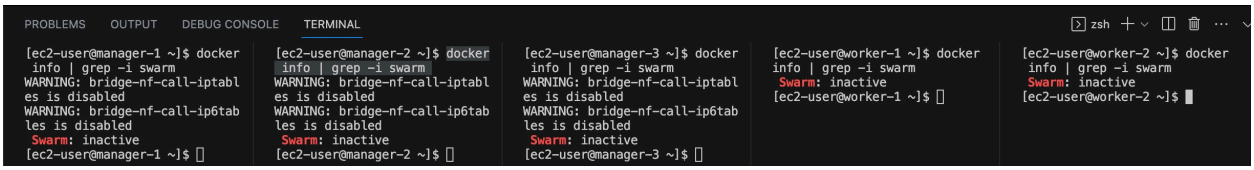


I did the same steps on the other terminal windows.

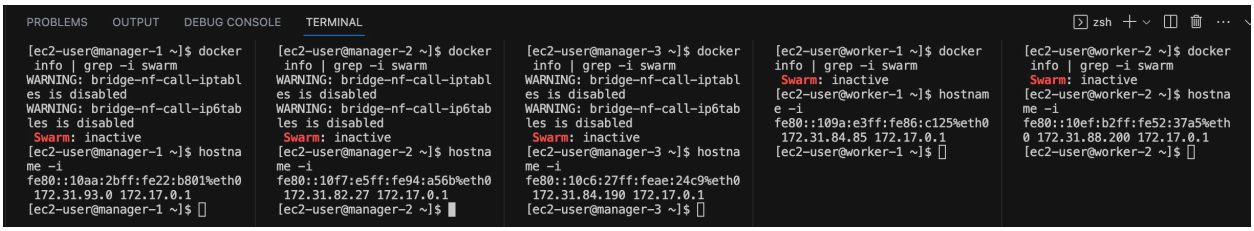




I check if the docker is active or not from the list of docker info by using ‘docker info | grep -i swarm’ code.

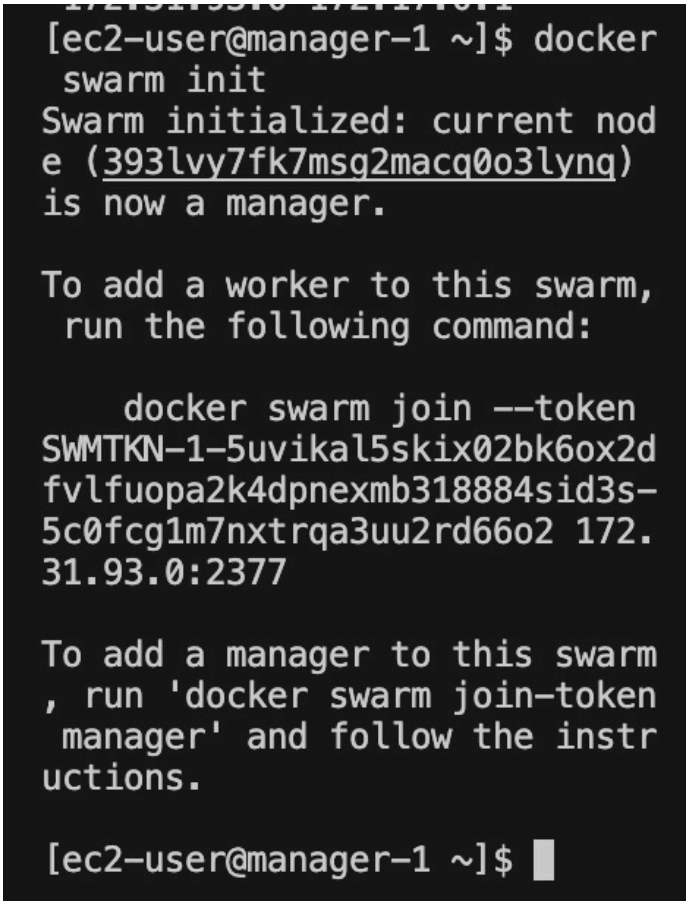


Now, I get the internal IP addresses of docker machines, I find out from the command line by running the ‘hostname -i’ command.



I initialize docker swarm with private IP and assign my first docker machine as manager.

To initialize the code is ‘docker swarm init’



In order to see node(instance) ‘docker node ls’, This node is leader/manager.

```
[ec2-user@manager-1 ~]$ docker
 node ls
ID
HOSTNAME    STATUS    AVAILABI
LITY   MANAGER STATUS   ENGINE
 VERSION
393lvy7fk7msg2macq0o3lynq *
manager-1   Ready     Active
       Leader           20.10.
23
[ec2-user@manager-1 ~]$
```

The ‘docker swarm join-token manager’ command is used to generate a token that allows a new node to join an existing Docker Swarm cluster as a manager node. This is useful when you want to add more manager nodes to Swarm Cluster.

```
[ec2-user@manager-1 ~]$ docker
 swarm join-token manager
To add a manager to this swarm
, run the following command:

    docker swarm join --token
SWMTKN-1-5uvikal5skix02bk6ox2d
fvlfuopa2k4dpnexmb318884sid3s-
ek1eo2ndjkeppsddmint4z7e5 172.
31.93.0:2377

[ec2-user@manager-1 ~]$
```

I copy this code and paste to manager-2 and manager-3 nodes to join Docker Swarm cluster as a manager node.

```
[ec2-user@manager-2 ~]$ docker
 swarm join --token SWMTKN-1-5
uvikal5skix02bk6ox2dfvlfuopa2k
4dpnexmb318884sid3s-ek1eo2ndjk
eppsddmint4z7e5 172.31.93.0:23
77
This node joined a swarm as a
manager.
[ec2-user@manager-2 ~]$
```

```
[ec2-user@manager-3 ~]$ docker
 swarm join --token SWMTKN-1-5
uvikal5skix02bk6ox2dfvlfuopa2k
4dpnexmb318884sid3s-ek1eo2ndjk
eppsddmint4z7e5 172.31.93.0:23
77
This node joined a swarm as a
manager.
[ec2-user@manager-3 ~]$
```

The ‘docker swarm join-token worker’ command is used to generate a token that allows a new node to join an existing Docker Swarm cluster as a worker node.

```
[ec2-user@manager-1 ~]$ docker
 swarm join-token worker
To add a worker to this swarm,
 run the following command:

    docker swarm join --token
SWMTKN-1-5uvikal5skix02bk6ox2d
fvlfuopa2k4dpnexmb318884sid3s-
5c0fcg1m7nxtrqa3uu2rd66o2 172.
31.93.0:2377
```

I copy this code and paste to worker-1 and worker-2 nodes to join Docker Swarm cluster as a worker node.

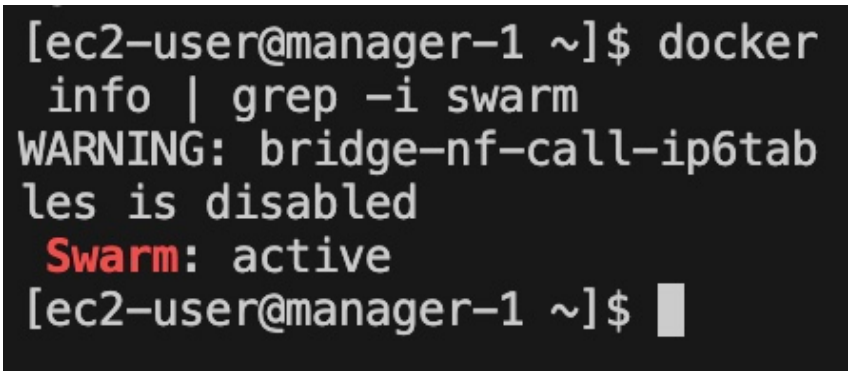
```
[ec2-user@worker-1 ~]$ docker
node ls
Error response from daemon: Th
is node is not a swarm manager
. Worker nodes can't be used t
o view or modify cluster state
. Please run this command on a
 manager node or promote the c
urrent node to a manager.
[ec2-user@worker-1 ~]$
```

```
[ec2-user@worker-2 ~]$  docke
r swarm join --token SWMTKN-1
-5uvikal5skix02bk6ox2dfvlfuop
a2k4dpnexmb318884sid3s-5c0fcg
1m7nxtrqa3uu2rd66o2 172.31.93
.0:2377
This node joined a swarm as a
 worker.
[ec2-user@worker-2 ~]$
```

To see nodes, ‘docker node ls’ on one of the manager node. I tried to give ‘docker node ls’ command on worker node, it gives error.

```
[ec2-user@manager-1 ~]$ docker
 node ls
ID
HOSTNAME    STATUS    AVAILABI
LITY   MANAGER STATUS   ENGINE
 VERSION
393lvy7fk7msg2macq0o3lynq *
manager-1   Ready     Active
      Leader           20.10.
23
7ou51j18ra4bw2lql7fwbv6yx
manager-2   Ready     Active
      Reachable        20.10.
23
fgqdv4nx7x7y0h5dsbds8hvxz
manager-3   Ready     Active
      Reachable        20.10.
23
qjrjq4pb51md4n7nahpl4bvma
worker-1    Ready     Active
                       20.10.
23
7002gls0mxb1b13rojuv8wnat
worker-2    Ready     Active
                       20.10.
23
[ec2-user@manager-1 ~]$
```

To check if the docker swarm is active or not ‘ docker info | grep -i swarm’

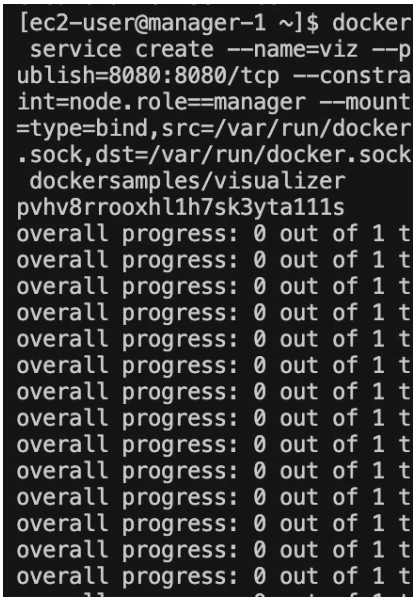


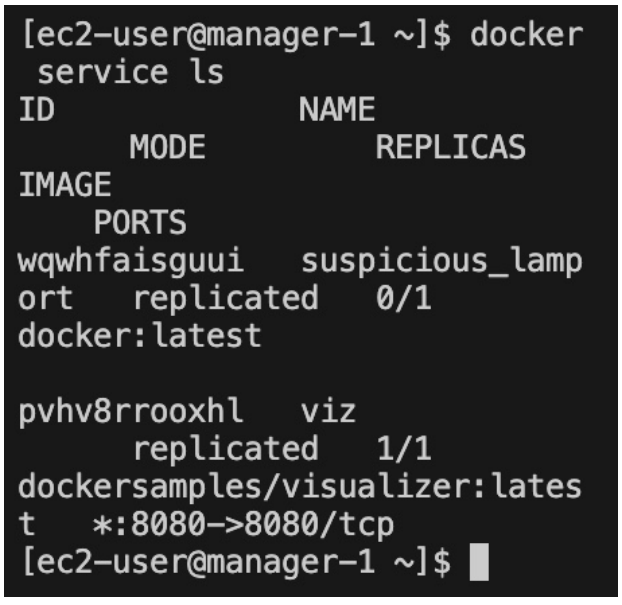
Part 3

Managing Docker Swarm Services

I create ‘Docker Swarm Visualizer’ for visualizing the state and structure of Docker Swarm cluster and its services.
The code is ‘docker service create docker service create --name=viz --publish=8080:8080/tcp --constraint=node.role==manager --mount=type=bind,src=/var/run/docker.sock,dst=/var/run/docker.sock dockersamples/visualizer’.

To see the service ‘docker service ls’

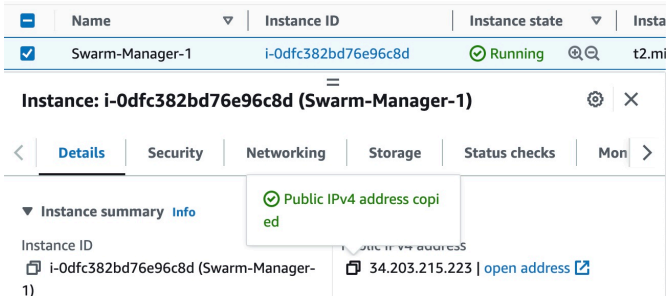


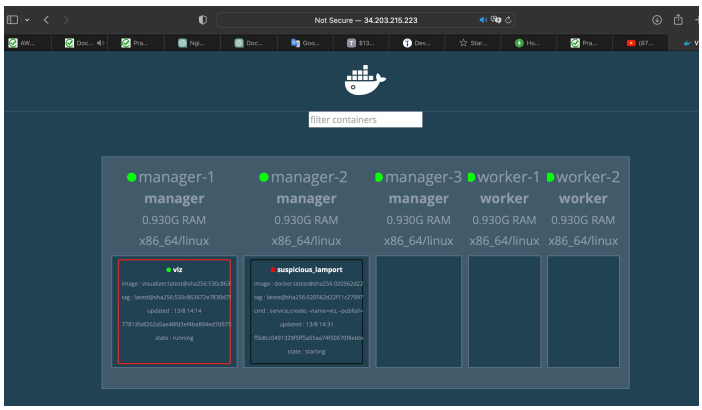


To list the tasks, ‘docker service ps viz’

```
[ec2-user@manager-1 ~]$ docker service ps viz
ID             NAME      IMAGE                             NODE        DESIRED STATE   CURRENT STATE           ERROR     POR
TS
q2heodu4mq38   viz.1     dockersamples/visualizer:latest   manager-1   Running         Running 8 minutes ago
```

I check if the visualizer is running by entering ‘http://<ec2-host-name of this node>:8080’ in a browser. I get the ‘Swarm-Manager-1’ IP from AWS.





Now, I start an nginx service with 5 replicas and see the replicas running on visualizer by using ‘docker service create --name webserver --replicas=5 -p 80:80 -d nginx’ it will work on detached mode, port is 80.

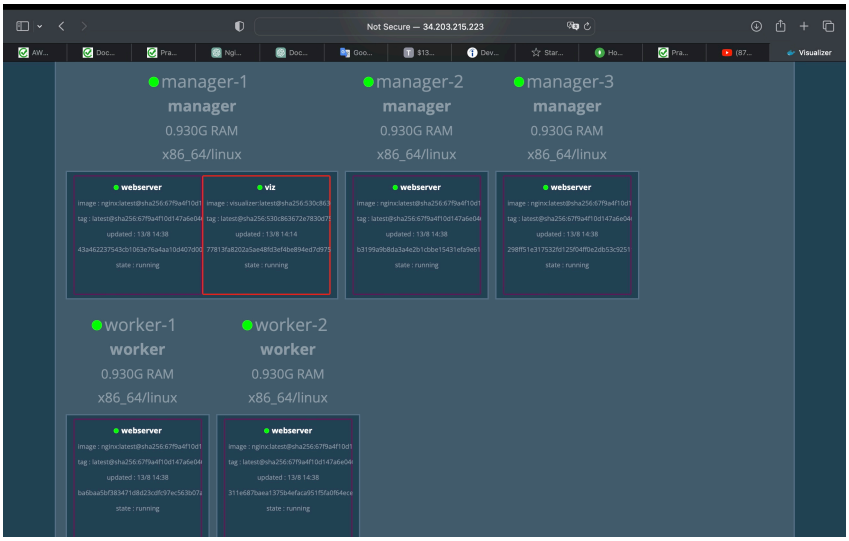
```
[ec2-user@manager-1 ~]$ docker service create --name webserver --replicas=5 -p 80:80 -d nginx
odyiyinuw2bj27rjpex619flz
```

```
[ec2-user@manager-1 ~]$ docker service ls
ID             NAME        MODE         REPLICAS   IMAGE                             PORTS
pvhv8rrooxhl   viz         replicated   1/1        dockersamples/visualizer:latest   *:8080->8080/tcp
odyiyinuw2bj   webserver   replicated   5/5        nginx:latest                      *:80->80/tcp
```

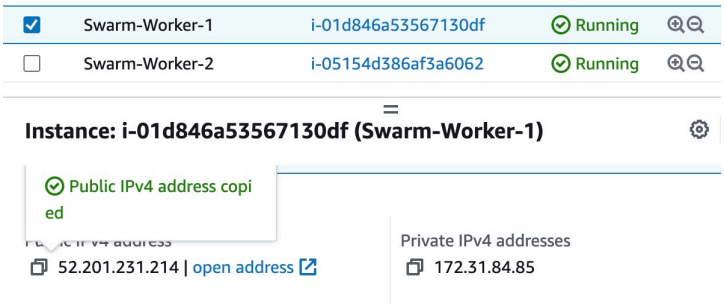
To list the tasks ‘docker service ps webserver’

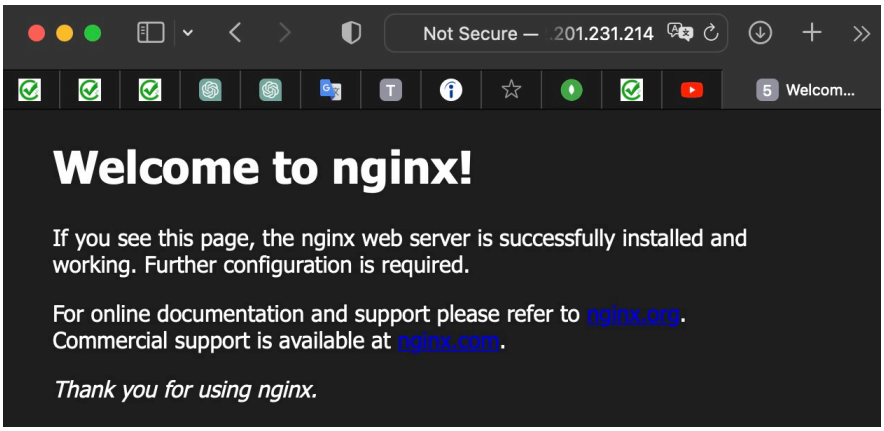
```
[ec2-user@manager-1 ~]$ docker service ps webserver
ID             NAME          IMAGE          NODE        DESIRED STATE   CURRENT STATE           ERROR     PORTS
ou3ztsil4s8k   webserver.1   nginx:latest   manager-1   Running         Running 4 minutes ago
qjt11u9d7i5t   webserver.2   nginx:latest   worker-2    Running         Running 4 minutes ago
s8d68vk2s32a   webserver.3   nginx:latest   manager-3   Running         Running 4 minutes ago
qjq64tjsaq7c   webserver.4   nginx:latest   worker-1    Running         Running 4 minutes ago
y70ifnzklw0r   webserver.5   nginx:latest   manager-2   Running         Running 4 minutes ago
```

I check the visualizer and all of nodes have webserver.



I check if the Nginx is running by entering ‘http://<ec2-host-name of any node>’ in a browser. I get the ‘Swarm-Worker-1’ IP from AWS.





I display detailed information on service by using ‘docker service inspect webserver’ and ‘docker service inspect —pretty webserver’ command. ‘—pretty’ code gives yaml format, without pretty its json format.

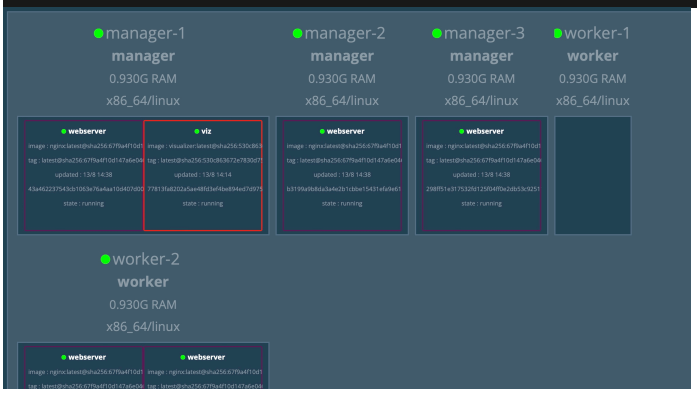
```
[ec2-user@manager-1 ~]$ docker service inspect --pretty webserver

ID:             odyiyinuw2bj27rjpex619flz
Name:           webserver
Service Mode:   Replicated
 Replicas:      5
Placement:
UpdateConfig:
 Parallelism:   1
 On failure:    pause
 Monitoring Period: 5s
 Max failure ratio: 0
 Update order:      stop-first
RollbackConfig:
 Parallelism:   1
 On failure:    pause
 Monitoring Period: 5s
 Max failure ratio: 0
 Rollback order:    stop-first
ContainerSpec:
 Image:         nginx:latest@sha256:67f9a4f10d147a6e04629340e6493c9
 Init:          false
Resources:
Endpoint Mode:  vip
Ports:
 PublishedPort = 80
  Protocol = tcp
  TargetPort = 80
  PublishMode = ingress
```

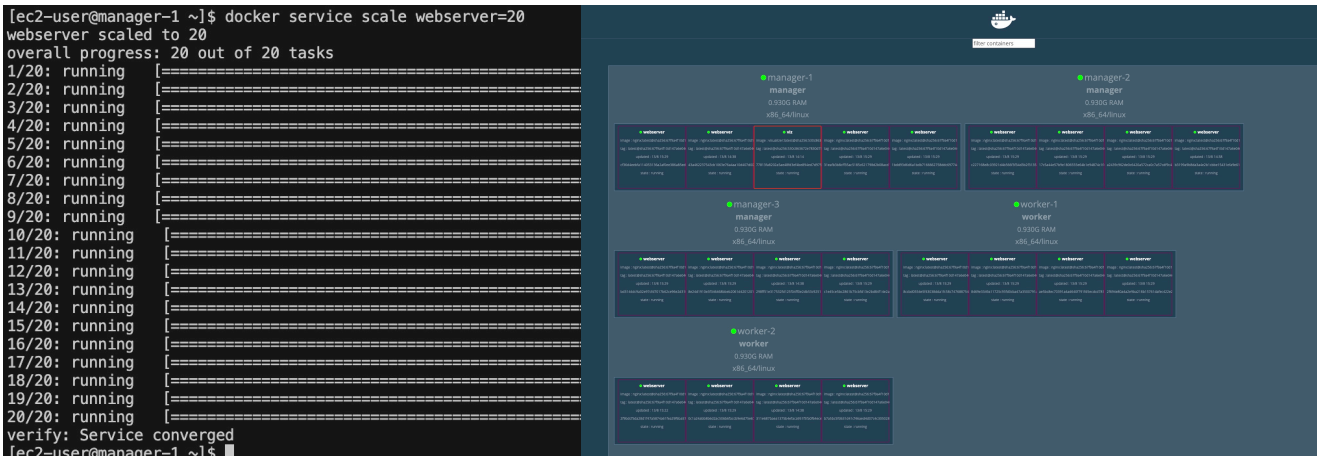

```
[ec2-user@manager-1 ~]$ docker service inspect webserver
[
    {
        "ID": "odyiyinuw2bj27rjpex619flz",
        "Version": {
            "Index": 2925
        },
        "CreatedAt": "2023-08-13T18:38:45.899842464Z",
        "UpdatedAt": "2023-08-13T18:38:45.903318268Z",
        "Spec": {
            "Name": "webserver",
            "Labels": {},
            "TaskTemplate": {
                "ContainerSpec": {
                    "Image": "nginx:latest@sha256:67f9a4f10d147a6e04629340e6493c9703300ca23a2f7f3aa56fe615d75d31ca",
                    "Init": false,
                    "StopGracePeriod": 10000000000,
                    "DNSConfig": {},
                    "Isolation": "default"
                },
                "Resources": {
                    "Limits": {},
                    "Reservations": {}
                },
                "RestartPolicy": {
                    "Condition": "any",
                    "Delay": 5000000000,
                    "MaxAttempts": 0
                },
                "Placement": {
                    "Platforms": [
                        {
                            "Architecture": "amd64",
                            "OS": "linux"
                        },
                        {
                            "OS": "linux"
                        },
                        {
                            "OS": "linux"
                        },
                        {
                            "Architecture": "arm64",
                            "OS": "linux"
```


Now, I reboot a worker node by using ‘ sudo reboot -f’ command. -f is force. Worker-2 get the task after rebooting.

```
[ec2-user@worker-1 ~]$ sudo reboot -f
Rebooting.
```

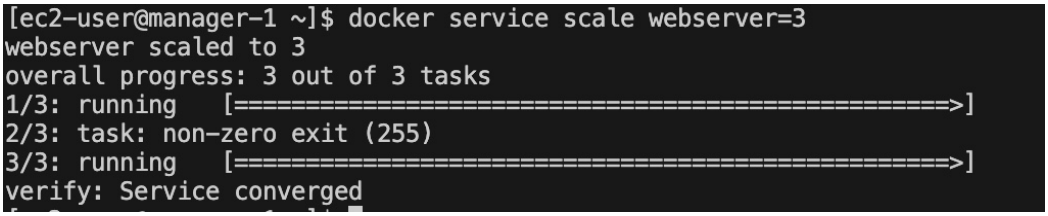


I scale up services by using ‘docker service scale webserver=20’ and check the changes on visualizer app.



Everyone share 20 tasks.

I scale down services by using ‘docker service scale webserver=3’ and check the changes on visualizer app.



Everyone share 3 tasks.