

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.

Security group name - required
docker-Swarm-Sec-Gr

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and _-/0#,@[]+=&,:\$*

Description - required Info
docker-Swarm-Sec-Gr

Inbound Security Group Rules

▼ Security group rule 1 (TCP, 22, 0.0.0.0/0)

Type Info	Protocol Info	Port range Info
ssh	TCP	22

Source type Info Source Info Description - optional Info
Anywhere e.g. SSH for admin desktop
0.0.0.0/0 X

▼ Security group rule 2 (TCP, 80, 0.0.0.0/0)

Type Info	Protocol Info	Port range Info
HTTP	TCP	80

Source type Info Source Info Description - optional Info
Anywhere e.g. SSH for admin desktop
0.0.0.0/0 X

▼ Security group rule 3 (TCP, 2377, 0.0.0.0/0)

Type Info	Protocol Info	Port range Info
Custom TCP	TCP	2377

Source type Info Source Info Description - optional Info
Anywhere e.g. SSH for admin desktop
0.0.0.0/0 X

▼ Security group rule 4 (TCP, 8080-8090, 0.0.0.0/0)

Type Info	Protocol Info	Port range Info
Custom TCP	TCP	8080-8090

Source type Info Source Info Description - optional Info
Anywhere e.g. SSH for admin desktop
0.0.0.0/0 X

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

User data - optional Info
Upload a file with your user data or enter it in the field.

```
#!/bin/bash
yum update -y
amazon-linux-extras install docker -y
systemctl start docker
systemctl enable docker
usermod -a -G docker ec2-user
curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
chmod +x /usr/local/bin/docker-compose
```

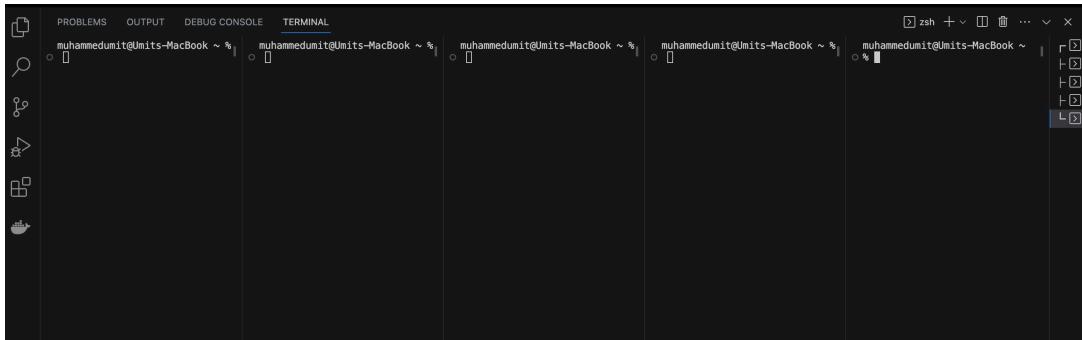
The number of instances selected 5.

▼ Summary	
Number of instances	Info
5	
When launching more than 1 instance, consider EC2 Auto Scaling .	
Software Image (AMI)	
Amazon Linux 2 Kernel 5.10 AMI... read more	
ami-09538990a0c4fe9be	
Virtual server type (instance type)	
t2.micro	
Firewall (security group)	
New security group	
Storage (volumes)	
1 volume(s) - 8 GiB	

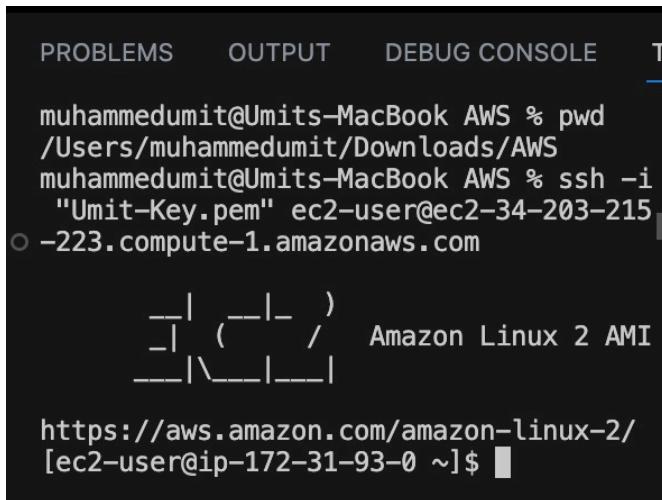
I changed the instances names as ‘Swarm-Manager-1’, ‘Swarm-Manager-2’, Swarm-Manager-3’, ‘Swarm-Worker-1’, ‘Swarm-Worker-2’.

Instances (5) Info		Create instance			
<input type="text"/> Find instance by attribute or tag (case-sensitive)					
	Name	Instance ID	Instance state	Instance type	
<input type="checkbox"/>	Swarm-Manager-1	i-09d080ee04b050fc7	Running		t2.micro
<input type="checkbox"/>	Swarm-Worker-2	i-05cbc6d8304085321	Running		t2.micro
<input type="checkbox"/>	Swarm-Manager-2	i-02a0a0d63472f6c2f	Running		t2.micro
<input type="checkbox"/>	Swarm-Manager-3	i-09815ea24a3472123	Running		t2.micro
<input type="checkbox"/>	Swarm-Worker-1	i-0299f8f13509074f3	Running		t2.micro

I split the VS-Code terminal window into 5.



I connect to instances using ssh client.



```
PROBLEMS OUTPUT DEBUG CONSOLE T

muhammedumit@Umits-MacBook AWS % pwd
/Users/muhammedumit/Downloads/AWS
muhammedumit@Umits-MacBook AWS % ssh -i
"Umit-Key.pem" ec2-user@ec2-34-203-215
○ -223.compute-1.amazonaws.com

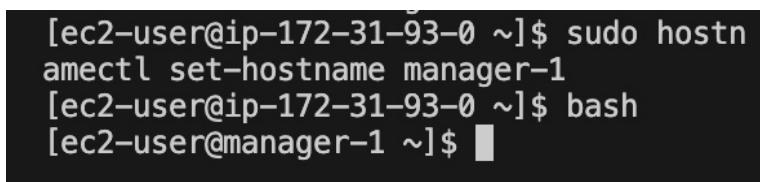
      _|_ ( _/_ /   Amazon Linux 2 AMI
      ___| \___|__|_|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-93-0 ~]$
```

Part 2

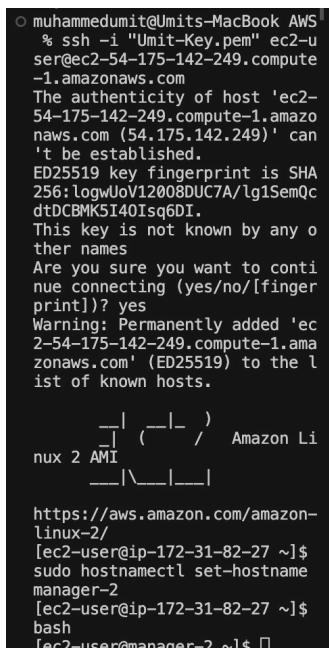
Set Up a Swarm Cluster with Manager and Worker Nodes

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.



```
[ec2-user@ip-172-31-93-0 ~]$ sudo hostn
amectl set-hostname manager-1
[ec2-user@ip-172-31-93-0 ~]$ bash
[ec2-user@manager-1 ~]$
```

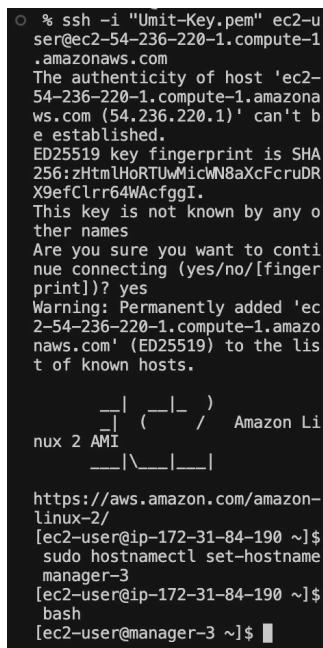
I did the same steps on the other terminal windows.



```
○ muhammedumit@Umits-MacBook AWS %
  % ssh -i "Umit-Key.pem" ec2-u
ser@ec2-54-175-142-249.compute
-1.amazonaws.com
The authenticity of host 'ec2-
54-175-142-249.compute-1.amazo
naws.com (54.175.142.249)' can
't be established.
ED25519 key fingerprint is SHA
256:zHtm1HoRTUwMicNW8xAcFcrDR
X9efCllr64WAcfgGI.
This key is not known by any o
ther names
Are you sure you want to conti
nue connecting (yes/no/[finger
print])? yes
Warning: Permanently added 'ec
2-54-175-142-249.compute-1.ama
zonaws.com' (ED25519) to the l
ist of known hosts.

      _|_ ( _/_ /   Amazon Li
nux 2 AMI
      ___| \___|__|_|

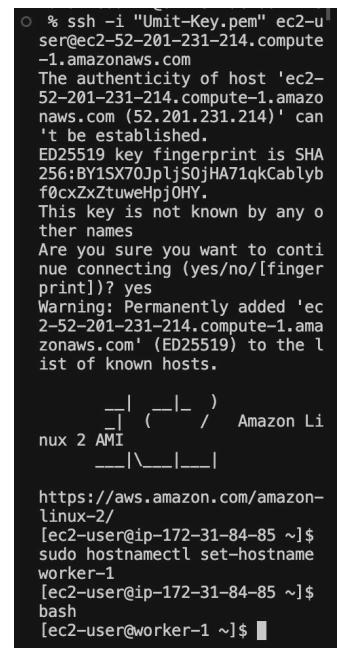
https://aws.amazon.com/amazon-
linux-2/
[ec2-user@ip-172-31-82-27 ~]$ s
udo hostnamectl set-hostname
manager-2
[ec2-user@ip-172-31-82-27 ~]$ b
ash
[ec2-user@manager-2 ~]$
```



```
○ % ssh -i "Umit-Key.pem" ec2-u
ser@ec2-54-220-1.compute-1.
amazonaws.com
The authenticity of host 'ec2-
54-220-1.compute-1.amazona
ws.com (54.236.220.1)' can't b
e established.
ED25519 key fingerprint is SHA
256:BY1SX7OjpljS0jHA71qKcAblyb
f0cxZtuweHpjOHY.
This key is not known by any o
ther names
Are you sure you want to conti
nue connecting (yes/no/[finger
print])? yes
Warning: Permanently added 'ec
2-54-220-1.compute-1.ama
zonaws.com' (ED25519) to the l
ist of known hosts.

      _|_ ( _/_ /   Amazon Li
nux 2 AMI
      ___| \___|__|_|

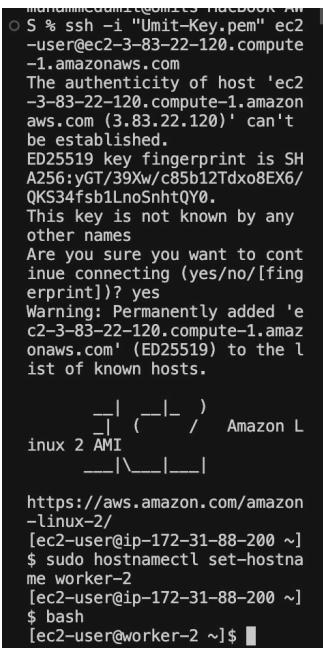
https://aws.amazon.com/amazon-
linux-2/
[ec2-user@ip-172-31-84-190 ~]$ s
udo hostnamectl set-hostname
manager-3
[ec2-user@ip-172-31-84-190 ~]$ b
ash
[ec2-user@manager-3 ~]$
```



```
○ % ssh -i "Umit-Key.pem" ec2-u
ser@ec2-201-231-214.compute
-1.amazonaws.com
The authenticity of host 'ec2-
201-231-214.compute-1.amazo
naws.com (52.201.231.214)' can
't be established.
ED25519 key fingerprint is SHA
256:BY1SX7OjpljS0jHA71qKcAblyb
f0cxZtuweHpjOHY.
This key is not known by any o
ther names
Are you sure you want to conti
nue connecting (yes/no/[finger
print])? yes
Warning: Permanently added 'ec
2-201-231-214.compute-1.ama
zonaws.com' (ED25519) to the l
ist of known hosts.

      _|_ ( _/_ /   Amazon Li
nux 2 AMI
      ___| \___|__|_|

https://aws.amazon.com/amazon-
linux-2/
[ec2-user@ip-172-31-84-85 ~]$ s
udo hostnamectl set-hostname
worker-1
[ec2-user@ip-172-31-84-85 ~]$ b
ash
[ec2-user@worker-1 ~]$
```

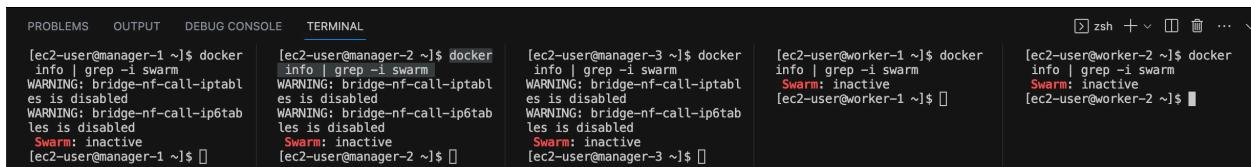


```
○ S % ssh -i "Umit-Key.pem" ec2
-user@ec2-3-83-22-120.compute
-1.amazonaws.com
The authenticity of host 'ec2
-3-83-22-120.compute-1.amazon
aws.com (3.83.22.120)' can't
be established.
ED25519 key fingerprint is SH
A256:yGT/39Xw/c85b12Tdxo8EX6/
QKS34fsb1Lno5hQtY0.
This key is not known by any o
ther names
Are you sure you want to conti
nue connecting (yes/no/[finger
print])? yes
Warning: Permanently added 'ec
2-3-83-22-120.compute-1.ama
zonaws.com' (ED25519) to the l
ist of known hosts.

      _|_ ( _/_ /   Amazon Li
nux 2 AMI
      ___| \___|__|_|

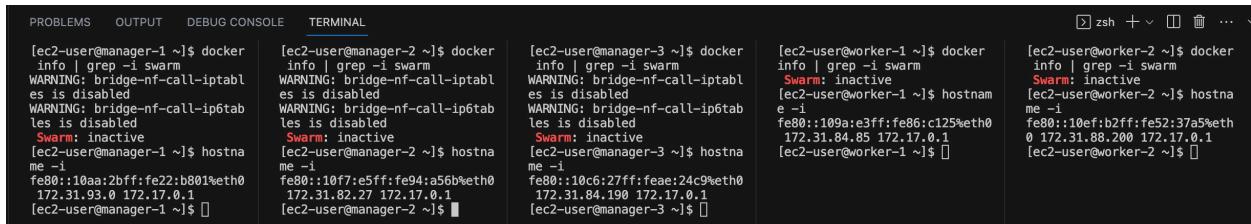
https://aws.amazon.com/amazon-
linux-2/
[ec2-user@ip-172-31-88-200 ~]$ s
udo hostnamectl set-hostna
me worker-2
[ec2-user@ip-172-31-88-200 ~]$ b
ash
[ec2-user@worker-2 ~]$
```

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



```
[ec2-user@manager-1 ~]$ docker info | grep -i swarm
WARNING: bridge-nf-call-iptables is disabled
WARNING: bridge-nf-call-ip6tables is disabled
Swarm: inactive
[ec2-user@manager-1 ~]$ 
[ec2-user@manager-2 ~]$ docker info | grep -i swarm
WARNING: bridge-nf-call-iptables is disabled
WARNING: bridge-nf-call-ip6tables is disabled
Swarm: inactive
[ec2-user@manager-2 ~]$ 
[ec2-user@manager-3 ~]$ docker info | grep -i swarm
WARNING: bridge-nf-call-iptables is disabled
WARNING: bridge-nf-call-ip6tables is disabled
Swarm: inactive
[ec2-user@manager-3 ~]$ 
[ec2-user@worker-1 ~]$ docker info | grep -i swarm
Swarm: inactive
[ec2-user@worker-1 ~]$ 
[ec2-user@worker-2 ~]$ docker info | grep -i swarm
Swarm: inactive
[ec2-user@worker-2 ~]$ 
```

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



```
[ec2-user@manager-1 ~]$ docker info | grep -i swarm
WARNING: bridge-nf-call-iptables is disabled
WARNING: bridge-nf-call-ip6tables is disabled
Swarm: inactive
[ec2-user@manager-1 ~]$ hostname -i
fe80::10aa:2bfff:fe22:b901%eth0
172.31.93.0 172.17.0.1
[ec2-user@manager-1 ~]$ 
[ec2-user@manager-2 ~]$ docker info | grep -i swarm
WARNING: bridge-nf-call-iptables is disabled
WARNING: bridge-nf-call-ip6tables is disabled
Swarm: inactive
[ec2-user@manager-2 ~]$ hostname -i
fe80::10f7:05ff:fe94:a5b8%eth0
172.31.82.27 172.17.0.1
[ec2-user@manager-2 ~]$ 
[ec2-user@manager-3 ~]$ docker info | grep -i swarm
WARNING: bridge-nf-call-iptables is disabled
WARNING: bridge-nf-call-ip6tables is disabled
Swarm: inactive
[ec2-user@manager-3 ~]$ hostname -i
fe80::10c6:27ff:feae:24c9%eth0
172.31.84.190 172.17.0.1
[ec2-user@manager-3 ~]$ 
[ec2-user@worker-1 ~]$ docker info | grep -i swarm
Swarm: inactive
[ec2-user@worker-1 ~]$ hostname -i
fe80::109a:e3ff:fe86:c125%eth0
172.31.84.85 172.17.0.1
[ec2-user@worker-1 ~]$ 
[ec2-user@worker-2 ~]$ docker info | grep -i swarm
Swarm: inactive
[ec2-user@worker-2 ~]$ hostname -i
fe80::10ef:b2fff:fe52:37a5%eth0
172.31.88.200 172.17.0.1
[ec2-user@worker-2 ~]$ 
```

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

To initialize the code is ‘docker swarm init’

```
[ec2-user@manager-1 ~]$ docker swarm init
Swarm initialized: current node (393lvy7fk7msg2macq0o3lyng)
is now a manager.
```

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
```

To add a manager to this swarm
, run ‘docker swarm join--token
manager’ and follow the instr
uctions.

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

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.

```
172.31.82.27 172.17.0.1
[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: The node is not a swarm manager. Worker nodes can't be used to view or modify cluster state. Please run this command on a manager node or promote the current node to a manager.
[ec2-user@worker-1 ~]$
```

```
[ec2-user@worker-2 ~]$ docker
      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
393lvy7fk7msg2macq0o3lyng *
manager-1 Ready Active
Leader 20.10.
23
7ou51j18ra4bw2lql7fwbv6yx
manager-2 Ready Active
Reachable 20.10.
23
fgqdv4nx7x7y0h5dsbds8hvzx
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’

```
[ec2-user@manager-1 ~]$ docker  
info | grep -i swarm  
WARNING: bridge-nf-call-ip6tables is disabled  
Swarm: active  
[ec2-user@manager-1 ~]$
```

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,dest=/var/run/docker.sock dockersamples/visualizer`’.

To see the service ‘docker service ls’

```
[ec2-user@manager-1 ~]$ docker service ls
ID          NAME
           MODE      REPLICAS
IMAGE
           PORTS
wqwhfaisguui    suspicious_lamp
ort      replicated   0/1
docker:latest

pvhv8rrooxhl    viz
                  replicated  1/1
dockersamples/visualizer:lates
t  *:8080->8080/tcp
[ec2-user@manager-1 ~]$
```

To list the tasks, ‘docker service ps viz’

ID	NAME	IMAGE	NODE	DESIRED STATE	CURRENT STATE	ERROR	PORTS
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.

The screenshot displays two windows. On the left is the AWS CloudWatch Metrics console showing the instance details for 'Swarm-Manager-1'. It includes tabs for Details, Security, Networking, Storage, Status checks, and Monitoring. Under the Details tab, it shows the Public IPv4 address as 34.203.215.223. On the right is a browser window showing the Docker Swarm UI. It lists nodes: manager-1 (manager), manager-2 (manager), manager-3 (manager), worker-1 (worker), and worker-2 (worker). Below the nodes, a list of services is shown. One service, 'viz' on 'manager-1', is highlighted with a red border, indicating it is currently running.

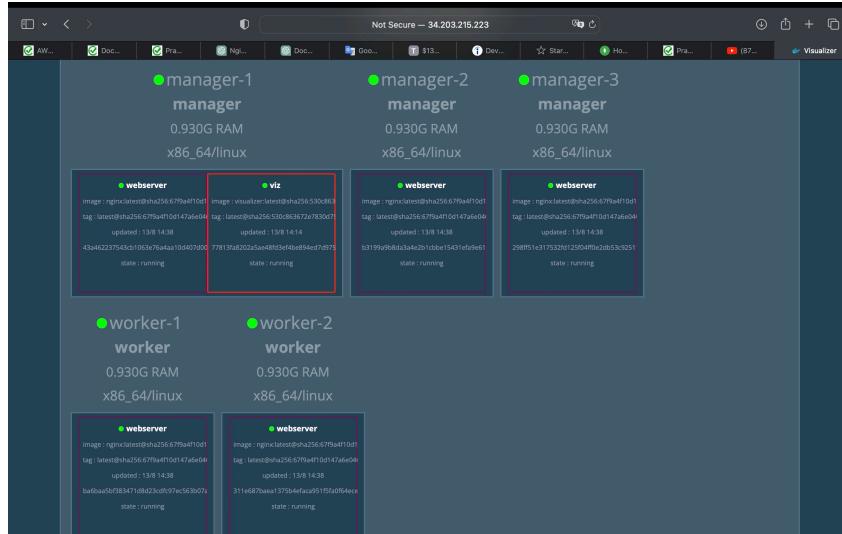
Now, I start an nginx service with 5 replicas and see the replicas running on visualizer by using ‘docker service create --name webserver --replicas=5 -p 80:80 -d nginx’ it will work on detached mode, port is 80.

ID	NAME	MODE	REPLICAS	IMAGE	PORTS
odyiyinuw2bj27rjpx619flz	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’

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		
y70ifnzk1w0r	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.

Swarm-Worker-1 i-01d846a53567130df Running
Swarm-Worker-2 i-05154d386af3a6062 Running

Instance: i-01d846a53567130df (Swarm-Worker-1)

Public IPv4 address copied
52.201.231.214 | open address

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org. Commercial support is available at nginx.com.

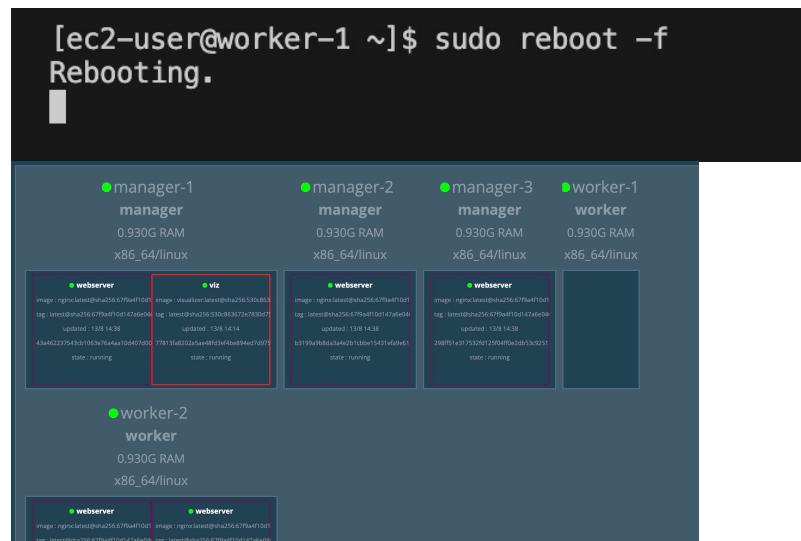
Thank you for using nginx.

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: odyiyinuw2bj27rjpx619flz
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

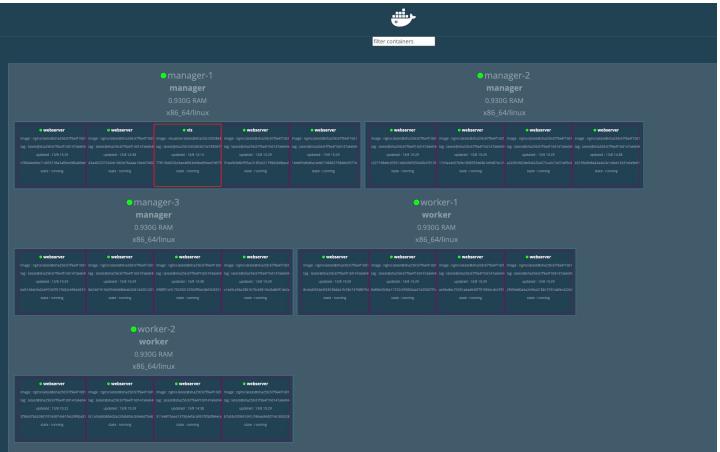
[ec2-user@manager-1 ~]$ docker service inspect webserver
[{"ID": "odyiyinuw2bj27rjpx619flz", "Version": {"Index": 2925}, "CreatedAt": "2023-08-13T18:38:45.899842464Z", "UpdatedAt": "2023-08-13T18:38:45.903318268Z", "Spec": {"Name": "webserver", "Labels": {}, "TaskTemplate": {"ContainerSpec": {"Image": "nginx:latest@sha256:67f9a4f10d147a6e04629340e6493c9703300ca23a2f7f3aa56fe615d75d31ca", "Init": false, "StopGracePeriod": "0000000000", "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.



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

```
[ec2-user@manager-1 ~]$ docker service scale webserver=20
webserver scaled to 20
overall progress: 20 out of 20 tasks
1/20: running [=====]
2/20: running [=====]
3/20: running [=====]
4/20: running [=====]
5/20: running [=====]
6/20: running [=====]
7/20: running [=====]
8/20: running [=====]
9/20: running [=====]
10/20: running [=====]
11/20: running [=====]
12/20: running [=====]
13/20: running [=====]
14/20: running [=====]
15/20: running [=====]
16/20: running [=====]
17/20: running [=====]
18/20: running [=====]
19/20: running [=====]
20/20: running [=====]
verify: Service converged
[ec2-user@manager-1 ~]$
```



Everyone share 20 tasks.

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

```
[ec2-user@manager-1 ~]$ docker service scale webserver=3
webserver scaled to 3
overall progress: 3 out of 3 tasks
1/3: running [=====>]
2/3: task: non-zero exit (255)
3/3: running [=====>]
verify: Service converged
[ec2-user@manager-1 ~]$
```



Everyone share 3 tasks.