

# Module-7: Containerization using Docker Part - II

---

Demo Document - 4

edureka!

**edureka!**

© Brain4ce Education Solutions Pvt. Ltd.

## DEMO-4: Starting Docker in Swarm Mode

**Note:** All commands are executed as root.

This setup requires 3 separate instances: 1 Manager node and 2 Worker nodes.

1. Create 3 ubuntu instance on GCP with Docker engine installed on them

<input type="checkbox"/>	<input checked="" type="checkbox"/>	docker-1	us-central1-a	10.128.0.27 (nic0)	34.72.112.157 <a href="#">↗</a>	SSH	▼	⋮
<input type="checkbox"/>	<input checked="" type="checkbox"/>	docker-2	us-central1-a	10.128.0.29 (nic0)	34.67.95.152 <a href="#">↗</a>	SSH	▼	⋮
<input type="checkbox"/>	<input checked="" type="checkbox"/>	docker-3	us-central1-a	10.128.0.30 (nic0)	34.71.103.105 <a href="#">↗</a>	SSH	▼	⋮

2. Before initializing with the init command set firewall rules for all the instances  
Create ingress and egress rules for TCP 2377, TCP and UDP 7946, and UDP 4789 ports  
Click on view network details and then Firewall

<input type="checkbox"/>	Name ^	Zone	Recommendation	In use by	Internal IP	External IP	Connect	
<input type="checkbox"/>	<input checked="" type="checkbox"/>	docker-1	us-central1-a		10.128.0.27 (nic0)	34.72.112.157 <a href="#">↗</a>	SSH	▼
<input type="checkbox"/>	<input checked="" type="checkbox"/>	docker-2	us-central1-a		10.128.0.29 (nic0)	34.67.95.152 <a href="#">↗</a>	SSH	▼
<input type="checkbox"/>	<input checked="" type="checkbox"/>	docker-3	us-central1-a		10.128.0.30 (nic0)	34.71.103.105 <a href="#">↗</a>	SSH	▼

Start  
Stop  
Reset  
Delete  
View network details  
New machine image  
View logs

3. Click on CREATE FIREWALL RULE

The screenshot shows the Google Cloud Platform interface for the 'KubernetesProject'. The 'Firewall' tab is selected, and the 'CREATE FIREWALL RULE' button is highlighted with a red box. The page includes a sidebar with 'VPC network', 'VPC networks', 'External IP addresses', and 'Firewall'. The main content area explains that firewall rules control incoming or outgoing traffic to an instance and provides a link to 'Learn more'.

4. Add a name for your rule

← Create a firewall rule

Firewall rules control incoming or outgoing traffic to an instance. By default, incoming traffic from outside your network is blocked. [Learn more](#)

Name \*

Lowercase letters, numbers, hyphens allowed

5. Select egress or ingress. Select all instances in network for Targets. In Source IP Ranges add the IPs for your instances

Direction of traffic ?

☒ Ingress

☐ Egress

Action on match ?

☒ Allow

☐ Deny

Targets

All instances in the network

Source filter

IP ranges

Source IP ranges \*

34.72.112.157 34.67.95.152 34.71.103.105

for example, 0.0.0.0/0, 192.168.2.0/24

6. Set the port and protocol and click on tcp

Protocols and ports ?

☐ Allow all

☒ Specified protocols and ports

☒ tcp : 2377

☐ udp : all

☐ Other protocols

protocols, comma separated, e.g. ah, sctp

▼ DISABLE RULE

CREATE CANCEL

7. On your manager node run the following command to initialize a swarm cluster

```
$ docker swarm init --advertise-addr <MANAGER-IP>
```

```
root@docker-1:~# docker swarm init --advertise-addr 34.72.112.157
unknown flag: --advertise-addr
See 'docker swarm init --help'.
root@docker-1:~# docker swarm init --advertise-addr 34.72.112.157
Swarm initialized: current node (tulukz7mmibu3a80b70bn4nfa) is now a manager.

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

    docker swarm join --token SWMTKN-1-4p7i7n56ytjf8gcyi3qz2xreyxjzg3vigfp15falkb50nbmvc7-3y1qfpqoug2d4hmkqnpe4nsoq 34.72.112.157:2377

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

8. Copy the output of the init command (marked red in the image above) and use it on the worker nodes

In case, you have lost the output, the join command can be obtained by running the following command

```
$ docker swarm join-token worker
```

```
root@docker-3:~# docker swarm join --token SWMTKN-1-4p7i7n56
q 34.72.112.157:2377
This node joined a swarm as a worker.
```

9. On the manager node run the node ls command to check all the connected nodes in the cluster

```
$ docker node ls
```

```
root@docker-1:~# docker node ls
```

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER STATUS	ENGINE VERSION
tulukz7mmibu3a80b70bn4nfa *	docker-1	Ready	Active	Leader	19.03.11
m1y4s9m41t18msdk2rgg1zdue	docker-2	Ready	Active		19.03.12
5aof5ybchhvxq7klvu7ukqv2	docker-3	Ready	Active		19.03.12

edureka!

## Deploy a Service in Swarm

**Note:** All commands are executed as root.

1. Simply deploy a service using the service create command. Deploying a tomcat service here

```
$ docker service create --name mysvc --replicas 4 -p 8006:8080 tomcat
```

```
root@docker-1:~# docker service create --name tomservice --replicas 3 -p 8006:8080 tomcat
q35e5yfp487qakm5rhoatwqg1
overall progress: 0 out of 3 tasks
1/3: preparing [=====>]
2/3: preparing [=====>]
3/3: preparing [=====>]
```

```
root@docker-1:~# docker service create --name tomservice --replicas 3 -p 8006:8080 tomcat
q35e5yfp487qakm5rhoatwqg1
overall progress: 3 out of 3 tasks
1/3: running [=====>]
2/3: running [=====>]
3/3: running [=====>]
verify: Service converged
```

2. To list the service deployed use:

```
$ docker service ls
```

```
root@docker-1:~# docker service ls
ID                NAME          MODE          REPLICAS          IMAGE          PORTS
q35e5yfp487q      tomservice    replicated    3/3               tomcat:latest  *:8006->8080/tcp
```

3. Check the tasks deployed by the service using the ps command

```
$ docker service ps <serviceName>
```

```
root@docker-1:~# docker service ps tomservice
ID                NAME          IMAGE          NODE          DESIRED STATE  CURRENT STATE          ERR
OR
9dadls5670jk      tomservice.1   tomcat:latest  docker-3      Running        Running 9 minutes ago
o31kikov9z0m      tomservice.2   tomcat:latest  docker-1      Running        Running 9 minutes ago
710bt0ydg6hh      tomservice.3   tomcat:latest  docker-2      Running        Running 8 minutes ago
```

4. To inspect service:

```
$ docker service inspect <serviceName>
```

5. To remove the service:

```
$ docker service rm <serviceName>
```