

Lab Experiment-8

- **Running an application across multiple containers using Docker Swarm.**

Docker swarm mode provides a means to deploy containers across multiple Docker hosts. It uses overlay networks for discovering services and provides a built-in load balancer for scaling the services.

We will create swarm cluster and deploy containers. Then we will also scale the application.

Prerequisite: Having two different instances of Docker.

The steps that need to be followed are:

1. Initialize Docker swarm on the node that is to be treated as manager.

Command: `docker swarm init`

```
Terminal Host 1 +
Your Interactive Bash Terminal. A safe place to learn and execute commands.

$ docker swarm init
Swarm initialized: current node (qhtawkjyz2zjpm3lx2begy3) is now a manager.

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

    docker swarm join --token SWMTKN-1-09tnc94qip78fkb1azueluhjdwrqx92w286o7q07j
    jt166n9ga-6to5n19zcji2mdxkutybkroxx 172.17.0.79:2377

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

$
```

2. Add a worker node to this swarm using the token generated in the above step.

Command: `docker swarm join --token <token-id>`

```
Terminal Host 2
Your Interactive Bash Terminal. A safe place to learn and execute commands.

$ docker swarm join 172.17.0.79:2377 --token $token
flag needs an argument: --token
see 'docker swarm join --help'.
$ docker swarm join --token SWMTKN-1-09tnc94qip78fkb1azueluhjdwrqx92w286o7q07j
    t166n9ga-6to5n19zcji2mdxkutybkroxx 172.17.0.79:2377
This node joined a swarm as a worker.
$
```

The available nodes in the cluster can be listed from manager node.

Command: `docker node ls`

Terminal Host 1 +

\$ docker node ls

ID	HOSTNAME	STATUS	AVAILABILITY
TY	MANAGER STATUS	ENGINE VERSION	
qhtawkjyz2zjpmda3lx2begy3 *	host01	Ready	Active
Leader	19.03.13		
g52aid4f7zhftxtui5b3df6up	host02	Ready	Active
	19.03.13		

\$

3. Now we will create an overlay network over which the containers across different hosts can communicate. We are creating a network named overlay. Run the following command on the manager node.

Command: `docker network create -d overlay skynet`

```
$ docker network create -d overlay skynet
z05fyf943rtopmoxqjdqr1mdk
$
```

4. Now we will deploy the service in the form of containers on these nodes. Here we are using the image `katacoda/docker-http-server` and we are providing it a name `http`. The service will be attached to the network created in the above step and we are creating two replicas of this service.

Command: `docker service create --name http --network skynet --replicas 2 -p 80:80 katacoda/docker-http-server`

```
$ docker service create --name http --network skynet --replicas 2 -p 80:80 katac
oda/docker-http-server
sn0h69ber65z855egj5hnyhxy
overall progress: 2 out of 2 tasks
1/2: running [=====>]
2/2: running [=====>]
verify: Service converged
$
```

We can view the services running on swarm cluster as follows on the manager node.

Command: `docker service ls`

Terminal Host 1 +			
\$ docker service ls			
ID	NAME	MODE	REPLICAS
IMAGE	PORTS		
sn0h69ber65z	http	replicated	2/2
katacoda/docker-http-server:latest	*:80->80/tcp		

5. Above step created two replicas of the service. One replica will run on the manager node and the other replica of the service will run on the worker node. We can verify this by running the following command on manager and worker node.

Command: `docker ps`

Terminal Host 1 +				
\$ docker ps				
CONTAINER ID	IMAGE	STATUS	PORTS	COMMAND
CREATED				NAMES
da6d5d8cd73b	katacoda/docker-http-server:latest	Up About a minute	80/tcp	"/app"
11sfbxk9c				http.2.vpqii865pte0loo5

Terminal Host 2				
\$ docker ps				
CONTAINER ID	IMAGE	STATUS	PORTS	COMMAND
CREATED				NAMES
c813241b46b0	katacoda/docker-http-server:latest	Up 4 minutes	80/tcp	"/app"
9k43mvxot				http.1.u41zt4jb70t6chce

Now, if we issue a request to the public port, it will be processed by either of the containers.

```
$ curl host01
<h1>This request was processed by host: c813241b46b0</h1>
$ curl host01
<h1>This request was processed by host: da6d5d8cd73b</h1>
```

The list of services and their tasks running across the cluster can be seen as follows in the manager node.

Command: `docker service ps <service-name>`

Terminal Host 1 +				
\$ docker service ps http				
ID	NAME	IMAGE		NOD
E	DESIRED STATE	CURRENT STATE	ERROR	
PORTS				
u41zt4jb70t6t02	http.1 Running	katacoda/docker-http-server:latest Running 5 minutes ago		hos
vpqii865pte0t01	http.2 Running	katacoda/docker-http-server:latest Running 5 minutes ago		hos

6. At present we have two replicas in total, with one running on each of the two nodes. Let us scale our service to have total five containers running across the nodes.

Command: `docker service scale http=5`

```
Terminal Host 1 +
$ docker service scale http=5
http scaled to 5
overall progress: 5 out of 5 tasks
1/5: running [=====>]
2/5: running [=====>]
3/5: running [=====>]
4/5: running [=====>]
5/5: running [=====>]
verify: Service converged
```