Lab 8 | Swarm Mode

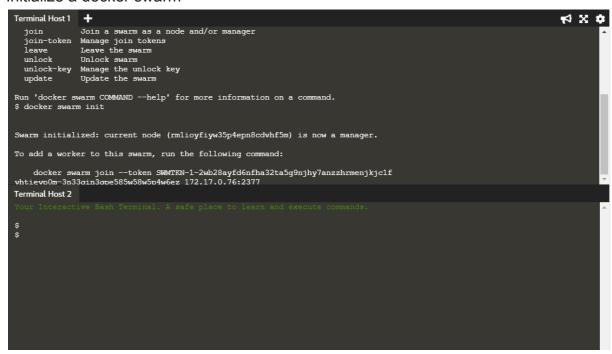
Name: Neha Singh

Roll No: 67

Sap Id: 500069028

A **Docker Swarm** is a group of either physical or virtual machines that are running the **Docker** application and that have been configured to join together in a cluster.

1. Initialize a docker swarm



2. To ass worker to cluster we will use token,

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

$ token=$(ssh -o StrictHostKeyChecking=no 172.17.0.76 "docker swarm join-token -q worker") && echo $token Warning: Permanently added '172.17.0.76' (ECDSA) to the list of known hosts.

SWMTKN-1-2wb28ayfd6nfha32ta5g9njhy7anzzhrmenjkjc1fyhtieyp0m-3n33qin3qpe585w58w5p4w6ez
```

3. Second node join via requesting access to the manager

```
Terminal Host 2

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

$ token=$(ssh -o StrictHostKeyChecking=no 172.17.0.76 "docker swarm join-token -q worker") && echo $token Warning: Permanently added '172.17.0.76' (ECDSA) to the list of known hosts.

SWMTKN-1-2wb28ayfd6nfha32ta5g9njhy7anzzhrmenjkjc1fyhtieyp0m-3n33qin3qpe585w58w5p4w6ez $ docker swarm join 172.17.0.76:2377 --token $token This node joined a swarm as a worker.

$
```

4. We will create a network, using which all nodes in the cluster can communicate with each other.

```
$ docker node ls
 ID
                               HOSTNAME
                                                   STATUS
                                                                        AVAILABILI
                               ENGINE VERSION
 TY
          MANAGER STATUS
 rmlioyfiyw35p4epn8cdvhf5m *
                               host01
                                                   Ready
                                                                        Active
          Leader
                               19.03.13
 v8p00ms7m3wz8kq2rv1qjh6fi
                               host02
                                                   Ready
                                                                        Active
                               19.03.13
 $ docker network create -d overlay skynet
 vsmpishtgfuediugzcd28921u
$ docker service create --name http --network skynet --replicas 2 -p 80:80 katac
oda/docker-http-server
109ajmqzphrxar4eimn4n6bib
overall progress: 2 out of 2 tasks
1/2: running
2/2: running
```

Check the running clusters

```
$ docker service ls

ID NAME MODE REPLICAS

IMAGE PORTS

109ajmqzphrx http replicated 2/2

katacoda/docker-http-server:latest *:80->80/tcp
```

7. we issue an HTTP request to the public port, it will be processed by the two containers

```
$ docker ps
CONTAINER ID
                                                         COMMAND
                   IMAGE
                                                                             CRE
                 STATUS
                                     PORTS
                                                         NAMES
ATED
259b50e1476d
                  katacoda/docker-http-server:latest
                                                         "/app"
inutes ago
                 Up 4 minutes
                                    80/tcp
                                                         http.2.x3vc7kfzao537jvr
gfb6y18mj
$ curl host01
<h1>This request was processed by host: 2f277a020e25</h1>
```

8. Listing container in second terminal

 \$ docker ps CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
2f277a020e25	katacoda/docker-http-server:latest pn2nafa10275if4i191jn6311	"/app"	4 minutes ago	Up 4 minutes	80/tcp

9. Viewing the list of all the tasks

\$ docker service ps http					
ID	NAME	IMAGE		NOD	
E PORTS	DESIRED STATE	CURRENT STATE	ERROR		
pn2nafa10275 t02	http.1 Running	katacoda/docker-l Running 6 minutes ad	http-server:latest go	hos	
x3vc7kfzao53 t01	http.2 Running	katacoda/docker-l Running 6 minutes ac	nttp-server:latest go	hos	

10. view the details and configuration of a service

```
$ docker service inspect --pretty http
ID:
               109ajmqzphrxar4eimn4n6bib
Name:
               http
Service Mode: Replicated
Replicas:
               2
Placement:
UpdateConfiq:
Parallelism:
On failure:
              pause
Monitoring Period: 5s
Max failure ratio: 0
Update order:
                  stop-first
RollbackConfig:
```

11. On each node, you can ask what tasks it is currently running. Self refers to the manager node Leader:

\$ docker node ps	s self		
ID	NAME	IMAGE	NOD
E PORTS	DESIRED STATE	CURRENT STATE E	RROR
x3vc7kfzao53 t01	http.2 Running	katacoda/docker-http-s Running 8 minutes ago	erver:latest hos

12. Using the ID of a node you can query individual hosts

```
$ docker node ps $(docker node ls -q | head -n1)

ID NAME IMAGE NOD

E DESIRED STATE CURRENT STATE ERROR

PORTS

x3vc7kfzao53 http.2 katacoda/docker-http-server:latest hos

t01 Running Running 9 minutes ago
```

13. you will see additional nodes being started

\$ docker ps				
CONTAINER ID	IMAGE		COMMAND	CRE
ATED	STATUS	PORTS	NAMES	
259b50e1476d	katacoda/dock	er-http-server:latest	"/app"	10
minutes ago gfb6y18mj	Up 10 minutes	80/tcp	http.2.x3vc7kf	Ezao537j v r