EXPERIMENT 7

AIM: Working with Docker Swarm

Steps to Complete:

Docker-Swarm

Prerequisite: Here, we will spin up 3 virtual machines (vagrant in our case) and setup swarm cluster with one manager and 2 node.

Create Swarm

Run the following command to create a new swarm:

```
docker swarm init
```

The following command creates a swarm on the manager machine: for example

```
docker swarm init --advertise-addr 192.168.99.100
```

OUTPUT:

```
Swarm initialized: current node (dxn1zf6l61qsb1josjja83ngz) is now a
manager.
```

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

```
docker swarm join \ --token SWMTKN-1-
49nj1cmql0jkz5s954yi3oex3nedyz0fb0xx14ie39trti4wxv-
8vxv8rssmk743ojnwacrr2e7c \192.168.99.100:2377
```

Run docker info to view the current state of the swarm:

```
docker info
```

Run the docker node ls command to view information about nodes:

```
docker node ls
```

```
ID       HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS
dxn1zf6l61qsb1josjja83ngz *  manager1  Ready   Active        Leader
```

Add nodes to the cluster

Run the command produced by the docker swarm init output from the Create a swarm tutorial step to create a worker node joined to the existing swarm:

```
docker swarm join --token SWMTKN-1-
49nj1cmql0jkz5s954yi3oex3nedyz0fb0xx14ie39trti4wxv-
8vxv8rssmk743ojnwacrr2e7c 192.168.99.100:2377
```

NOTE: Repeat above step in all your nodes which has to be part of swarm cluster

Now from the manager node run docker node ls command to check the status of the joined nodes

```
docker node ls
```

```
ID                           HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS
03g1y59jwfg7cf99w4lt0f662    worker2   Ready   Active
9j68exjopxe7wfl6yuxml7a7j    worker1   Ready   Active
dxn1zf6l61qsb1josjja83ngz *  manager1  Ready   Active        Leader
```

The MANAGER column identifies the manager nodes in the swarm. The empty status in this column for worker1 and worker2 identifies them as worker nodes.

Swarm management commands like docker node ls only work on manager nodes.

Deploy service

Open a terminal and ssh into the machine where you run your manager node. For example, the tutorial uses a machine named manager1.

```
docker service create --replicas 1 --name helloworld alpine ping docker.com
```

The docker service creates command creates the service. The --name flag names the service helloworld. The --replicas flag specifies the desired state of 1 running instance. The arguments alpine ping docker.com define the service as an Alpine Linux container that executes the command ping docker.com.

Run docker service ls to see the list of running services:

```
docker service ls
```

```
ID            NAME        SCALE  IMAGE   COMMAND
9uk4639qpg7n  helloworld  1/1    alpine  ping docker.com
```

Inspect the service

Run docker service inspect --pretty <SERVICE-ID> to display the details about a service in an easily readable format.
To see the details on the helloworld service:

```
docker service inspect --pretty helloworld
```

Run docker service ps <SERVICE-ID> to see which nodes are running the service:

```
docker service ps helloworld
```

Run docker ps on the node where the task is running to see details about the container for the task.

```
docker ps
```

Scale the service in the swarm

Run the following command to change the desired state of the service running in the swarm:

```
docker service scale <SERVICE-ID>=<NUMBER-OF-TASKS>

docker service scale helloworld=5
```

Run docker service ps <SERVICE-ID> to see the updated task list:

```
docker service ps helloworld
```

You can see that swarm has created 4 new tasks to scale to a total of 5 running instances of Alpine Linux. The tasks are distributed between the three nodes of the swarm. One is running on manager1.

Run docker ps to see the containers running on the node where you're connected. The following example shows the tasks running on manager1:

```
docker ps
```

Delete a service

Run docker service rm helloworld to remove the helloworld service.

```
docker service rm helloworld
```