Docker Swarm:

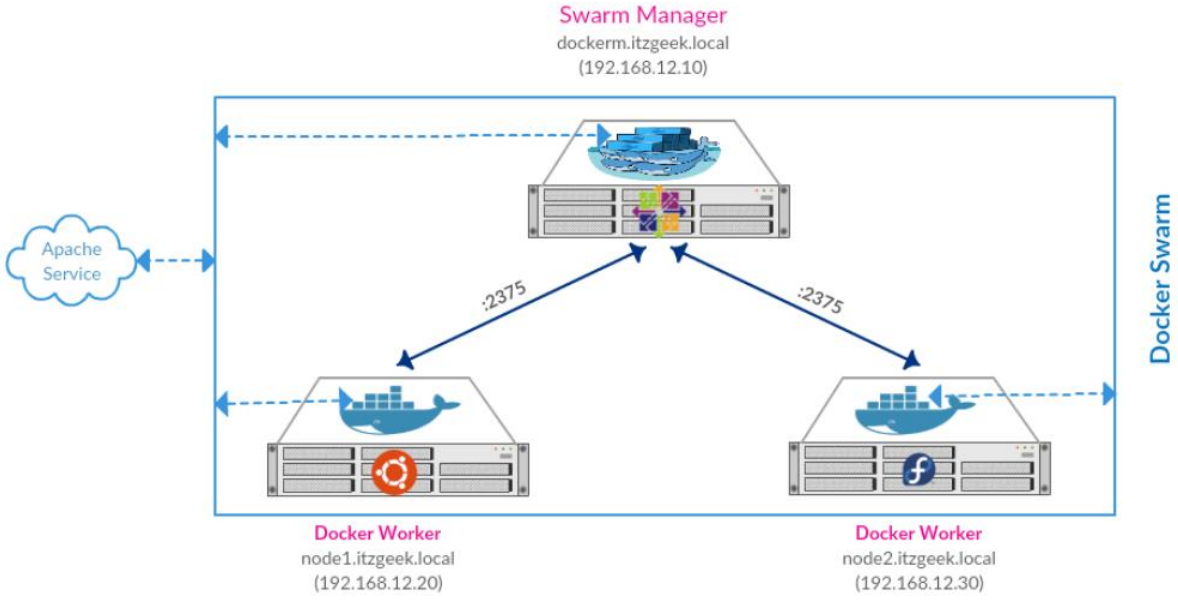
The Problem(s)

Container Orchestration:
Containers are great, but when you get lots of them running, at some point, you need them all working together to solve business problems.

The problem is, when you have lots of containers running, they need to be managed: you want enough of them running to handle the load, but not so many they are bogging down the machines in the cluster. And well, from time to time, containers are going to crash, and need to be restarted.

Docker Swarm

Docker swarm mode allows you to manage a cluster of Docker Engines, natively within the Docker platform. You can use the Docker CLI to create a swarm, deploy application services to a swarm, and manage swarm behaviour.



- Coordinate between containers and allocate tasks to groups of containers
- Perform health checks and manage lifecycle of individual containers
- Provide redundancy and failover in case nodes experience failure
- Scale the number of containers up and down depending on load
- Perform rolling updates of software across multiple containers

Lab :

1. Create one ec2 instance (called as manager), install docker

yum install -y docker

service docker start

2. Add node to docker swarm

docker swarm init

```
[root@ip-172-31-8-30 ec2-user]# docker swarm init
Swarm initialized: current node (jjaiubb1yc3smu3rfja5uwo6b) is now a manager.

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

    docker swarm join --token SWMTKN-1-45cj1l7ii0rxst1lvtgx23yik0emno2tizdhxw62d
oof9v32mr-4z1xxsc1rnk5ibnu9snd7i0f4 172.31.8.30:2377

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

3. Create one more instance of ec2 (called as worker node), install docker on it.

4. Now make this instance as worker node to swarm.

5. copy the output of (docker swarm init) command and paste on worker node.

```
[root@ip-172-31-6-152 ec2-user]# docker swarm join --token SWMTKN-1-45cj1l7ii0rx
st1lvtgx23yik0emno2tizdhxw62doof9v32mr-4z1xxsc1rnk5ibnu9snd7i0f4 172.31.8.30:237
7
This node joined a swarm as a worker.
```

6. Check if the node has been added to swarm, on manager run

docker node ls

```
[root@ip-172-31-8-30 ec2-user]# docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS
uxvh17am7c6hje6osvpye3iwr     ip-172-31-6-152     Ready               Active
jjaiubb1yc3smu3rfja5uwo6b *   ip-172-31-8-30      Ready               Active              Leader
```

7. Check networks with

docker network ls

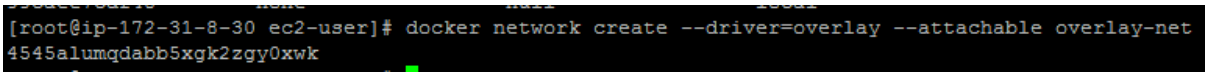
```
[root@ip-172-31-8-30 ec2-user]# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
aa66f0c0a651        bridge              bridge              local
26e942cac6d0        docker_gwbridge     bridge              local
60b44ff38da4        host                host                local
t8r4ufrkbped        ingress             overlay             swarm
99caee7cdf48        none                null                local
```

8. Each of them now has an overlay network called **ingress** and a bridge network called **docker_gwbridge**

// The docker_gwbridge connects the ingress network to the Docker host's network interface so that traffic can flow to and from swarm managers and workers. If you create swarm services and do not specify a network, they are connected to the ingress network. It is recommended that you use separate overlay networks for each application or group of applications which will work together. In the next procedure, you will create two overlay networks and connect a service to each of them.//

9. Now on manager node create an attachable overlay network called "Overlay-net"

docker network create --driver=overlay --attachable overlay-net

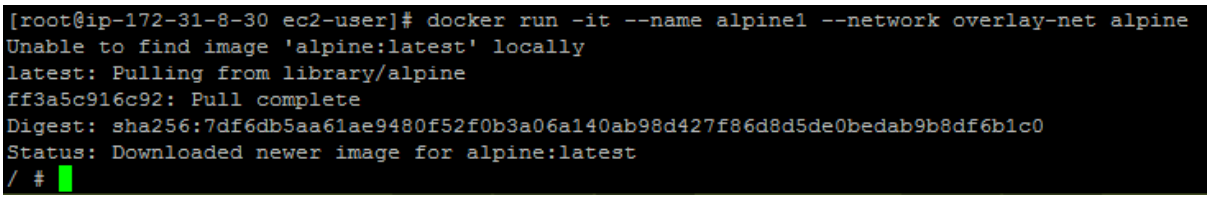


// Notice the returned NETWORK ID -- you will see it again when you connect to it from worker node//

10.New network has been created only on master but not on node

11. On manager node create an interactive container alpine1(Small. Simple. Secure. Alpine Linux is a security-oriented, lightweight Linux distribution based) that connect to overlay-net

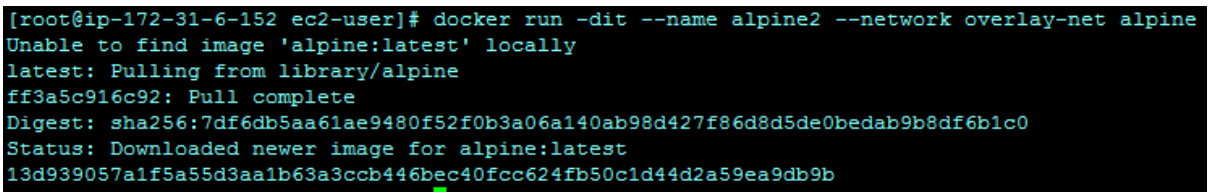
docker run -it --name alpine1 --network overlay-net alpine



12. Check on worker node, but the overlay-net is still not present (docker network ls)

13. On worker node created an alpine2 container in interactive & detached mode.

docker run -dit --name alpine2 --network overlay-net alpine



// Automatic DNS container discovery only works with unique container names//

14.Now overlay-net will be listed on worker node

15.Try to ping container alpine2 running (container must be running, thats why we have started it in detached mode) on worker node from apline1 running on manager node.

ping -c 2 alpine2

```
/ # ping -c 2 alpine2
PING alpine2 (10.0.0.5): 56 data bytes
64 bytes from 10.0.0.5: seq=0 ttl=255 time=0.535 ms
64 bytes from 10.0.0.5: seq=1 ttl=255 time=0.688 ms

--- alpine2 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.535/0.611/0.688 ms
/ #
```

16. Ping apline1 from alpine2 and should be able to reach.

```
/ # ping -c 2 alpine1
PING alpine1 (10.0.0.6): 56 data bytes
64 bytes from 10.0.0.6: seq=0 ttl=255 time=0.611 ms
64 bytes from 10.0.0.6: seq=1 ttl=255 time=0.693 ms

--- alpine1 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.611/0.652/0.693 ms
```