**Install Oracle JDK 1.8 in AWS EC2 instance**

$ wget --no-check-certificate --no-cookies --header "Cookie: oraclelicense=accept-securebackup-cookie" <http://download.oracle.com/otn-pub/java/jdk/8u141-b15/336fa29ff2bb4ef291e347e091f7f4a7/jdk-8u141-linux-x64.rpm>

$sudo yum install -y jdk-8u141-linux-x64.rpm

$sudo alternatives --config java

There are 2 programs which provide 'java'.

Selection Command

-----------------------------------------------

\*+ 1 /usr/lib/jvm/jre-1.7.0-openjdk.x86\_64/bin/java

2 /opt/jdk1.8.0\_151/bin/java

Enter to keep the current selection[+], or type selection number: 2

**Docker Installation on Linux:**

1. Setup the proxy parameters
2. yum install -y yum-utils device-mapper-persistent-data lvm2
3. yum-config-manager --add-repo <https://download.docker.com/linux/centos/docker-ce.repo>
4. yum install docker-ce

or

1. If you have the docker.repo in any other host under /etc/yum.repos.d, copy it onto the same path on the new host
2. Yum install docker-engine

**Docker installation on AWS EC2 instance:**

1. Connect to the EC2 instance
2. sudo yum update –y
3. sudo yum install docker

[ec2-user@ip-172-31-19-2 ~]$ docker version

Client:

Version: 18.03.1-ce

API version: 1.37

Go version: go1.9.4

Git commit: 3dfb8343b139d6342acfd9975d7f1068b5b1c3d3

Built: Thu May 24 22:21:27 2018

OS/Arch: linux/amd64

Experimental: false

Orchestrator: swarm

Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running?

[ec2-user@ip-172-31-19-2 ~]$ sudo service docker start

Starting cgconfig service: [ OK ]

Starting docker: . [ OK ]

[ec2-user@ip-172-31-19-2 ~]$

**What happens if we try to create a swarm on a host on which Swarm is already Active?**

bash-4.2$ docker swarm init --advertise-addr 10.0.0.1

Error response from daemon: This node is already part of a swarm. Use "docker swarm leave" to leave this swarm and join another one.

bash-4.2$ docker swarm leave

Error response from daemon: You are attempting to leave the swarm on a node that is participating as a manager. Removing the last manager erases all current state of the swarm. Use `--force` to ignore this message.

bash-4.2$ docker swarm leave --force

Node left the swarm.

**Docker Swarm Explained!!**

bash-4.2$ docker swarm init --advertise-addr 10.0.0.1

Swarm initialized: current node (sta2u1iemi4n9rmo51xg3pw51) is now a manager.

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

docker swarm join \

--token SWMTKN-1-3iecp1mx49zdk1sigss20pyky85jcrjbzhnhd4oc6q8kwrjqvp-e30pvhqz1ghu9l3t527k3cnnj \

10.0.0.1:2377

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

bash-4.2$

With the above command, in addition to the swarm init, we also specified the --advertise-addr flag with a value of 10.0.0.1. This is the IP that the Swarm node manager will use to advertise the Swarm Cluster Service. While this address can be a private IP, it’s important to note that in order for nodes to join this swarm, those nodes will need to be able to connect to the node manager over this IP on port 2377.

After running the docker swarm init command, we can see that the swarm was given a node name (**sta2u1iemi4n9rmo51xg3pw51**) and made the manager of this swarm. The output also supplies two commands: one command is to add a node worker to the swarm and the other is to add another node manager to the swarm.

Docker Swarm Mode can support multiple node managers. It will, however, elect one of them to be the primary node manager which will be responsible for orchestration within the Swarm.

#### Adding a node worker to the Swarm Cluster

With the Swarm Cluster created, we can now add a new node worker using the docker command provided by the output of the Swarm creation.

**Issue: Error response from daemon: --live-restore daemon configuration is incompatible with swarm mode**

docker swarm join \

> --token SWMTKN-1-3iecp1mx49zdk1sigss20pyky85jcrjbzhnhd4oc6q8kwrjqvp-e30pvhqz1ghu9l3t527k3cnnj \

> 10.0.0.1:2377

Error response from daemon: --live-restore daemon configuration is incompatible with swarm mode

**Solution: Set the live-restore parameter to false in the below file**

vi /etc/docker/daemon.json

{

"ipv6": true,

"live-restore": true,

"fixed-cidr-v6": "2017:db8:1::/64",

"insecure-registries" : ["den01swq.us.oracle.com:5000"]

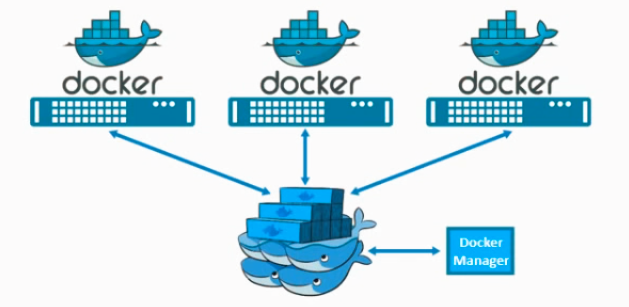
}

**Docker Swarm:**

Docker Swarm is a technique to create and maintain a cluster of Docker Engines. So there would be many docker engines connected to each other forming a network which is nothing but a Docker Swarm.

Service deployed in any node can be accessed on other nodes in the same cluster. Services can also run on the Manager Node, however the prime responsibility of it is to manage the services running on the nodes.

**Docker Swarm Architecture:**



**Features of Docker Swarm:**

**High Availability of Services:** If the node on which a service is hosted goes down, the service can be served by the other nodes in the Swarm and the service would be started on a new node by the Swarm Manager.

**Auto Load Balancing:**

**Decentralized Access:** We can access the Manager or Nodes from anywhere by SSH ‘ing onto the host on which they are residing. If you SSH onto the Docker Manager host, we can manage the other nodes and not vice versa.

**Easy to scale up Deployments:**

**Rolling Updates:** At some point of time, we might have to update the services, but it is n ‘t feasible to manually logon to each host and make the update. Instead, we ca make use of “delay”, each service gets updated one at a time and the others can still serve the traffic and after the set delay, the service on the next host is also updated.