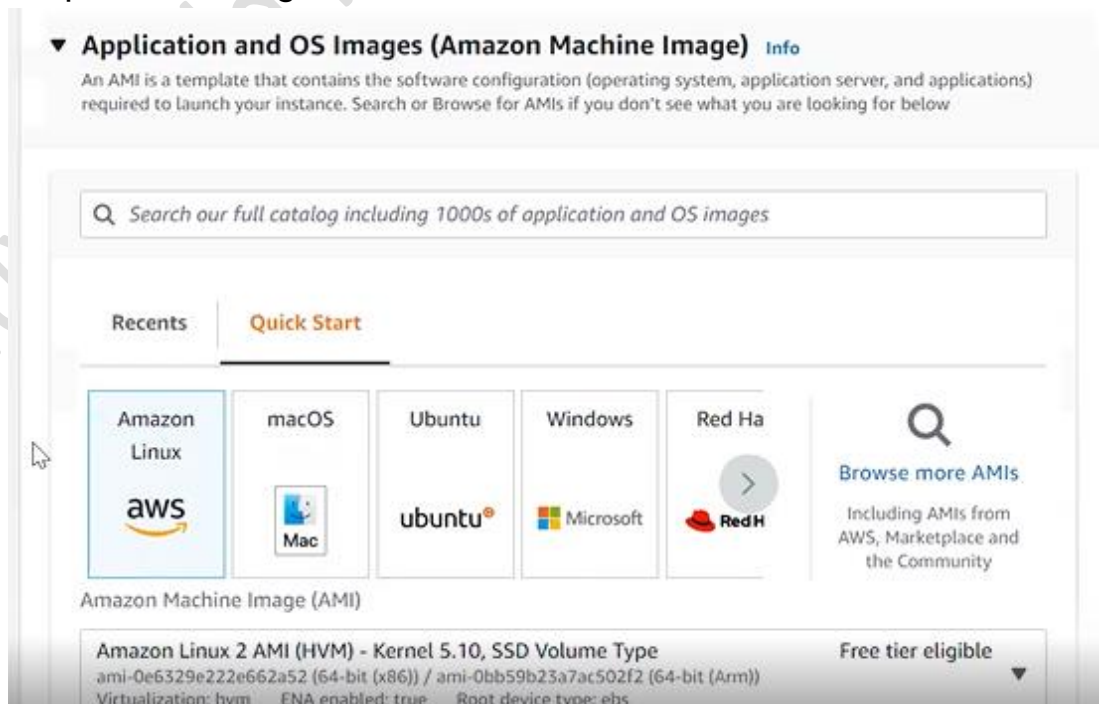




Summary

Session No – 15

- For launching the container minimum requirement is hardware resources & operating system
- If we run a website on a container it is completely dependent on the host system's hardware & if there is a possibility that our operating system goes down then the entire website goes down. Here we can say our operating system is a single point of failure
- Adding more resources (CPU / RAM / HD) on one operating system is called vertical scaling
- One of the limitations of vertical scaling is downtime
- Adding additional machines to our infrastructure is called horizontal scaling
- In master-slave architecture one device or system (Master) controls the other devices (Slave)
- Master-slave cluster setup
 - Step 1: launching 4 amazon Linux instances on AWS cloud



Number of instances [Info](#)

4

When launching more than 1 instance, [consider EC2 Auto Scaling](#).

Software Image (AMI)

Amazon Linux 2 Kernel 5.10 AMI...[read more](#)
ami-0e6329e222e662a52

Virtual server type (instance type)

t2.micro

Firewall (security group)

- Step 2:- Installing Docker in all the 4 instances

```
[root@ip-172-31-2-3 ~]#
[root@ip-172-31-2-3 ~]# yum install docker -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
```

- Step 3 :- Starting docker services

```
[root@ip-172-31-2-3 ~]# systemctl start docker
[root@ip-172-31-2-3 ~]#
[root@ip-172-31-2-3 ~]# docker info
```

- Step 4:- (**ON MASTER**) Edit the inbound rule or create a new rule

EC2 > Security Groups > sg-0ad12b58546935a5a - launch-wizard-14 > Edit inbound rules

Edit inbound rules [Info](#)

Inbound rules control the incoming traffic that's allowed to reach the instance.

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info
-	All traffic	All	All	Anywh... Q	

[Add rule](#)

[Cancel](#) [Preview changes](#) [Save rules](#)

- Step 5:- Pinging from slave to master (Connectivity check)

```
[root@ip-172-31-3-99 ~]# ping 172.31.2.3
PING 172.31.2.3 (172.31.2.3) 56(84) bytes of data.
64 bytes from 172.31.2.3: icmp_seq=1 ttl=255 time=0.574 ms
64 bytes from 172.31.2.3: icmp_seq=2 ttl=255 time=0.433 ms
64 bytes from 172.31.2.3: icmp_seq=3 ttl=255 time=0.432 ms
64 bytes from 172.31.2.3: icmp_seq=4 ttl=255 time=0.440 ms
```

- Step 6:- (**ON MASTER**) Initializing cluster
 - Command:- **docker swarm init --advertise-addr (Master Ip)**

```
[root@ip-172-31-2-3 ~]# docker swarm init --advertise-addr 172.31.2.3
Swarm initialized: current node (uvykc1f39gvwckh2ycliwc2zs) is now a manager.

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

    docker swarm join --token SWMTKN-1-2w2ua5nhvc3ivhpbnf35prhifxt3wbi96f9w9h7vrurpxr7ztf-0io6cv0tgwhcbjgi526rwqq3f 172.31.2.3:2377

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

- Step 7:- Listing nodes in cluster
 - Command :- **docker node ls**

```
[root@ip-172-31-2-3 ~]# docker node ls
ID                                HOSTNAME                                STATUS  AVAILABILITY  MANAGER STATUS  EN
GINE VERSION
uvykc1f39gvwckh2ycliwc2zs *      ip-172-31-2-3.ap-south-1.compute.inte  Ready  Active        Leader          20
.10.17
```

- Step 8:- For adding a worker node they give a pre-created command at the time of initializing the cluster

```
[root@ip-172-31-2-3 ~]# docker swarm init --advertise-addr 172.31.2.3
Swarm initialized: current node (uvykc1f39gvwckh2ycliwc2zs) is now a manager.

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

    docker swarm join --token SWMTKN-1-2w2ua5nhvc3ivhpbnf35prhifxt3wbi96f9w9h7vrurpxr7ztf-0io6cv0tgwhcbjgi526rwqq3f 172.31.2.3:2377

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

- Step 9 :- (**ON SLAVE**) Joining slave to master

```
[root@ip-172-31-3-99 ~]# docker swarm join --token SWMTKN-1-2w2ua5nhvc3ivhpbnf35prhifxt3wbi96f9w9h7vrurpxr7ztf-0io6cv0tgwhcbjgi526rwqq3f 172.31.2.3:2377
This node joined a swarm as a worker.
[root@ip-172-31-3-99 ~]#
```

- Listing the node in cluster

```
ip-172-31-2-3.ap-south-1.compute.inte
[root@ip-172-31-2-3 ~]# docker node ls
ID                                HOSTNAME                                STATUS  AVAILABILITY  MANAGER STATUS  E
NGINE VERSION
uvykc1f39gvwckh2ycliwc2zs *      ip-172-31-2-3.ap-south-1.compute.inte  Ready  Active        Leader          2
0.10.17
idxdku7aor668iuol8q8yq2tq      ip-172-31-3-99.ap-south-1.compute.inte  Ready  Active                                     2
0.10.17
[root@ip-172-31-2-3 ~]#
```

- If we run the **docker info** command on the master we can see swarm is active and managing worker nodes

```
Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
Swarm: active
NodeID: uvykc1f39gvwckh2ycliwc2zs
Is Manager: true
ClusterID: maoobafkvdvrw5oodqamdwxoe
Managers: 1
Nodes: 2
Default Address Pool: 10.0.0.0/8
SubnetSize: 24
Data Path Port: 4789
Orchestration:
  Task History Retention Limit: 5
Raft:
  Snapshot Interval: 10000
  Number of Old Snapshots to Retain: 0
  Heartbeat Tick: 1
```

- Launching container in the swarm cluster
 - Command:- **docker service create --name webserver httpd**

```
[root@ip-172-31-2-3 ~]# docker service ls
ID                NAME          MODE          REPLICAS    IMAGE          PORTS
[root@ip-172-31-2-3 ~]# docker service create --name webserver httpd
nzdpo5nb806ofmhbqj9qu8aj2
overall progress: 1 out of 1 tasks
1/1: running [=====>]
verify: Service converged
[root@ip-172-31-2-3 ~]# docker service ls
ID                NAME          MODE          REPLICAS    IMAGE          PORTS
nzdpo5nb806o     webserver     replicated    1/1         httpd:latest
```