CONTAINER ORCHESTRATION AND INFRASTRUCTURE AUTOMATION

NAME- SHUBHI DIXIT

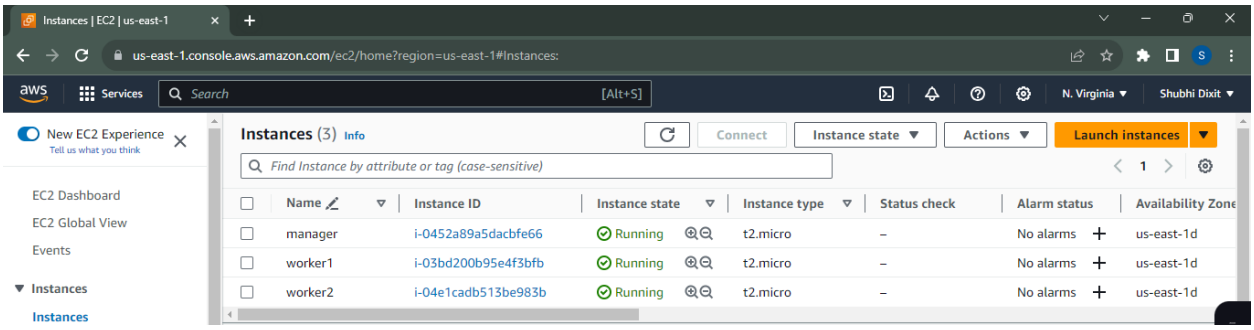
BATCH- 05

SAP ID- 500094571

DOCKER SWARM

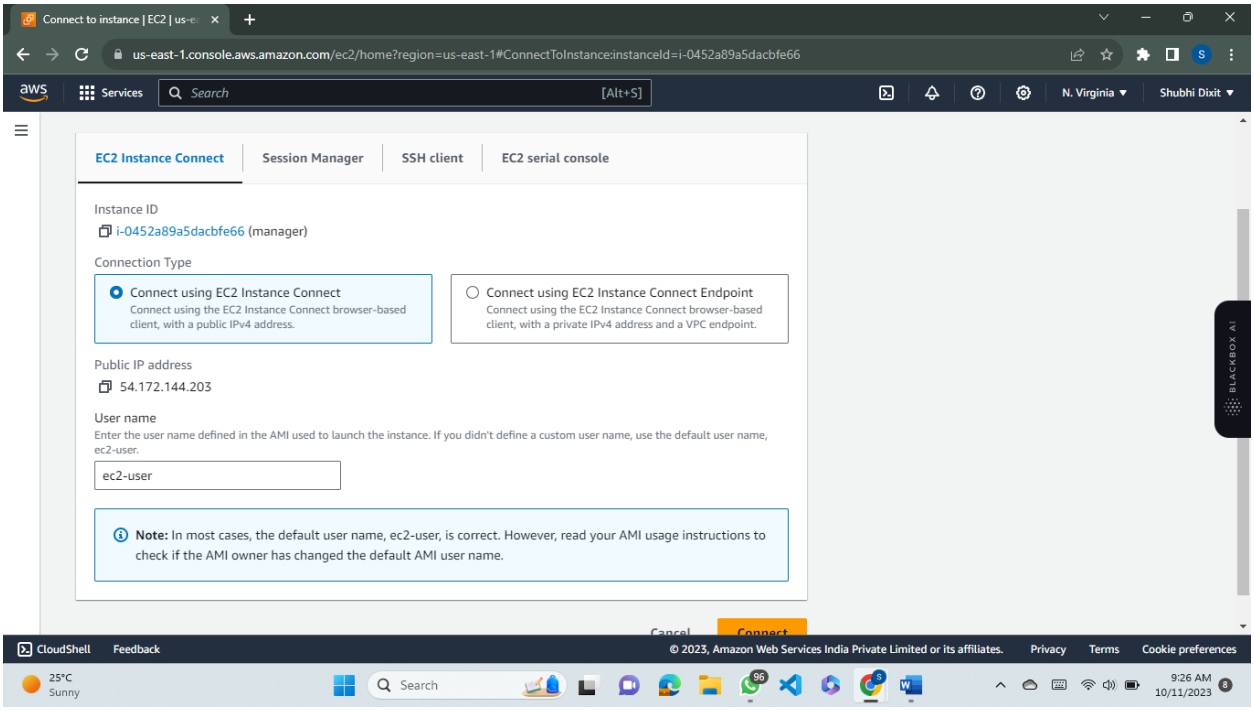
- Create a Docker Swarm having one manager and 2 worker nodes
- Deploy a service to the swarm and see the list of running services
- Inspect the service
- Delete the service
- Apply rolling updates
- Drain the node on the swarm
- Promote and Demote a node

Step-1: Go to aws management console and create three EC2 linux instances.



Note: Make sure you add “All TCP” and “All UDP” in the security groups of all the three instances.

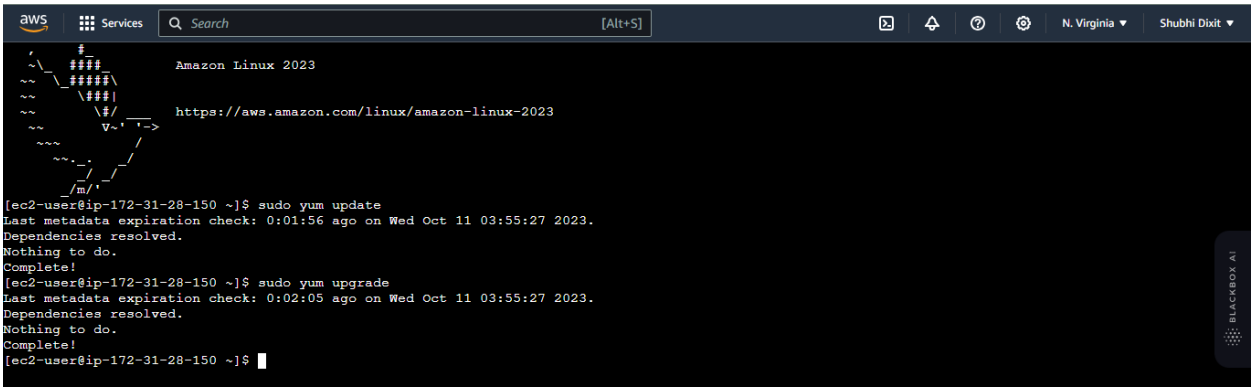
Step-2: Now select the “manager” instance and click on connect.



Step-3: After connecting to it successfully, update and upgrade the linux instance

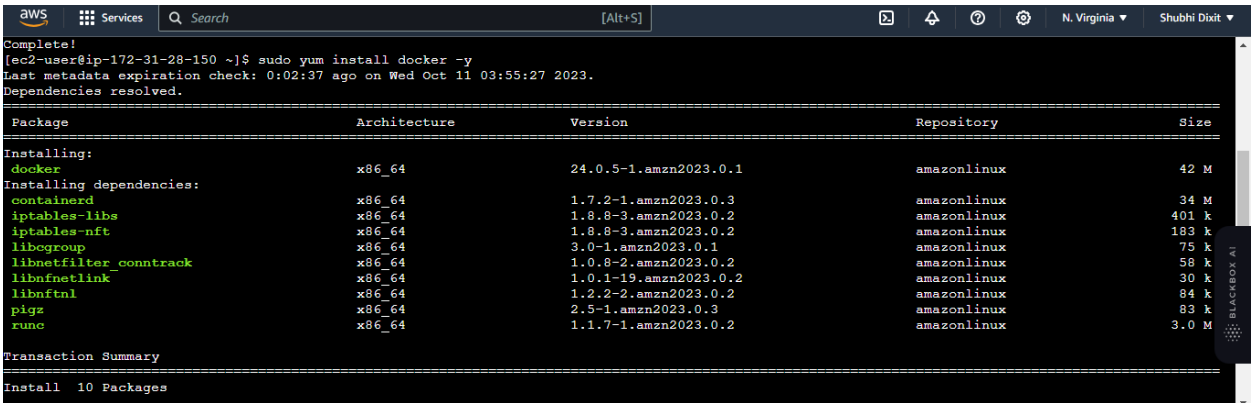
sudo yum update

sudo yum upgrade



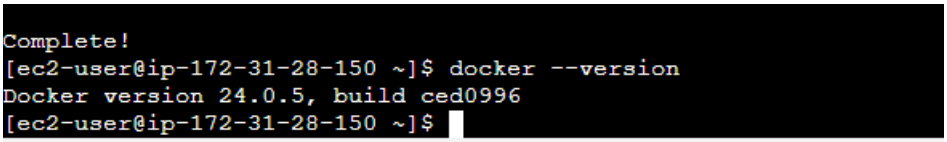
Step-4: Install docker in the instance.

sudo yum install docker -y



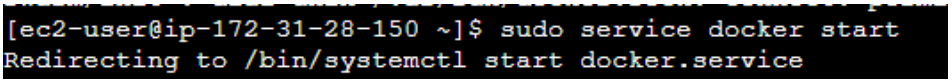
Step-5: Check the docker version and check whether it is installed successfully or not.

docker --version



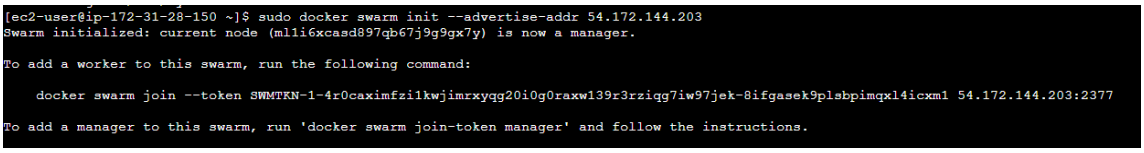
Step-6: Start the docker service.

sudo service docker start



Step-7: Select the manager and then copy its “public IP”.

Step-8: Now initialise swarm and paste the IP address of the manager node that you copied in the following command:



sudo docker swarm init - -advertise-addr <public-ip-of-same-machine>

Here copy the docker swarm join command.

Step-9: Go to the worker instances and repeat the steps from 1-6 in the instances.

Step-10: Now after performing the above steps, paste the “docker swarm join …” command- **(worker1)**

sudo service docker start

```
[ec2-user@ip-172-31-30-62 ~]$ sudo service docker start
Redirecting to /bin/systemctl start docker.service
[ec2-user@ip-172-31-30-62 ~]$ sudo docker swarm join --token SWMTKN-1-4r0caximfzi1kwjimrxyqg20i0g0raxw139r3rziqg7iw97jek-8ifgasek9plsbpimqxl4icxm1 
172.144.203:2377
This node joined a swarm as a worker.
[ec2-user@ip-172-31-30-62 ~]$
```

i-03bd200b95e4f3bfb (worker1)

```
[ec2-user@ip-172-31-30-205 ~]$ sudo service docker start
Redirecting to /bin/systemctl start docker.service
[ec2-user@ip-172-31-30-205 ~]$ sudo docker swarm join --token SWMTKN-1-4r0caximfzi1kwjimrxyqg20i0g0raxw139r3rziqg7iw97jek-8ifgasek9plsbpimqxl4icxm
.172.144.203:2377
This node joined a swarm as a worker.
[ec2-user@ip-172-31-30-205 ~]$
```

i-04e1cadb513be983b (worker2)

Step-11: Check the swarm status and the number of nodes in the manager.

sudo docker info

```
[ec2-user@ip-172-31-28-150 ~]$ sudo docker info
Client:
 Version:    24.0.5
 Context:    default
 Debug Mode: false
 Plugins:
  buildx: Docker Buildx (Docker Inc.)
    Version:  v0.0.0+unknown
    Path:     /usr/libexec/docker/cli-plugins/docker-buildx
```

Step-11: (manager) Check the available nodes.

sudo docker node ls

```
[ec2-user@ip-172-31-28-150 ~]$ sudo docker node ls
ID                            HOSTNAME                          STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
ml1i6xcasd897qb67j9g9gx7y *   ip-172-31-28-150.ec2.internal     Ready     Active         Leader           24.0.5
zglqr74w5c4o4oqu3eguch6ti     ip-172-31-30-62.ec2.internal      Ready     Active                          24.0.5
k51o2wec5zhl1wqjka49f4rhg     ip-172-31-30-205.ec2.internal     Ready     Active                          24.0.5
[ec2-user@ip-172-31-28-150 ~]$
```

Create a service

Step-4: (manager) Run the below command. This command creates a Docker service named "new-service" with 3 replicas running the "nginx:latest" image, mapping port 80 from the host to port 80 in the container.

sudo docker service create - -name new-service - -replicas 3 -p 80:80 nginx:latest

```
[ec2-user@ip-172-31-28-150 ~]$ sudo docker service create --replicas 3 --name helloworld3 alpine ping docker.com
its9l13ria2wnjxit1nqx5nrx
overall progress: 3 out of 3 tasks
1/3: running   [==================================================>]
2/3: running   [==================================================>]
3/3: running   [==================================================>]
verify: Service converged
[ec2-user@ip-172-31-28-150 ~]$
```

Step-5: (manager) Check whether the service is created or not.

sudo docker service ls

```
[ec2-user@ip-172-31-28-150 ~]$ sudo docker service ls
ID             NAME              MODE          REPLICAS   IMAGE                PORTS
9sijhy7f3hr2   helloworld        replicated    0/2        alpine:latest
pb39ih5t0h1f   helloworld1       replicated    1/1        alpine:latest
its9l13ria2w   helloworld3       replicated    3/3        alpine:latest
3mn6grsz2eqf   peaceful_cohen    replicated    0/1        newservice:latest
94arcum92wg5   service           replicated    0/2        alpine:latest
oyuh1l6co9rk   shubhi            replicated    1/2        nginx:latest
[ec2-user@ip-172-31-28-150 ~]$
```

Step-6: (manager) Check whether the container is created or not.

sudo docker ps

```
[ec2-user@ip-172-31-28-150 ~]$ sudo docker ps
CONTAINER ID   IMAGE           COMMAND             CREATED              STATUS              PORTS     NAMES
25a5ea9d513a   alpine:latest   "ping docker.com"   About a minute ago   Up About a minute             helloworld3.3.nmgx1gf42kteo0obs6cggtlwg
2293afdc823c   alpine:latest   "ping docker.com"   3 minutes ago        Up 3 minutes                  helloworld1.1.s595gvtmdjnjneiv4jqrom1fz
[ec2-user@ip-172-31-28-150 ~]$
```

Step-7: (worker1) Check whether the container is created or not on worker nodes as well.

sudo docker ps

```
[ec2-user@ip-172-31-30-205 ~]$ sudo docker ps
CONTAINER ID   IMAGE           COMMAND             CREATED              STATUS              PORTS     NAMES
2da02a252965   alpine:latest   "ping docker.com"   About a minute ago   Up About a minute             helloworld3.2.2fap00jlkd9aeuw5j5breiq92
[ec2-user@ip-172-31-30-205 ~]$
```

Step-8: (worker2) Check whether the container is created or not.

sudo docker ps

```
[ec2-user@ip-172-31-30-62 ~]$ sudo docker ps
CONTAINER ID   IMAGE           COMMAND             CREATED              STATUS              PORTS     NAMES
da6cf045d686   alpine:latest   "ping docker.com"   About a minute ago   Up About a minute             helloworld3.1.peufaflenwvgu21c04nsl8v69
[ec2-user@ip-172-31-30-62 ~]$
```

Note: Now you can copy the public IPs of the instances and check them on browser whether they are accessible or not.

Step-9: **(worker2)** Now let's remove the container that is created on the worker2 node.

sudo docker rm -f <container-id>

```
[ec2-user@ip-172-31-30-62 ~]$ sudo docker rm -f da6cf045d686
da6cf045d686
[ec2-user@ip-172-31-30-62 ~]$
```

Step-10:(worker2) After a few seconds, the manager detects that the container is stopped and not working on the worker node so it relaunches the container on that node.

sudo docker ps

Step 11: sudo docker service inspect - -pretty old-service to inspect a service in a pretty format

```
[ec2-user@ip-172-31-28-150 ~]$ sudo docker service inspect --pretty redis

ID:             8u71leweeiwukaspbas2q0q5o
Name:           redis
Service Mode:   Replicated
 Replicas:      3
Placement:
UpdateConfig:
 Parallelism:   1
 On failure:    pause
 Monitoring Period: 5s
 Max failure ratio: 0
 Update order:      stop-first
RollbackConfig:
 Parallelism:   1
 On failure:    pause
```

Rolling Updates

Step-1: (manager) Using the update command we can update the version of the redis service from 3.0.6 to 3.0.7

sudo docker service update - -image redis:3.0.7 redis

```
[ec2-user@ip-172-31-28-150 ~]$ sudo docker service update --image redis:3.0.7 redis
redis
overall progress: 3 out of 3 tasks
1/3: running   [==================================================>]
2/3: running   [==================================================>]
3/3: running   [==================================================>]
verify: Service converged
[ec2-user@ip-172-31-28-150 ~]$
```

Draining a Node

Step-1: (manager) The below command sets a worker node's availability to "drain" in a Docker Swarm cluster, meaning it stops and reschedules containers on that node to prepare it for maintenance or removal.

sudo docker node update - - availability drain <workernode-id>

```
[ec2-user@ip-172-31-28-150 ~]$ sudo docker node update --availability drain zglqr74w5c4o4oqu3eguch6ti
zglqr74w5c4o4oqu3eguch6ti
```

Step-2: (worker2) Check the status of the node. It shows “Drain”

sudo docker node ls

```
[ec2-user@ip-172-31-28-150 ~]$ sudo docker node ls
ID                            HOSTNAME                          STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
ml1i6xcasd897qb67j9g9gx7y *   ip-172-31-28-150.ec2.internal     Ready     Active         Leader           24.0.5
zglqr74w5c4o4oqu3eguch6ti     ip-172-31-30-62.ec2.internal      Ready     Drain                           24.0.5
k51o2wec5zhl1wqjka49f4rhg     ip-172-31-30-205.ec2.internal     Ready     Active                          24.0.5
```

Note: At this time no container will be running on the testworker1 because we have drained it.

Step-3: (manager) In the same way, we can set the status to “active” again.

sudo docker node update - - availability active <workernode-id>

```
[ec2-user@ip-172-31-28-150 ~]$ sudo docker node update --availability active zglqr74w5c4o4oqu3eguch6ti
zglqr74w5c4o4oqu3eguch6ti
[ec2-user@ip-172-31-28-150 ~]$
```

Promote and demote a Node

```
[ec2-user@ip-172-31-28-150 ~]$ sudo docker node promote zglqr74w5c4o4oqu3eguch6ti
Node zglqr74w5c4o4oqu3eguch6ti promoted to a manager in the swarm.
[ec2-user@ip-172-31-28-150 ~]$ sudo docker node ls
ID                            HOSTNAME                          STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
ml1i6xcasd897qb67j9g9gx7y *   ip-172-31-28-150.ec2.internal     Ready     Active         Leader           24.0.5
zglqr74w5c4o4oqu3eguch6ti     ip-172-31-30-62.ec2.internal      Ready     Active         Reachable        24.0.5
k51o2wec5zhl1wqjka49f4rhg     ip-172-31-30-205.ec2.internal     Ready     Active                          24.0.5
[ec2-user@ip-172-31-28-150 ~]$ sudo docker node demote ml1i6xcasd897qb67j9g9gx7y
Manager ml1i6xcasd897qb67j9g9gx7y demoted in the swarm.
[ec2-user@ip-172-31-28-150 ~]$ sudo docker node ls
Error response from daemon: This node is not a swarm manager. Worker nodes can't be used to view or modify cluster state. Please run this command
 a manager node or promote the current node to a manager.
[ec2-user@ip-172-31-28-150 ~]$
```