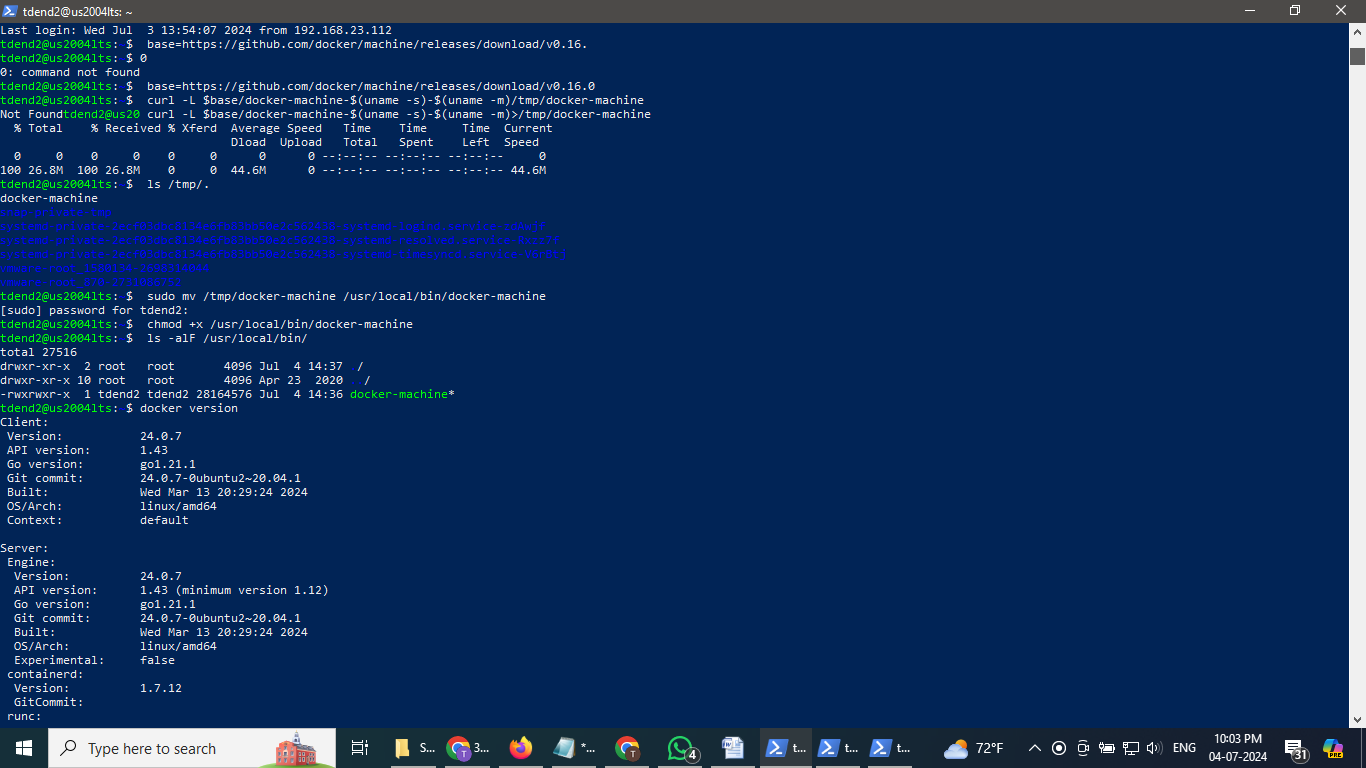
**Deploying MultiContainer Apps on Multi-nodes with Swarm**

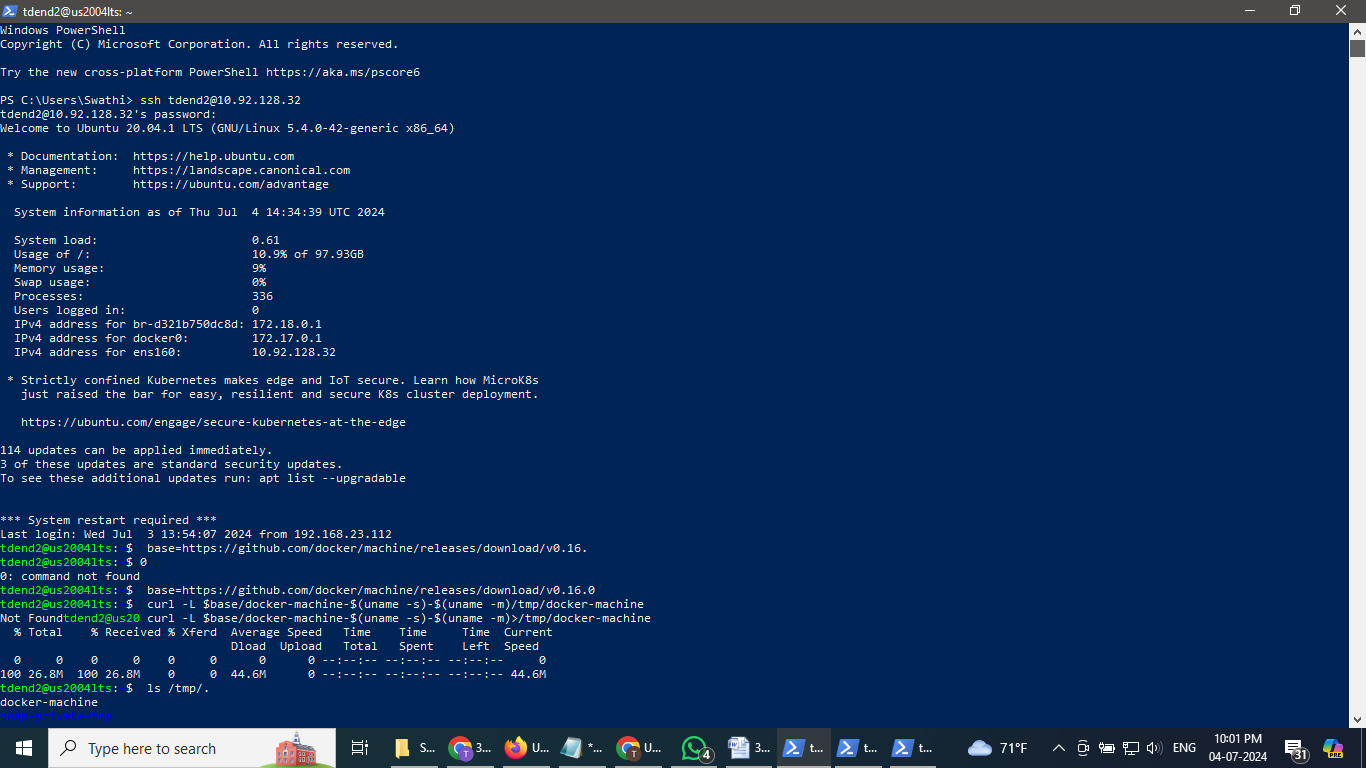
a. Section 1.3 Preparing multiple virtual nodes using Docker Machine

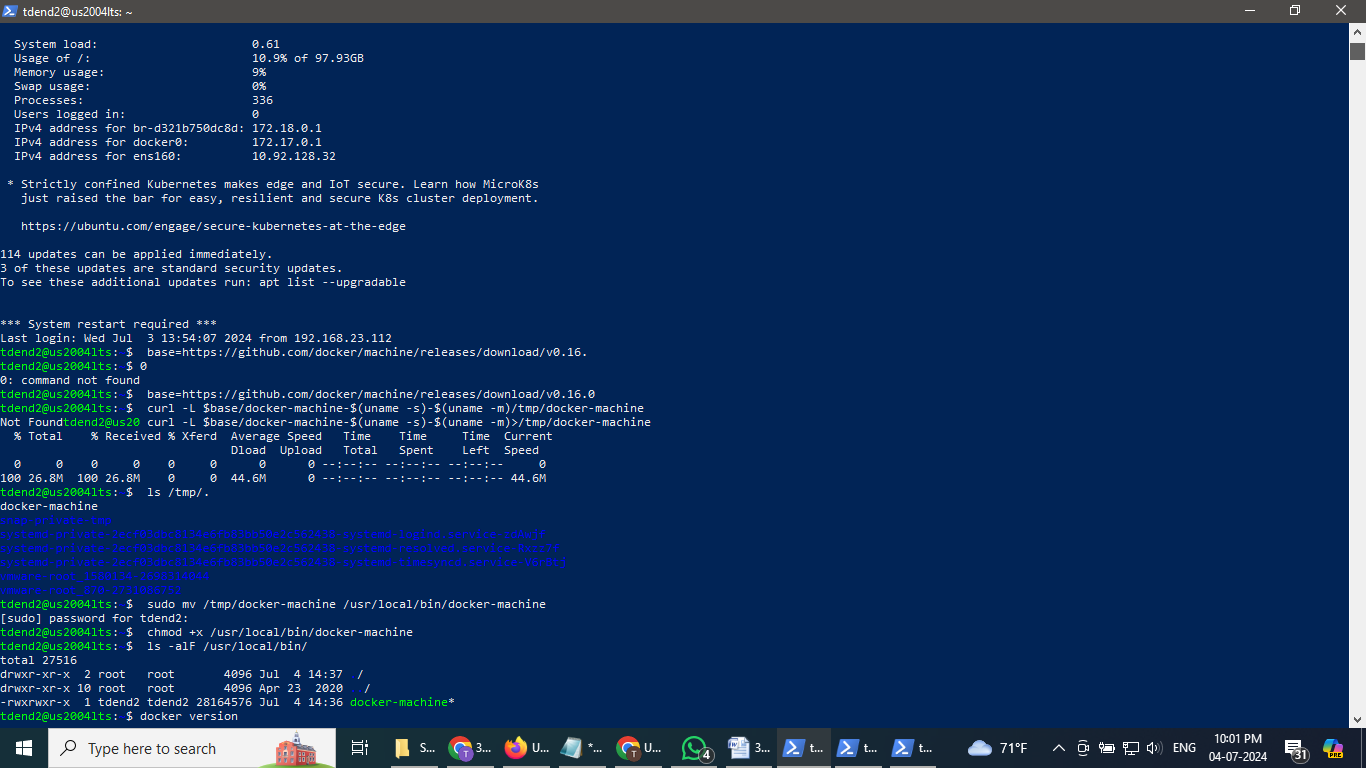
Created multiple virtual machines using the Docker machine as shown below. Docker Machine is a tool that is used to install Docker Engine on virtual hosts and manage the hosts with docker-machine commands. We use a Docker machine to create Docker hosts on our local Mac or Windows box, on our company network, in our data center, or on cloud providers like Azure, AWS, or Digital Ocean.

1.3.1 Installing docker-machine

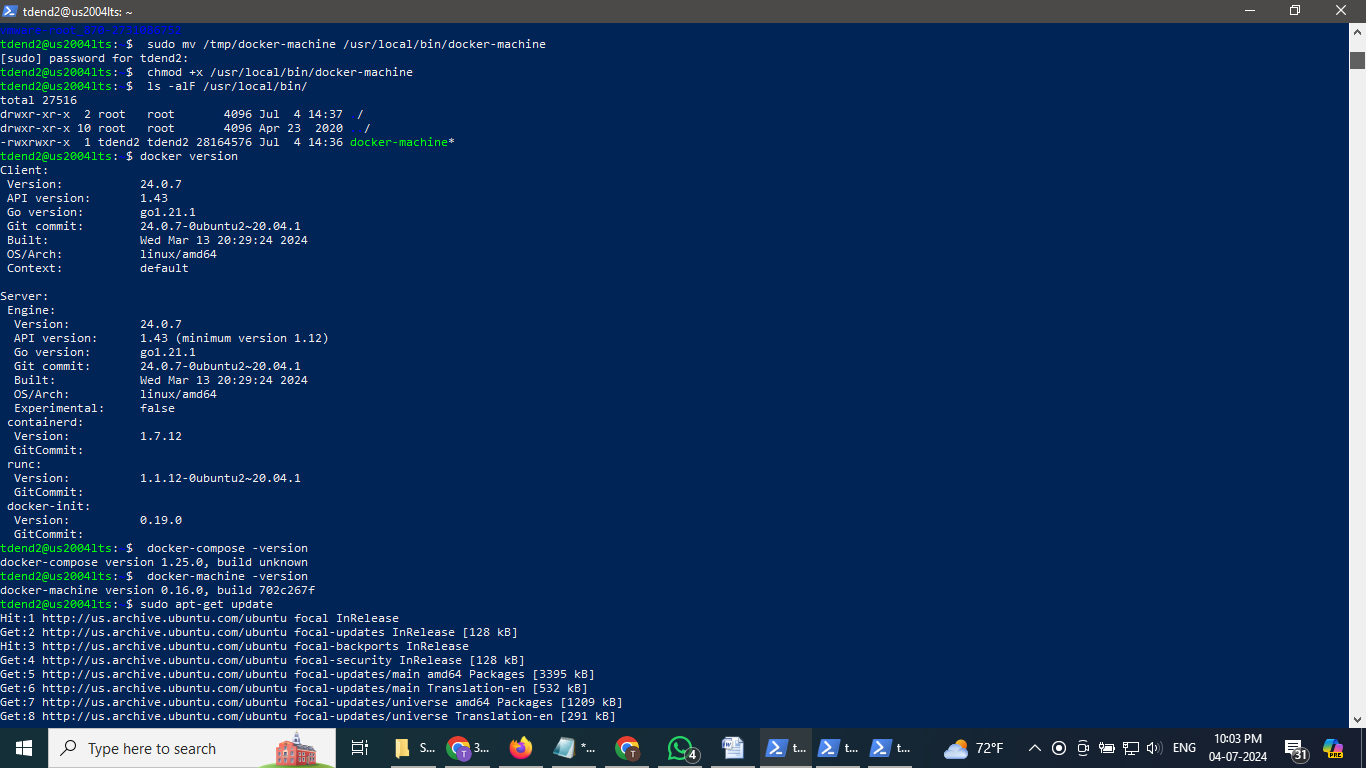
Downloaded the Docker Machine binary and extracted it to PATH

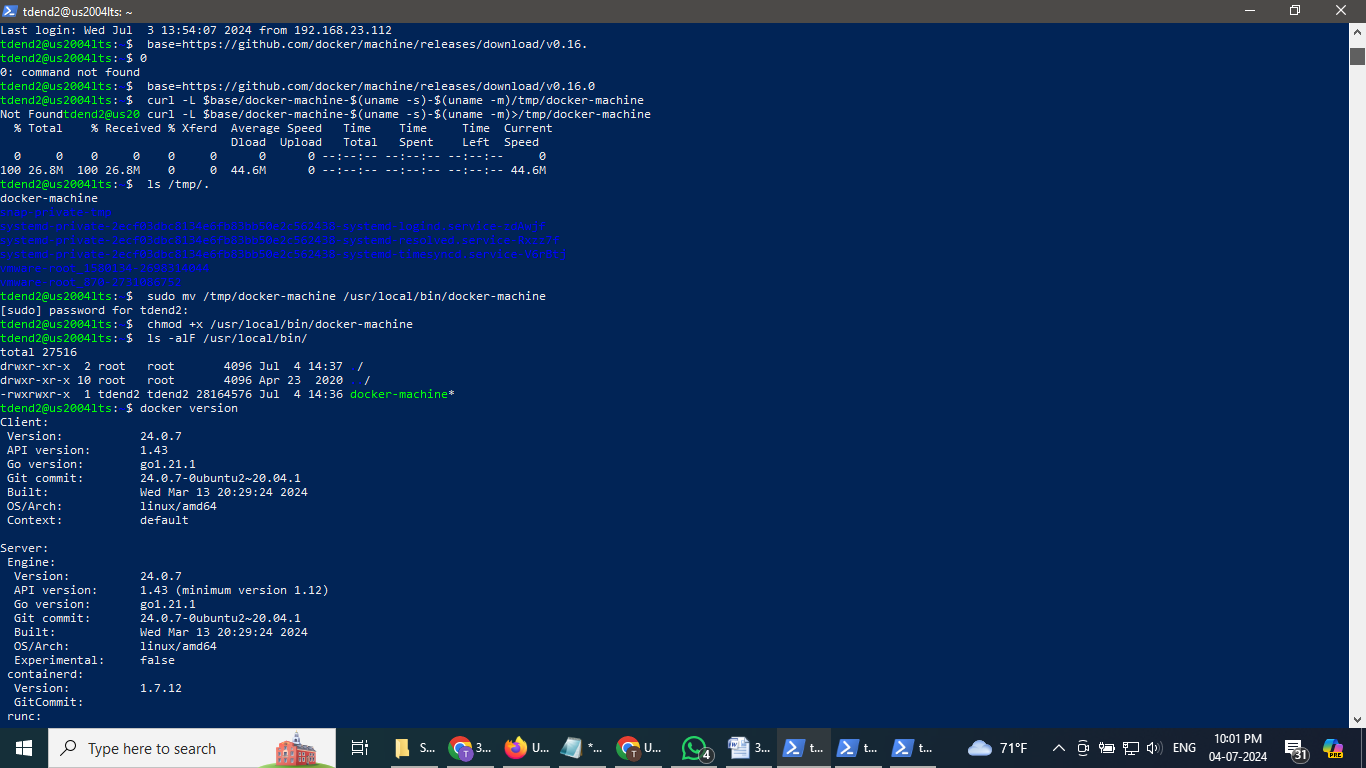




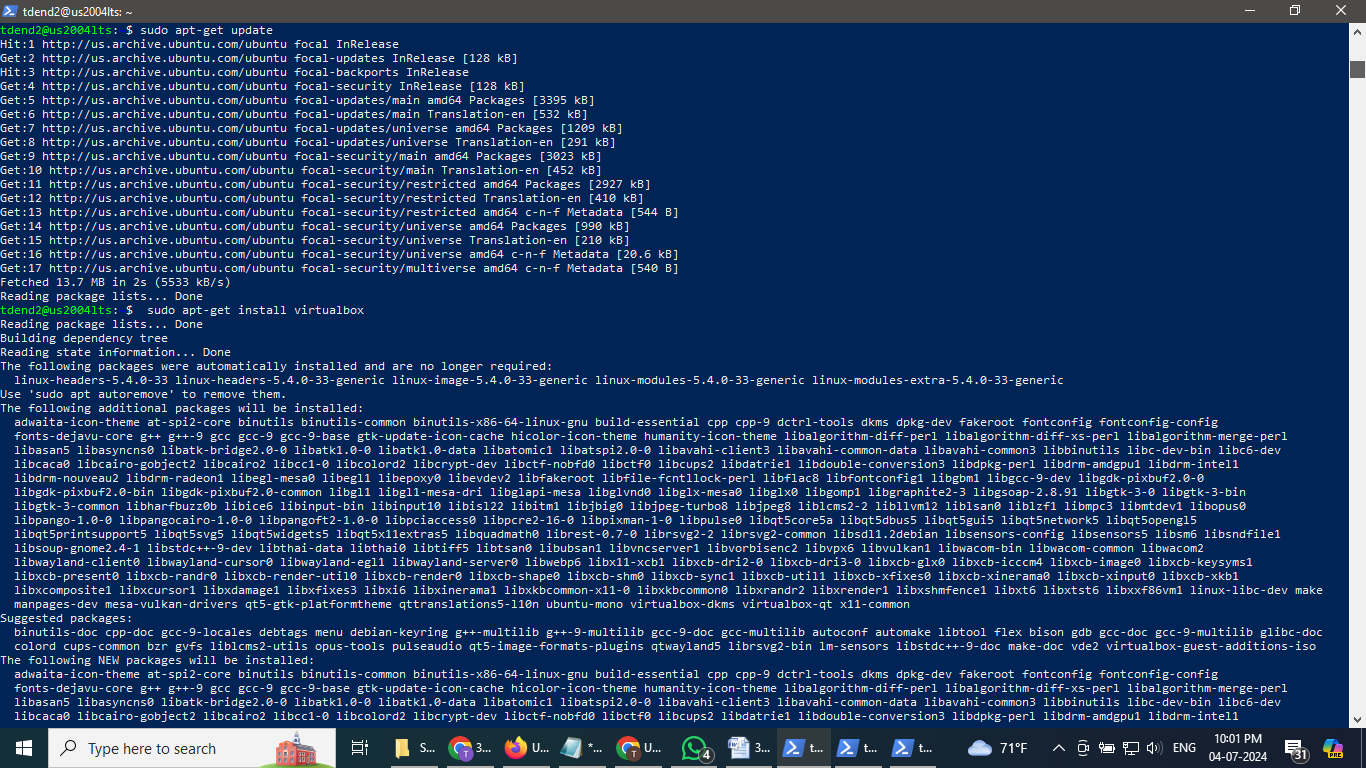


checked versions of Docker, Docker compose, and Docker machine as shown below:





1.3.2 Install VirtualBox



1.3.3 Create virtual nodes with docker-machine

To avoid access-denied errors regarding the host only network, create networks.conf file under /etc/vbox, then type this. “\* 0.0.0.0/0 ::/0” allows any addresses, effectively disabling range control.

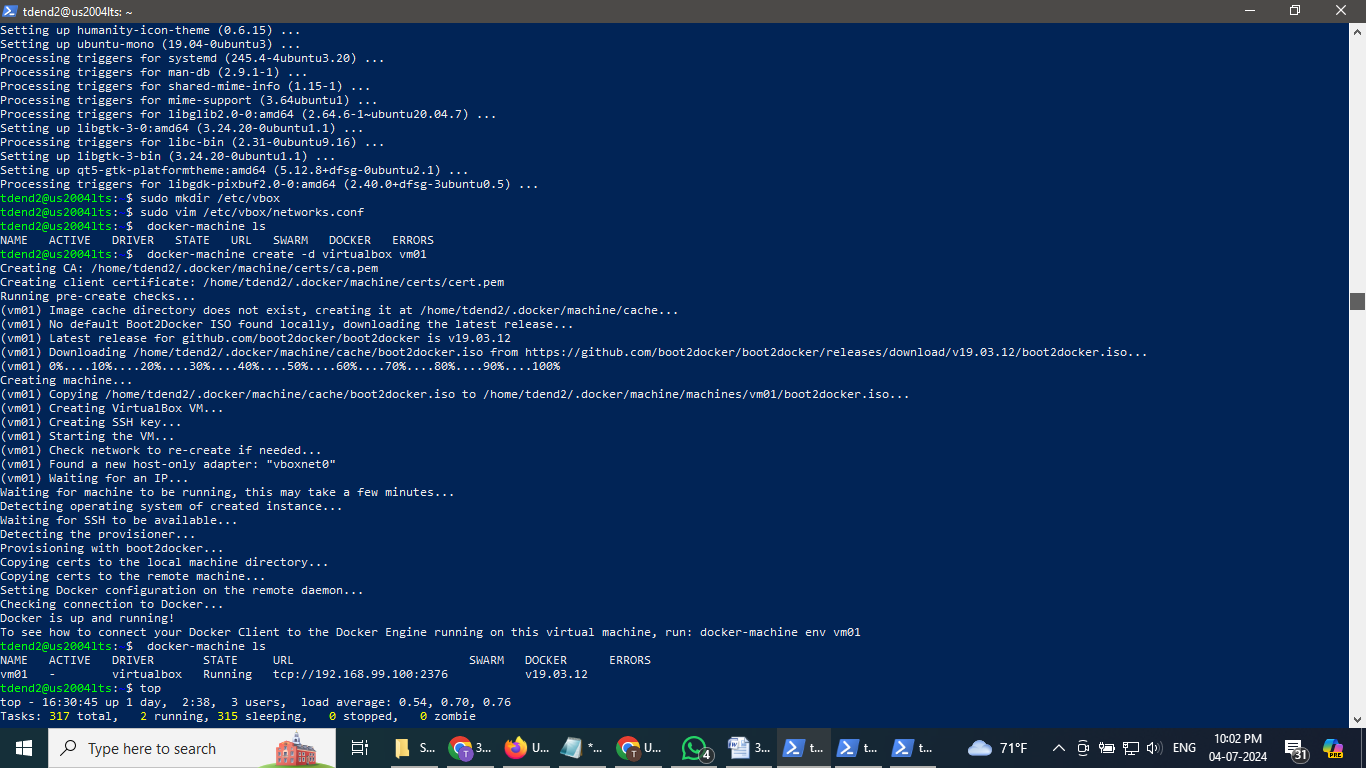
The docker-machine basic command, ls, which lists virtual machines/nodes.

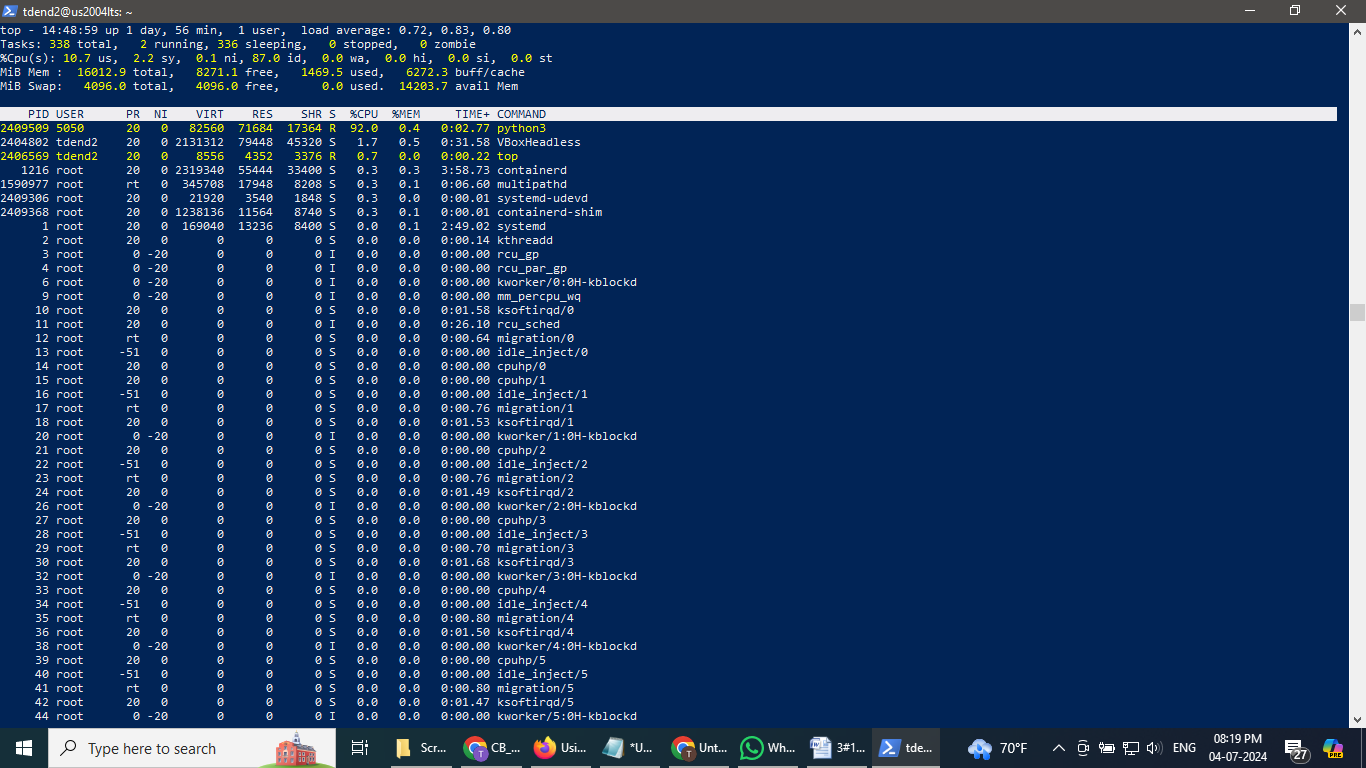
Used the virtual box as a hypervisor (-d virtualbox) and name the node as vm01

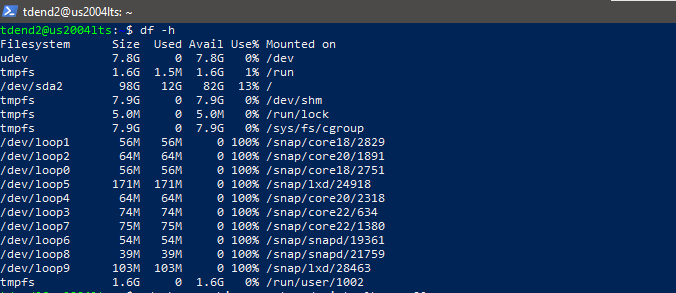
Docker downloaded the Boot2Docker ISO image from GitHub. Boot2Docker is a lightweight Linux distribution made specifically to run Docker containers. It runs completely from RAM, is a ~45MB download, and boots quickly; see https://github.com/boot2docker/boot2docker Note: Docker is up and running inside a virtual node on VirtualBox as shown below. Let’s verify your new virtual node with the command ‘docker-machine ls’. Used IP address of vm01 as obtained and shown here.

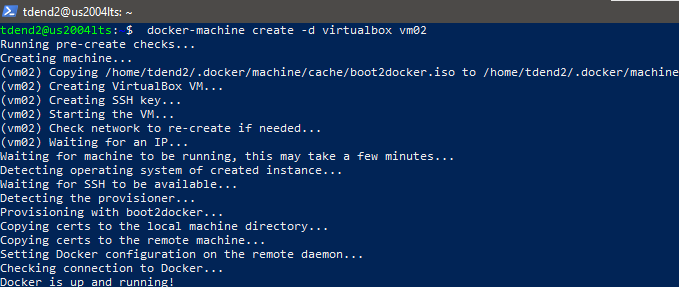
Created another virtual node using docker-machine, vm02

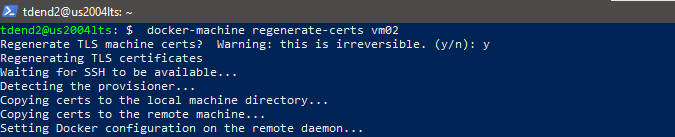
Regenerated TLC machine certs, using the ‘docker-machine regenerate-certs’ command. Specified the virtual node’s name, vm02. You have two virtual nodes, vm01, and vm02

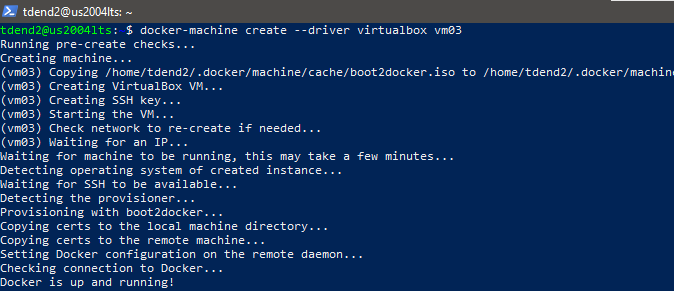




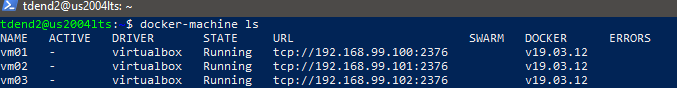




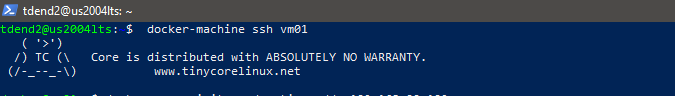




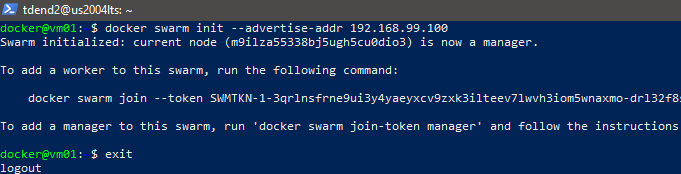
1.4 Setting up a Swarm cluster with Docker Swarm



To configure your Docker Swarm cluster, you will ssh to vm01. Checked if we are in HostVM, a container, or a virtual node.



Executed the ‘docker swarm init’ command. Passed the --advertise-addr argument with the IP address of the vm01



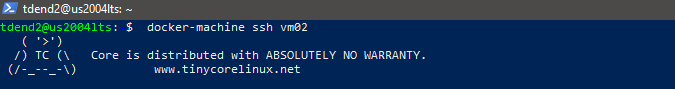
Join token

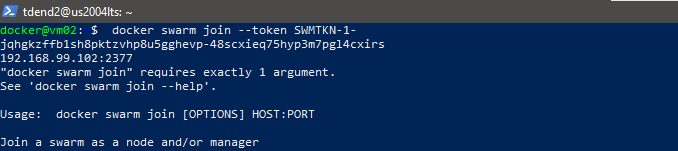
Vm02 is made as woker node as follows:

docker swarm join --token SWMTKN-1-3qrlnsfrne9ui3y4yaeyxcv9zxk3ilteev7lwvh3iom5wnaxmo-drl32f8sboj1th4q4wu2ismhi 192.168.99.100:2377



To make vm3 a worker node, SSHed to the vm3.





Vm2 as worker:

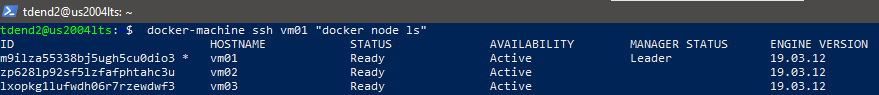


Vm3as worker node:

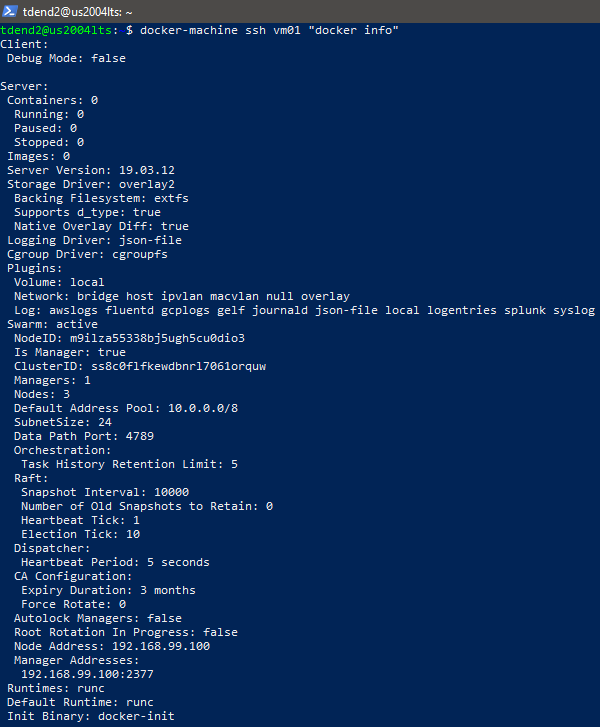


All 3 nodes:

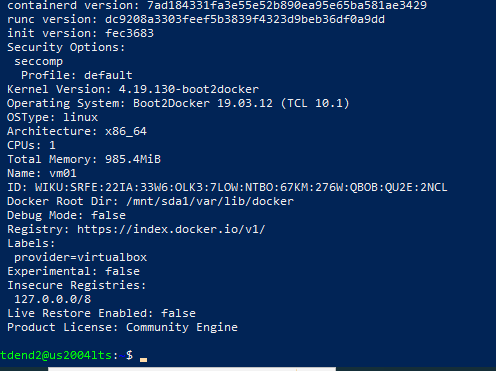
After joining vm02 and vm03 as workers, checked from vm01 if those nodes are available from the manager. Again, passed the command to ssh. We have three nodes, and vm01 is a manager or leader.



Some information about the node inside the VM:



Continued below:



c. Section 1.5 Building a Cassandra Cluster

i. Shown below that three nodes are running using ‘nodetool status’

Note: cassandra01 is a "seed" node for Cassandra cluster, so it must be started before other

worker nodes are started. See https://docs.docker.com/compose/composefile/#depends\_on and https://cassandra.apache.org/doc/latest/faq/#what-are-seeds

1.5.1 Creating a docker-compose file for deploying a Cassandra cluster Login to vm01 and created a directory, casscluster.

Inside the node, vm01, create a docker-compose file. We will have three services: cassandra01, cassandra02, and cassandra03. We want to deploy multi-container apps on multi-node with Swarm so that we will put cassandra01, 02, and 03 on vm01, vm02, and vm03, respectively, see placement: constraints:-node.hostname in the compose file.

The ***docker-compose.yml*** with cassandra image 3.11 file is as follows:

version: "3.7"

services:

# Node01

cassandra01:

image: cassandra:3.11

environment:

CASSANDRA\_LISTEN\_ADDRESS: "tasks.cassandra01"

deploy:

restart\_policy:

condition: on-failure

max\_attempts: 3

window: 120s

placement:

constraints:

- node.hostname == vm01

ports:

- 9042

volumes:

- cass-data:/var/lib/cassandra

networks:

- cass-net

# Node02

cassandra02:

image: cassandra:3.11

environment:

CASSANDRA\_LISTEN\_ADDRESS: "tasks.cassandra02"

CASSANDRA\_SEEDS: "tasks.cassandra01"

depends\_on:

- cassandra01

deploy:

restart\_policy:

condition: on-failure

max\_attempts: 3

window: 120s

placement:

constraints:

- node.hostname == vm02

volumes:

- cass-data:/var/lib/cassandra

networks:

- cass-net

# Node03

cassandra03:

image: cassandra:3.11

environment:

CASSANDRA\_LISTEN\_ADDRESS: "tasks.cassandra03"

CASSANDRA\_SEEDS: "tasks.cassandra01"

depends\_on:

- cassandra01

deploy:

restart\_policy:

condition: on-failure

max\_attempts: 3

window: 120s

placement:

constraints:

- node.hostname == vm03

volumes:

- cass-data:/var/lib/cassandra

networks:

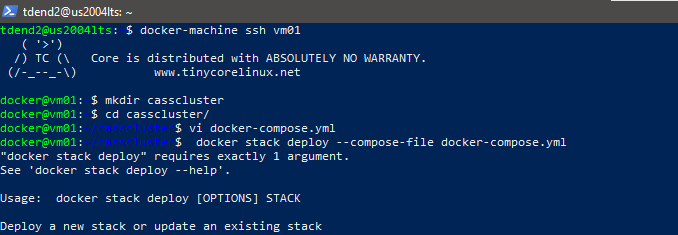
- cass-net

networks:

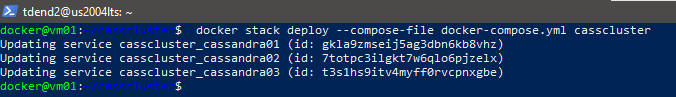
cass-net:

volumes:

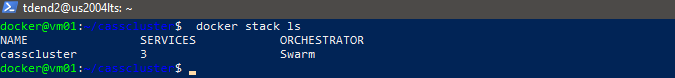
cass-data:



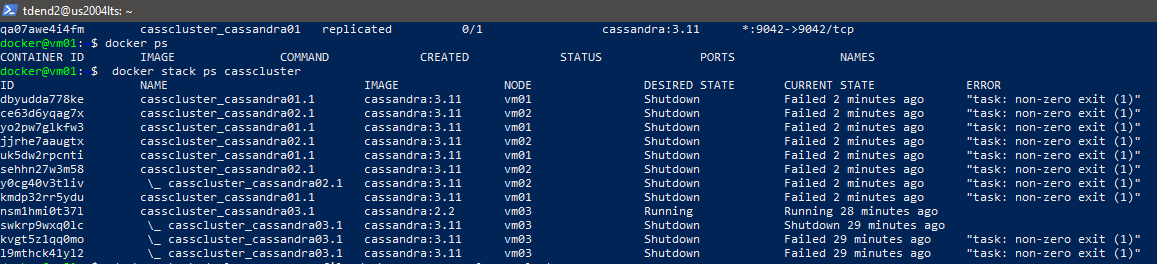
*Deployed the compose file to the Swarm cluster using the Docker stack to swarm*



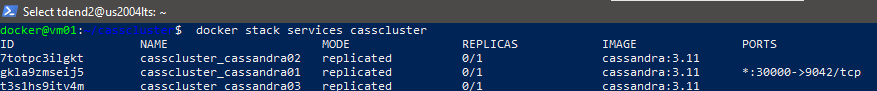
Verified with the ‘docker stack ls’ command.The casscluster with three services orchestrated by Swarm as shown below:



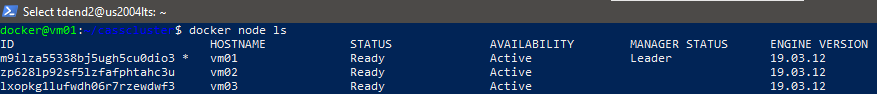
cassandra01, 02, and 03 are running on vm01, 02, and 03 respectively:



List the services in the stack



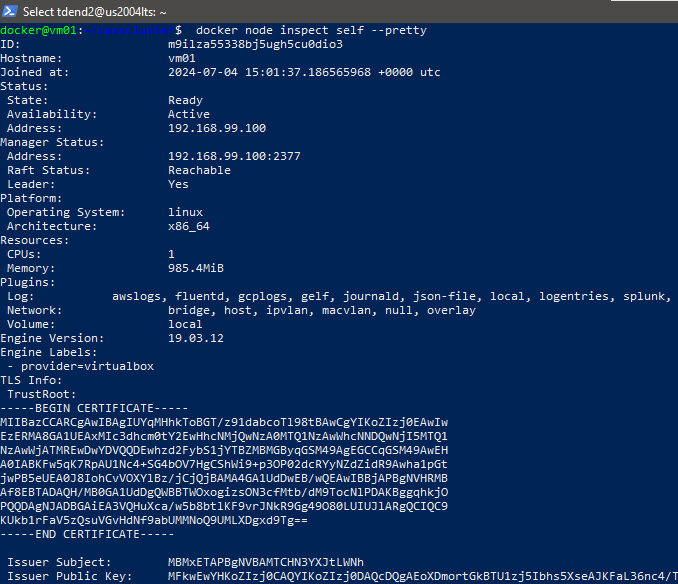
Listing nodes in the swarm



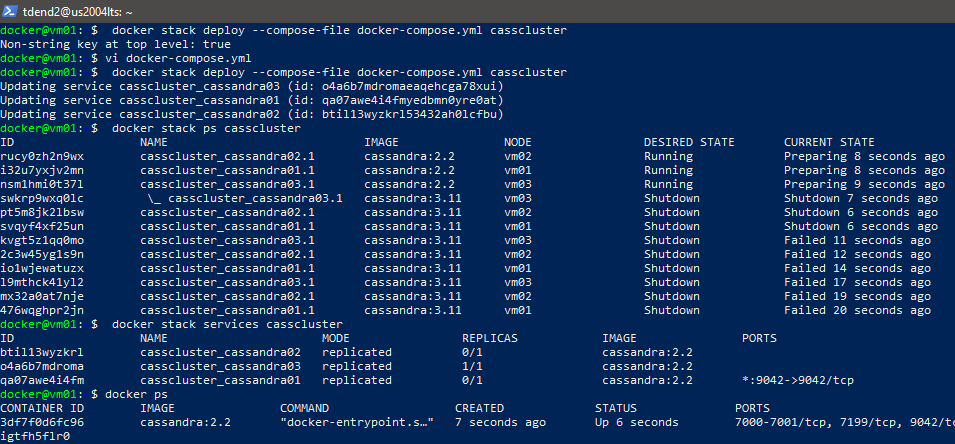
By inferring the current status of services (non-exit error in ***docker stack ps casscluste*r** and replica (0/1) state through ***docker stack services casscluster, docker ps which resulted no running container*** ) of cluster we concluded as follows:

**With cassandra 3.11 in the above docker-compose.yml cluster could not be successfully deployed and tried different versions of cassandra and found that the compose file worked with cassandra 2.2 as shown below.**

Ran docker node inspect on a manager node to view the details for an individual node. The output defaults to JSON format, but you can pass the --pretty flag to print the results in a human-readable format:



When tried with cassandra 2.2 cluster was built successfully:



As shown above when executed using the below, it is found that three nodes are working, and their status is UN which means Up and Normal. Cassandra NoSQL database cluster is working fine.

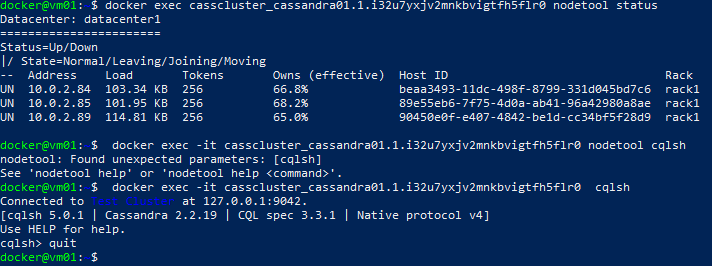
***docker exec casscluster\_cassandra01.1.i32u7yxjv2mnkbvigtfh5flr0 nodetool status***

Checked the cluster by running cqlsh

***docker exec -it casscluster\_cassandra01.1.i32u7yxjv2mnkbvigtfh5flr0 nodetool cqlsh***

***Nameof cluster:***

***casscluster\_cassandra01.1.i32u7yxjv2mnkbvigtfh5flr0***



========================THE END=======================