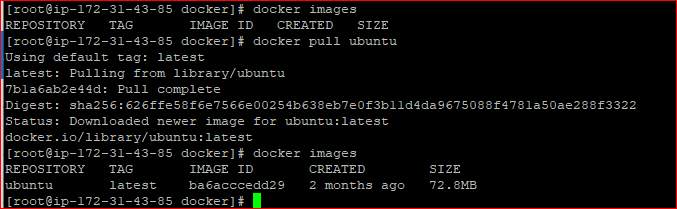
**Module-3: Docker – I Assignment – 1**



docker run -it --name ubuntucont -p 80:80 ubuntu /bin/bash

apt-get update –y ; apt-get install apache2 –y

# configured custome tcp por 80 on ec2 security group inbound.



docker exec -it ubuntucont /bin/bash

service apache2 start



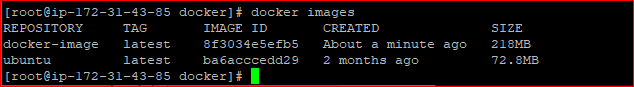
URL: <http://65.2.143.25/> # ec2pulib IP



**Module-3: Docker-l Assignment – 2**







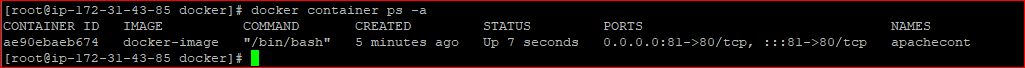
docker run -it --name apachecont -p 81:80 docker-image /bin/bash

service apache2 start

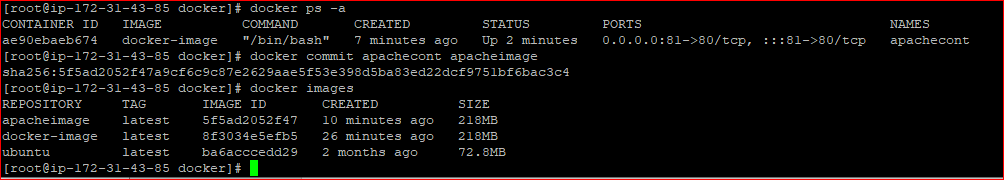
root@ae90ebaeb674:/# service apache2 status

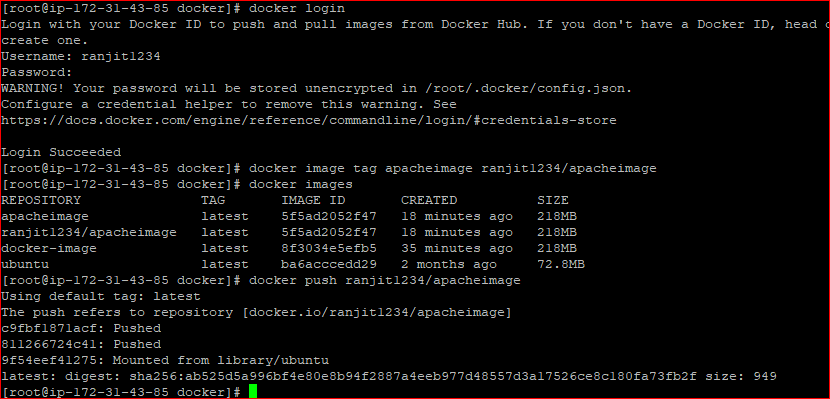
\* apache2 is running



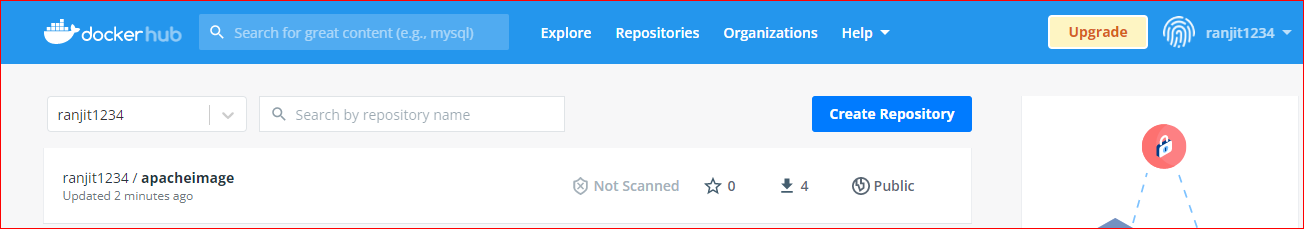


**Module – 3: Docker –l Assignment – 3**

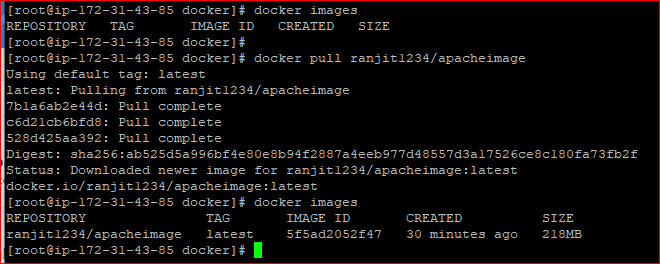




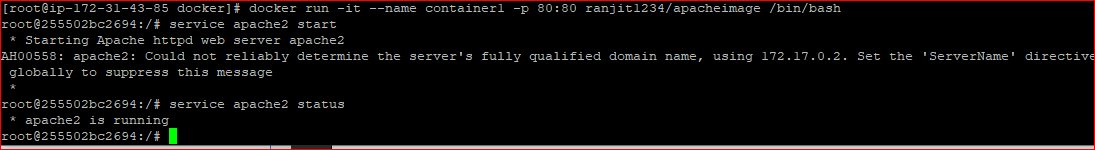
Images pushed on docker hub

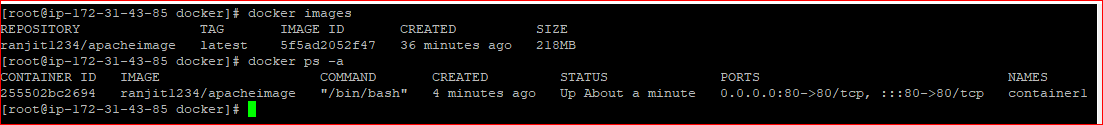


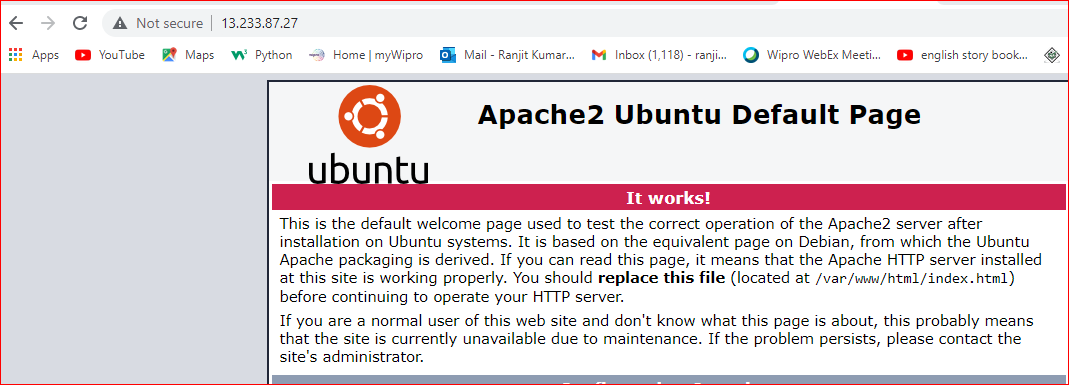
Now puling image on separate machine from docker hub.



Now start container using latest image.







**Module-3: Docker – l Assignment – 4**

Creating a docker file

Vi Dockerfile

FROM ubuntu

RUN apt-get update

RUN apt-get -y install tzdata

RUN apt-get -y install apache2

ADD . /var/www/html

ENTRYPOINT apachectl -D FOREGROUND

ENV name Intellipaat

**Module-3: Docker – l Assignment – 5**

Create a sample html file

Vi index.html

<HEAD>

<HEAD>

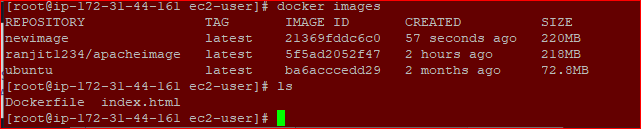
<BODY>

<H1> I am taking training from Intellipat.com </H1>

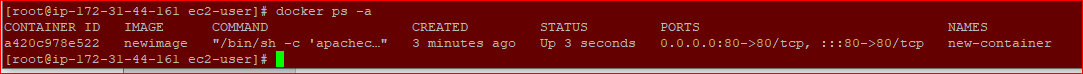
</BODY>

</HTML>

docker build -t newimage .



docker run -itd --name new-container -p 80:80 newimage /bin/bash





**CASE STUDY – CONTAINERIZATION USING DOCKER 1**

You work as a Devops Engineer in a leading Sofware Company. You have been asked to Dockerize the applications on the production server. The company uses custom software, therefor there is no pre-built container which can be used.

Assume the following things:

1. Assume the software to be installed is apache
2. Use an Ubuntu container

The company wants the following things:

1. Push a container to DockerHub with above config
2. The Developers will not be working with Docker, hence from their side you will just get the code. Write a Dockerfile which could put the code in the custom image that you have built.

docker tag newimage ranjit1234/newimage

[root@ip-172-31-44-161 ec2-user]# docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

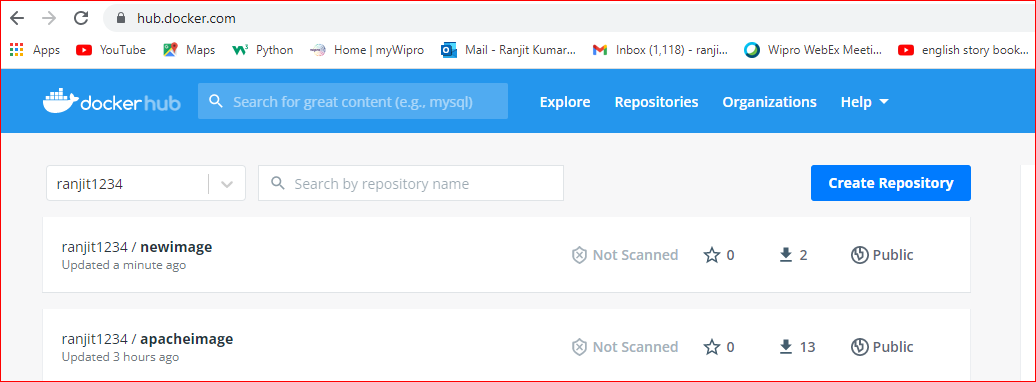
app1 latest b39ec0faea7e 18 minutes ago 220MB

newimage latest b39ec0faea7e 18 minutes ago 220MB

ranjit1234/newimage latest b39ec0faea7e 18 minutes ago 220MB

docker login

docker push ranjit1234/newimage

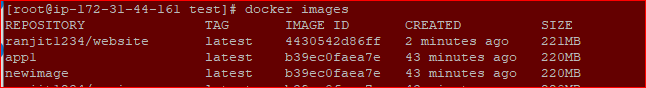


Now download simple website using below link

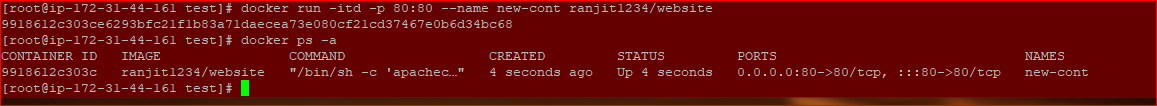
wget https://www.free-css.com/assets/files/free-css-templates/download/page273/progressus.zip

unzip progressus.zip

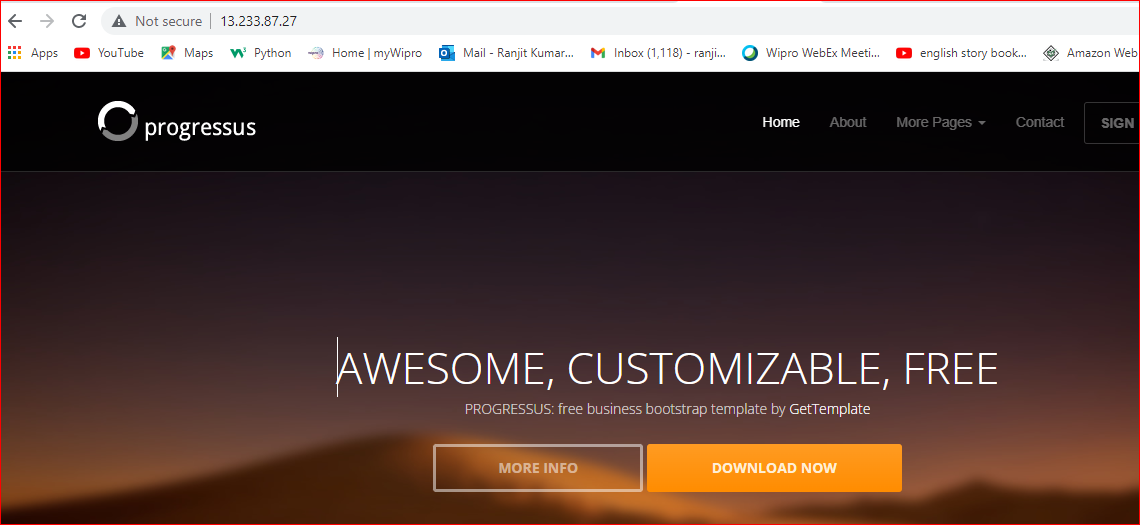
docker build -t ranjit1234/website . # building image



docker push ranjit1234/website # now pushed on docker hub.



Now container running. And below website is running.



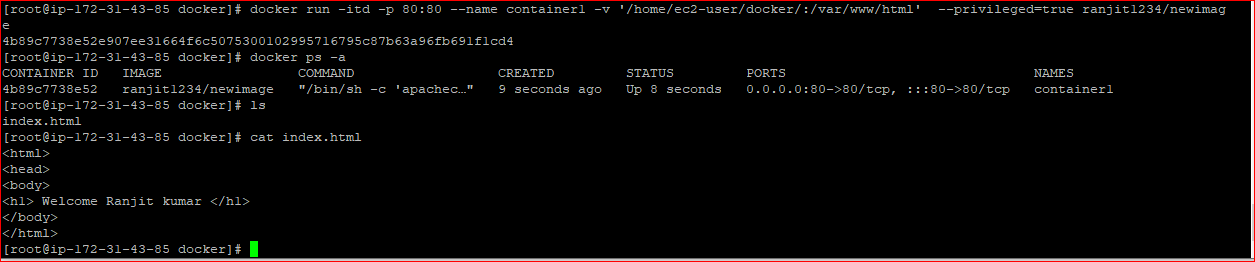
Now goto Route53 and create host zone with created domain name, record set and configure record set. And copy all host zone and configure on freenom as well . Then we will access directly using domain name.

[www.websit.com](http://www.websit.com). # Now I am not able to create free domain.

Module – 4: Docker – ll : Assignment – 1

You have been asked to:

* Launch the Apache2 container created in previous model
* Create a Docker volume on /var/www/html

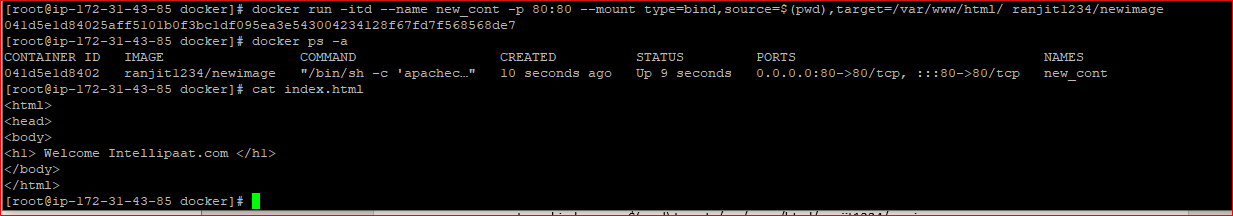
docker run -itd -p 80:80 --name container1 -v '/home/ec2-user/docker/:/var/www/html' --privileged=true ranjit1234/newimage



**Module-4: Docker – ll: Assignment – 2**

* Use the apache2 container created in previous module
* Created a bind mount on /var/www/html to replace html files dynamically

docker]# docker run -itd --name new\_cont -p 80:80 --mount type=bind,source=$(pwd),target=/var/www/html/ ranjit1234/newimage



If I change any content of index.html on local host then it will automatically reflected on container.



**Module-4: Docker – ll: Assignment – 3**

* Create 5 custom container, with 5 different default pages
* Using docker compose, deploy these 5 containers on port 81, 82, 83, 84 and 85 respectively.

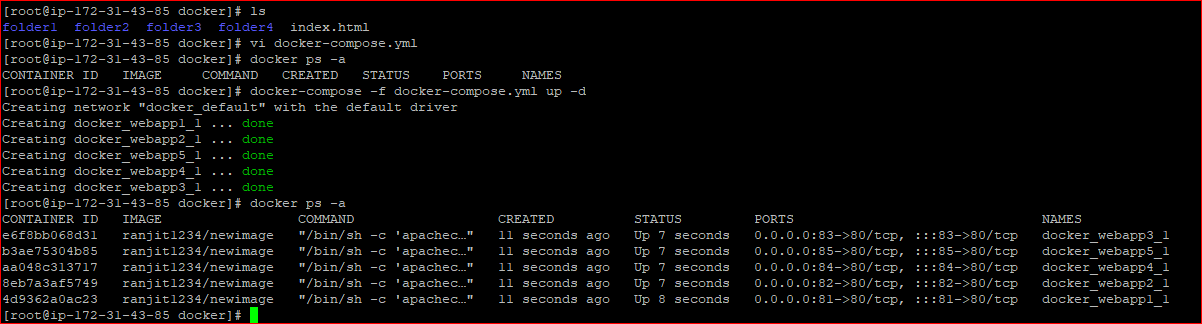
First install docker compose using below command.

sudo curl -L https://github.com/docker/compose/releases/download/1.24.1/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose

$ chmod +x /usr/local/bin/docker-compose

# sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose

Now all port need to configure in ec2 security group inbound rule first.



docker-compose -f docker-compose.yml down

* Every folder has individual index.html is there.

vi docker-compose.yml

version: '3'

services:

webapp1:

image: ranjit1234/newimage

ports:

- "81:80"

volumes:

- /home/ec2-user/docker:/var/www/html/

webapp2:

image: ranjit1234/newimage

ports:

- "82:80"

volumes:

- /home/ec2-user/docker/folder1:/var/www/html/

webapp3:

image: ranjit1234/newimage

ports:

- "83:80"

volumes:

- /home/ec2-user/docker/folder2:/var/www/html/

webapp4:

image: ranjit1234/newimage

ports:

- "84:80"

volumes:

- /home/ec2-user/docker/folder3:/var/www/html/

webapp5:

image: ranjit1234/newimage

ports:

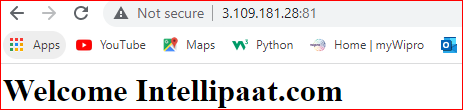
- "85:80"

volumes:

- /home/ec2-user/docker/folder4:/var/www/html/

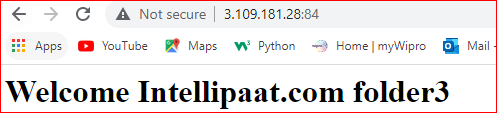
docker-compose -f docker-compose.yml up –d

Now all container working are fine as below.







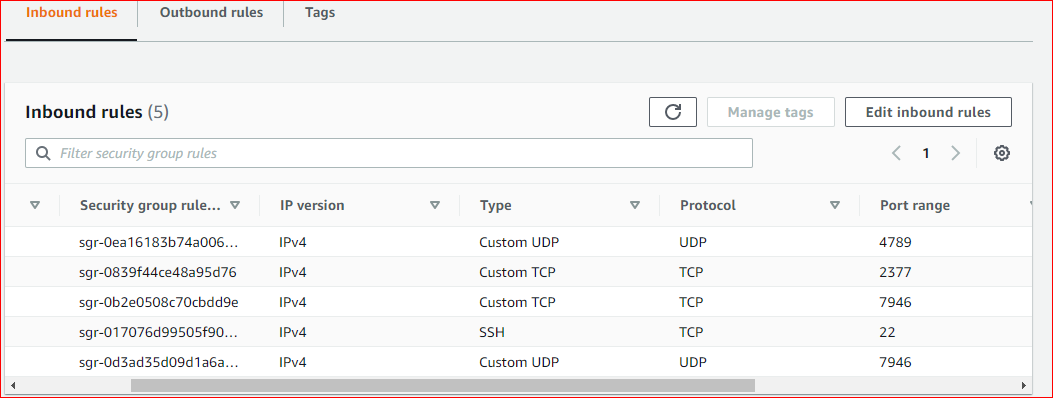


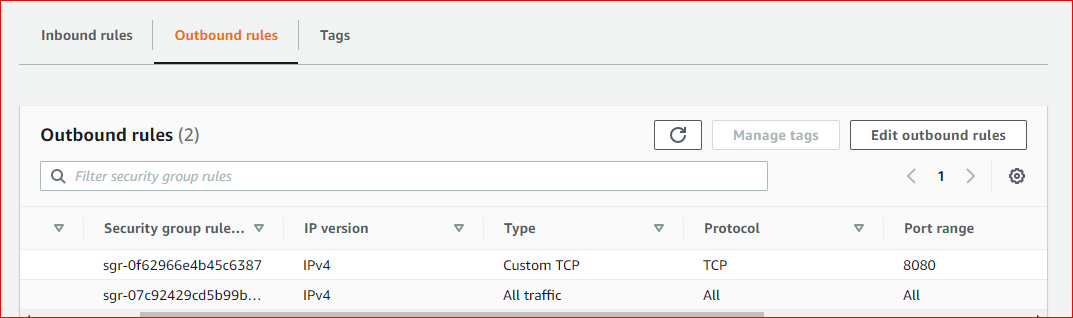


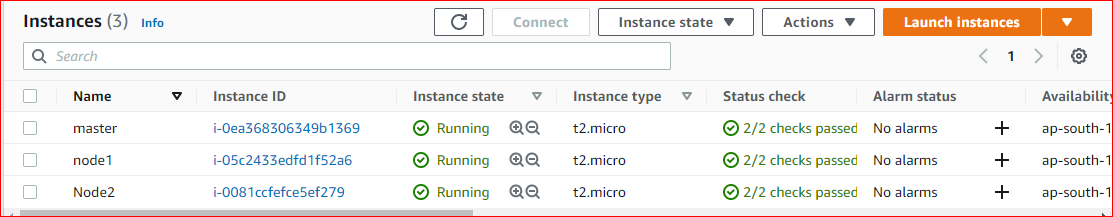
**Module-4: Docker – ll: Assignment – 4**

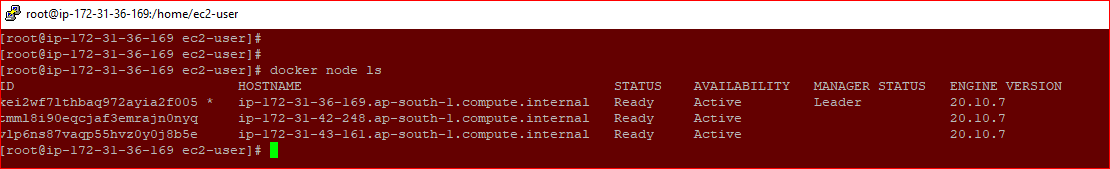
* Create a Docker swarm cluster with 3 nodes
* Deploy an apache container with 4 replicas

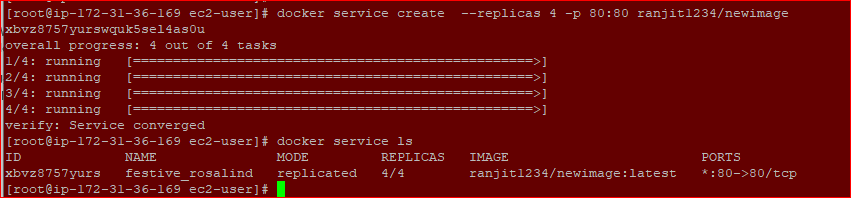
First configured security group inbound rule and outbound rule and attach the security group with all node server. Then Docker swarm cluster will be create.

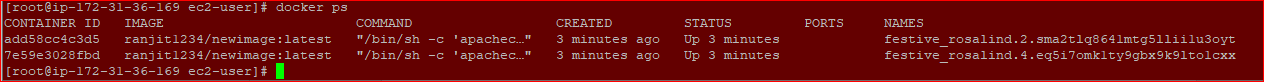


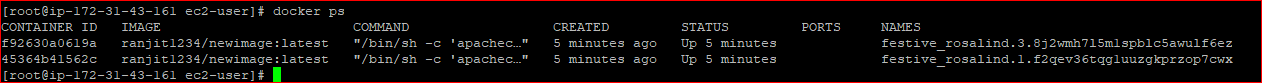












######## MASTER Node machine

docker info # swarm inactive

docker swarm init # clustered and certificate created along with Join token

docker node ls # Now only 1 node cluster available which is master manager status Leader

# now execute "docker swarm join token" command on worker1/2 node first

docker node ls # now show 3 node cluster (master and worker1, worker2)

# docker node ls : this command work only in master node

docker swarm join-token worker # used to retrieve join --token for worker node (incase forgotten token)

docker swarm join-token manager # provide key of manager, when i joined this key in any worker node then it will be join/add as manager node not worker node

# if any cluster worker node is down then follow below command to remove.

docker node rm worker2 # now exists from manager node.

docker node rm -f worker1 # now worker1 node forcely removed from manager/master cluster and with need to execute docker swarm leave on worker1 node

docker node ls # now only show Active master/manager node

docker info|less # now show Down

##### WORKER1 Node machine

docker swarm join --token........ (copy and past) # after executed, This worker1 node add/joined with master swarm manager cluster

docker info|less # swarm Active

docker swarm leave # This worker1 node left the master swarm, (docker node ls execute on master node then status will be Down for worker1)

##### WORKER2 Node machine

docker swarm join --token........ (copy and past) # after executed, This worker2 node add/joined with master swarm manager cluster

docker info|less # swarm Active

docker swarm leave # This worker2 node left the master swarm, (docker node ls execute on master node then status will be Down for worker2)

docker info|less # swarm InActive

On master server

docker service create --replicas 4 -p 80:80 ranjit1234/newimage # in this image apache2 is installed on Ubuntu already

docker ps –a # it listing 4 replica of apache-container as above screen.

URL: any node pulic IP:80 # apache working working fine.

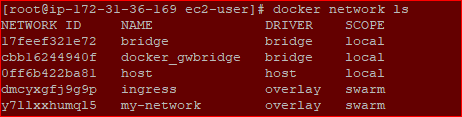


**Module-4: Docker – ll: Assignment – 5**

* Use the previous assignment’s deployment
* Deploy any 2 containers in a overlay network
* Try pinging each of the containers from within the containers.

First create overlay network as below command.

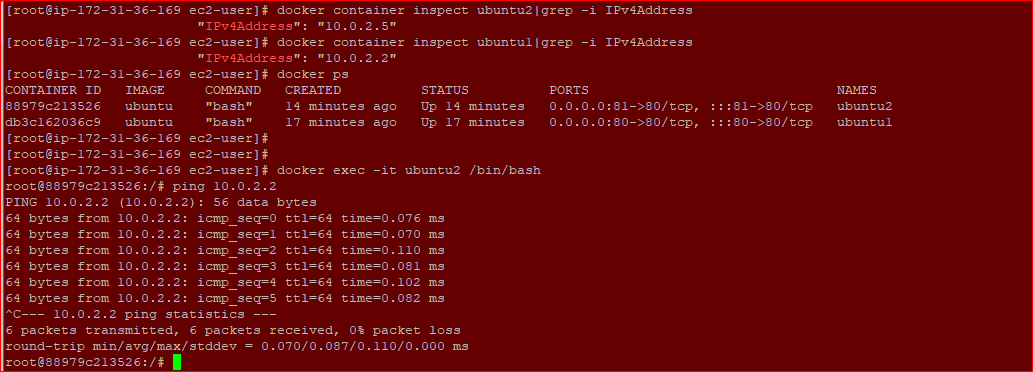
docker network create --driver overlay my-network –attachable



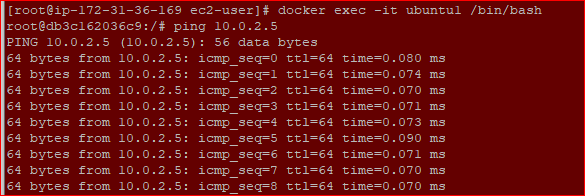
docker run -itd --name ubuntu1 -p 81:80 --network my-network ubuntu

docker run -itd --name ubuntu2 -p 81:80 --network my-network ubuntu

Please check below container IP and also pinging to each container.

Docker exec –it ubuntu2 /bin/bash

ping 10.0.2.5



CASE STUDY – CONTAINERIZATION USING DOCKER ll

Problem Statements:

You have been hired as a Devops Engineer in GrapeVine Pvt. Ltd. You have been asked to improve the way the company is managing their Docker containers.

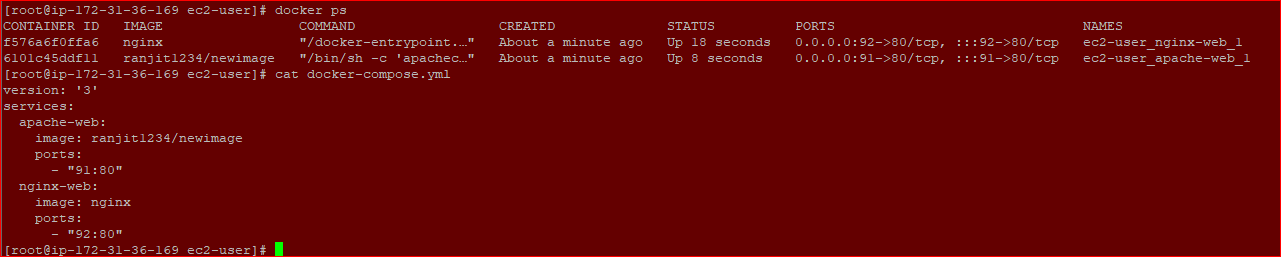
Following task have been assigned:

1. Deploy a sample HTML website in any apache container, and demonstrate how we can dynamically change content in the container by making changes on the host machine (Bind Mounts)

Ans: This answer is already there on 2nd assignment as same.

1. Deploy apache and nginx containers using Docker Compose, Apache should be exposed on Port 91 and nginx on port 92

Ans:





version: '3'

services:

apache-web:

image: ranjit1234/newimage

ports:

- "91:80"

nginx-web:

image: nginx

ports:

- "92:80"

1. . Initialize a Docker Swarm Cluster, and deploy two Ubuntu containers in a overlay network. Demonstrate they can communicate with each other, by pinging them

Ans: This answer is there is assignment 5.