Story title: Build New Docker Swarm in Dev

Sprint Session: 46

Installed By: Martin Tapala

Team: DevOPs

Date: 09-24-21

Updated Date: 20-28-21

Updated By:

$$$$$$$$ Docker Installation as shown below: $$$$$$$$

Install using the repository

#######################################################

Before you install Docker Engine for the first time on a new host machine, you need

to set up the Docker repository. Afterward, you can install and update Docker from the repository.

1.Set up the repository

#######################################################

Update the apt package index and install packages to allow apt to use a repository over HTTPS: On Ubuntu

sudo apt-get update

sudo apt-get install

ca-certificates

curl

gnupg

lsb-release

Add Docker’s official GPG key as shown below:

#######################################################

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

B.Use the following command to set up the stable repository. To add the nightly or test repository, add the word nightly or test (or both)

after the word stable in the commands below. Learn about nightly and test channels.

echo

"deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu

$(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

Install Docker Engine

###########################################################

Update the apt package index, and install the latest version of Docker Engine and

containerd, or go to the next step to install a specific version:

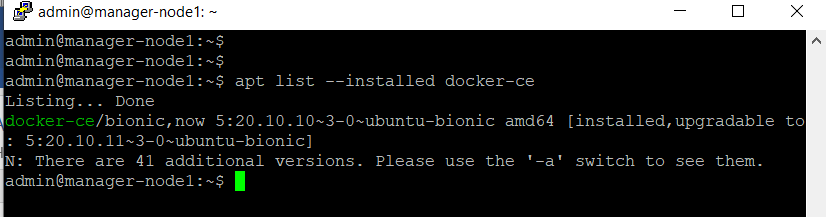
sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io

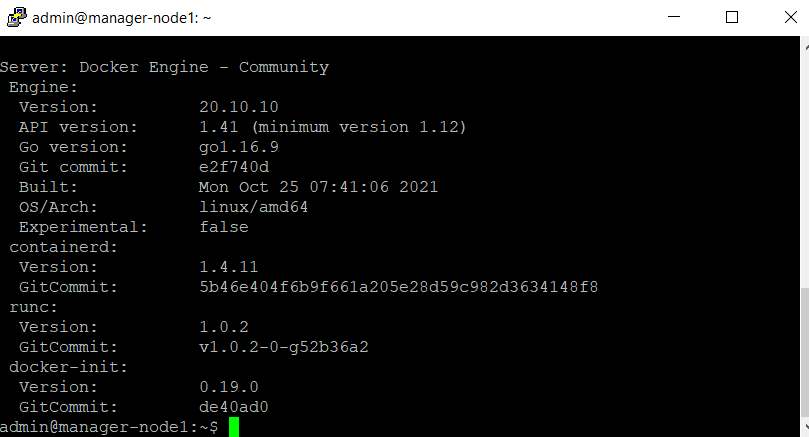
TEST DOCKER INSTALLATION & VERSION as shown below:

To Verify if docker has been installed as shown below:

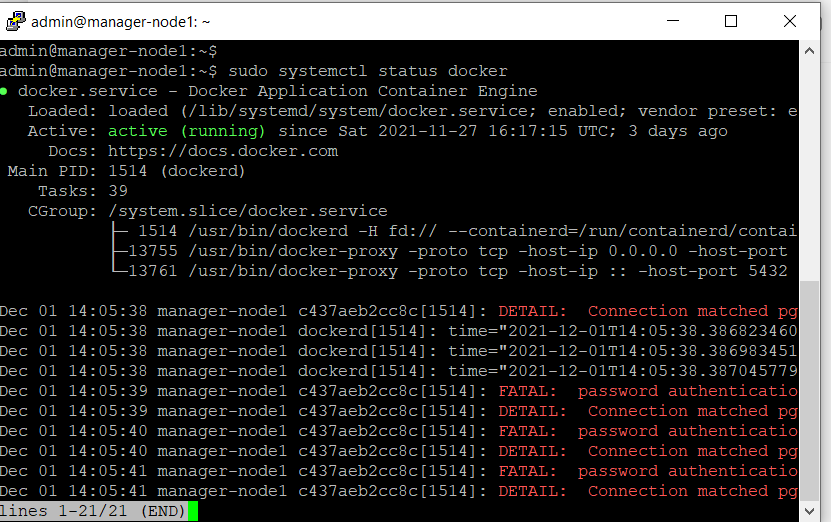
apt list --installed docker-ce



$ docker version



sudo systemctl status docker



sudo systemctl start docker

docker --version ( For Rhel or Centos and Debian)

See docs.docker.com link for Docker installation in Ubuntu

Link:(https://docs.docker.com/engine/install/ubuntu/)

###################################################################

To Open ports see below commands:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

in ubuntu $ echo DOCKER\_OPTS="-H=tcp://0.0.0.0:4243 >> /etc/default/docker

OR

$ /usr/bin/docker -H tcp://0.0.0.0:4243 -d &

Or

$ /usr/bin/docker -d -H tcp://0.0.0.0:2375 -H unix://var/run/docker.sock

Creating Containers

6.#################################################################################

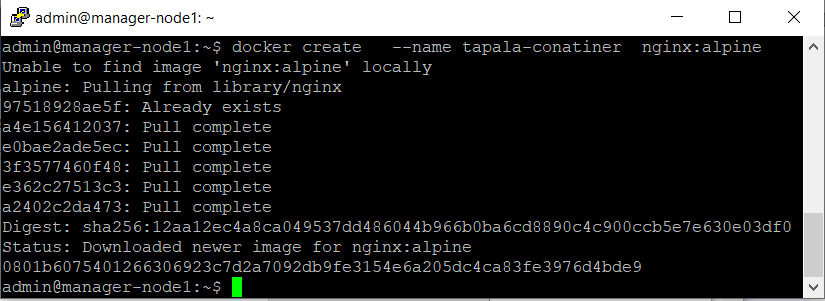
$ docker container --help

# d

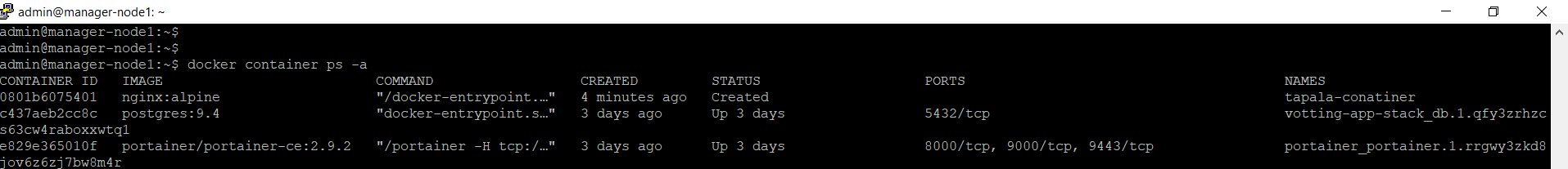
ocker run -d nginx

OR

docker container create --name tapala-conatiner nginx:alpine



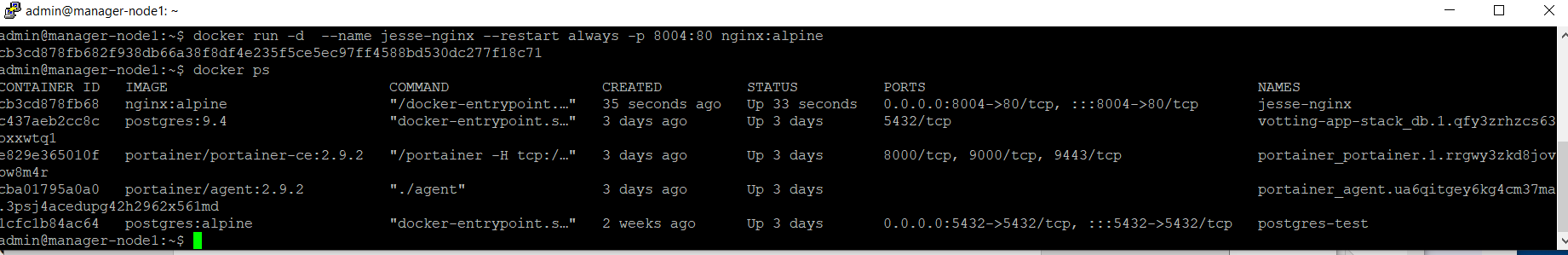
$docker container ls OR docker container ps -a



To Configure Restart Policy as shown below:

#############################################################################

docker run -d --name jesse-nginx --restart always -p 8004:80 nginx:alpine

-

7B. DOCKER-COMPOSE INSTALLATION

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

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

1.Run this command to download the current stable release of Docker Compose:

$ sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

Apply executable permissions to the binary:

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

FYI: Note: If the command docker-compose fails after installation, check your path. You can also create a symbolic link to /usr/bin or any other directory in your path.

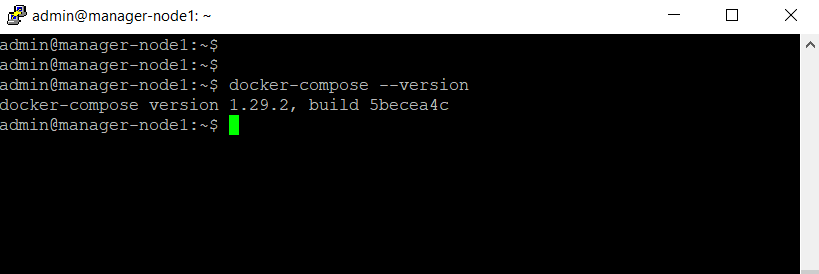
For example:

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

Test the installation.

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

$ docker-compose –version



docker-compose version 1.29.2, build 1110ad01

FYI;

\*\*\* Docker-compose References doc notes: https://docs.docker.com/compose/install/

DOCKER-SWARM INSTALLATION

8.###############################################################################

Manager-Node: docker

docker swarm init --advertise-addr

# docker swarm init --advertise-addr 192.168.1.103

Always put static iP address on Manager-Node

Open the Following Ports:

Port 2377 for TCP

Port 7946 for Host communication

Port 4789 for UDP Network

In order to choose Worker-Node execute the following cmd as shown below:

9.#######################################################################################

First test if docker swarm has been install (If not it will show inactive) and if installed it will show "swarm: Active")

Use below commands to test if docker swarm has been ACTIVATED:

#

docker system info | grep swarm

Sample will be as: "swarm: inactive (Then activate it by running the below init command)

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

#docker swarm init

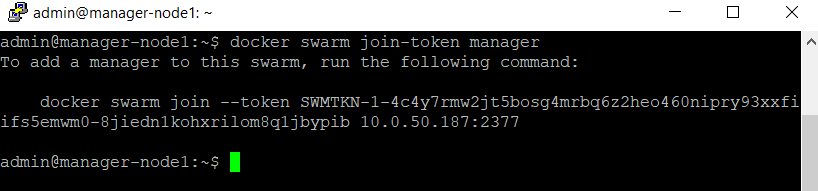
OR

$ docker swarm init --advertise-addr 192.168.1.103

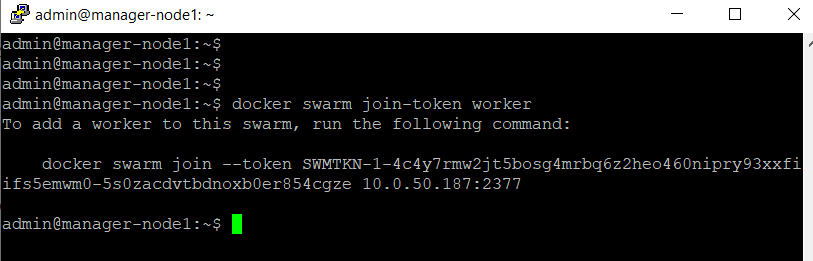
OR

After swarm has been initialized and you need to add more manager-nodes or worker-nodes to swarm use below command:

$docker swarm join-token manager



$ docker swarm join-token worker

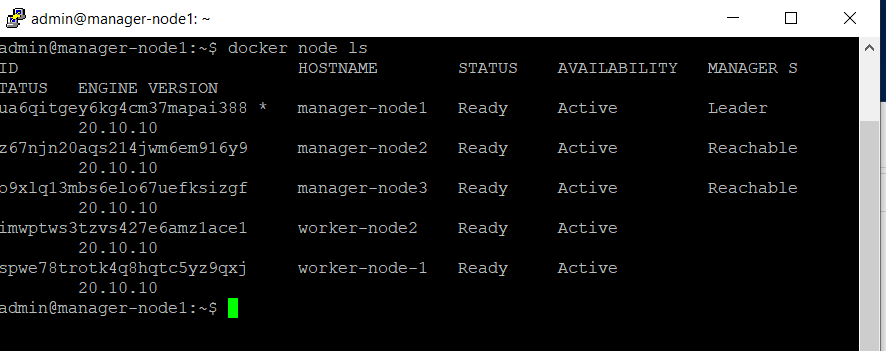


"Add worker to this swarm" copy the token and paste on worker node (VM) etc (As many worker as you need per your requirements etc

To list the nodes in swarm cluster run the below commands and \* (asterick) indicate Manager-node or Host-NAME.

10.#################################################################################################################################

#docker node ls



### After few months In order to add nodes to docker-sawrm run below cmd on MANAGER-Node if you forgot first init cmd :

#########################################################################################################################

docker swarm join-token worker ( See same commands as shown above)

docker swarm join-token manager ( You will see a token, then Copy & paste the TOKEN below on node you want to join the manager)

NB: In case you forgot the token, to copy and paste on worker-node in order to join the swarm after some months etc

just execute the following commands as shown below and the Manager will provide the token again:

##########################################################################################

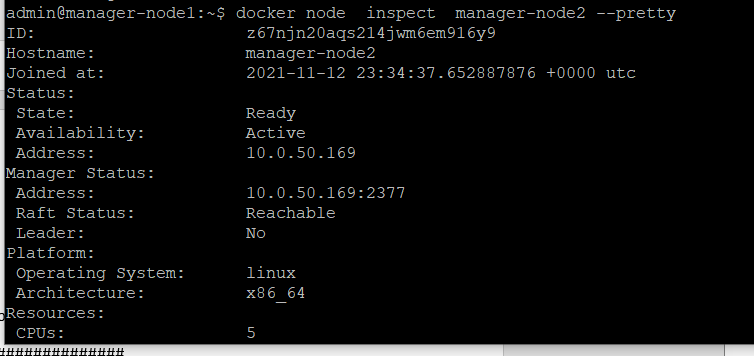
#docker swarm join-token worker [ See same command and screenshot as shown above]

"Copy token here and paste on worker-Nodes "

To show full manager-Nodes Status use the below commands:

#################################################################

docker node inspect manager-node-name –pretty



The above commands will show the following status as shown on screenshot above:

#########################################################

Swarm: active

Pause: After Paused manager can't schedule any tasks to worker-nodes but the existing tasks will STILL be RUNNING

Drain: No tasks will be assigned to worker-nodes and the existing tasks will STOP running as well after DRAINED

DRIANED is used for maintenance and after the worker may re-join the cluster

To Promote Worker to Manager-Node use the below commands:

#################################################################################

First List the Existing Nodes in the Cluster:

docker node ls

Second: Promote Worker to Manager as shown below:

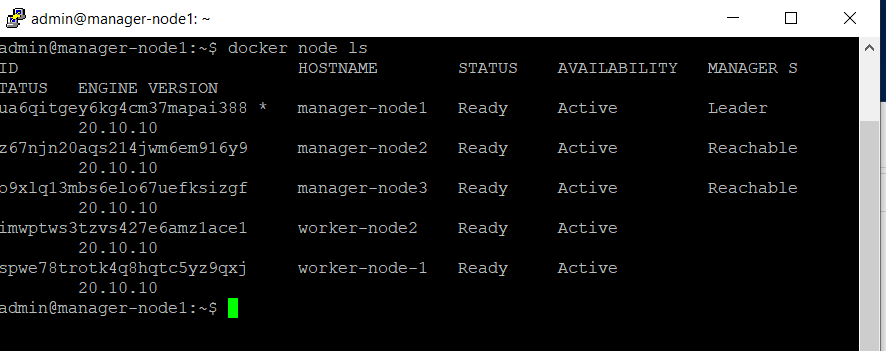
# docker node promote Worker-Node-Name or Worker-Node-ID

List the Nodes again to verify if worker

has been Promoted to MANAGER and Check for "MANAGER STATUS"

LOOK FOR MANAGER STATUS and THE Promoted worker

will shows as "Reachable" Under MANAGER Status (ONLY One \* Leader (Manager) at a time)



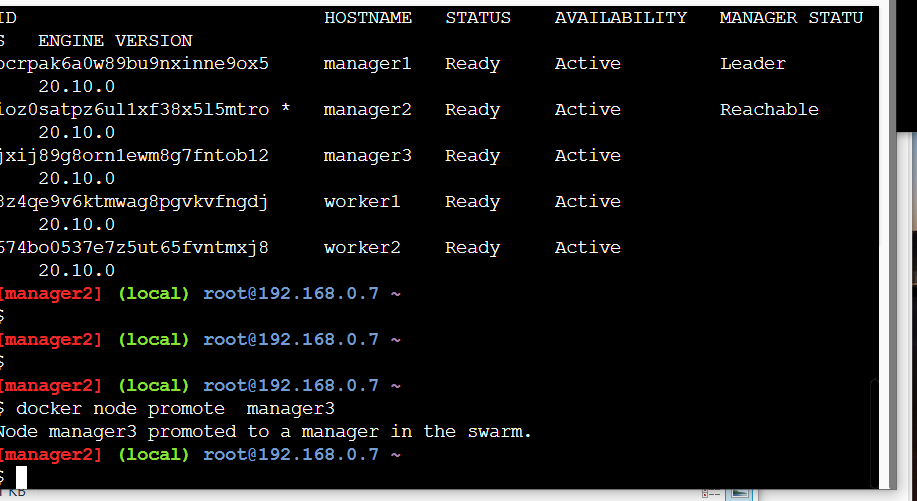
docker node inspect worker [TO SEE IN JSON format | See same screenshot above)

OR

#docker node inspect --pretty [Not in Json Format & more simple with details ]

To DEMOTE/REMOVE Worker from MANAGER Back to Worker Node:

docker node promote worker



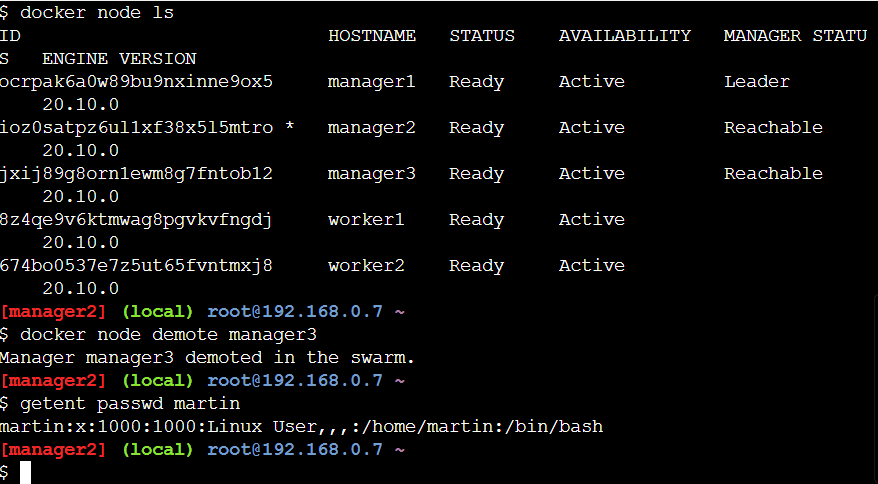
OR

docker node demote worker-NAME or ID

To Verify by using below command:

docker node ls

Notes: Check the MANAGER STATUS if worker has been demoted or promoted

#### Drain docker node #### 

See all the 5 Steps to take off a node from Docker-Swarm

###################################################################

1st : Drain the node

2nd : Node leave the swarm (FYI, (This must be run only on the node that

you want to leave swarm and NOT in the MANAGER-Node)

3rd : Remove the node from swarm

4th : Start maintenance

5th : Bring back the drained node to swarm by

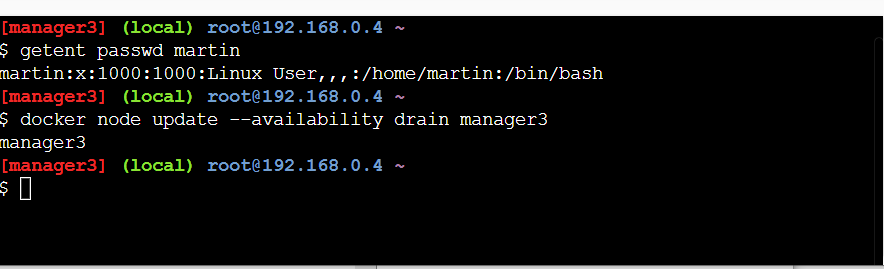
docker swarm join-token worker

Copy the token & paste on worker-Node

docker node update --availability drain worker-name

OR

docker node update --availability drain worker-node-NAME OR ID



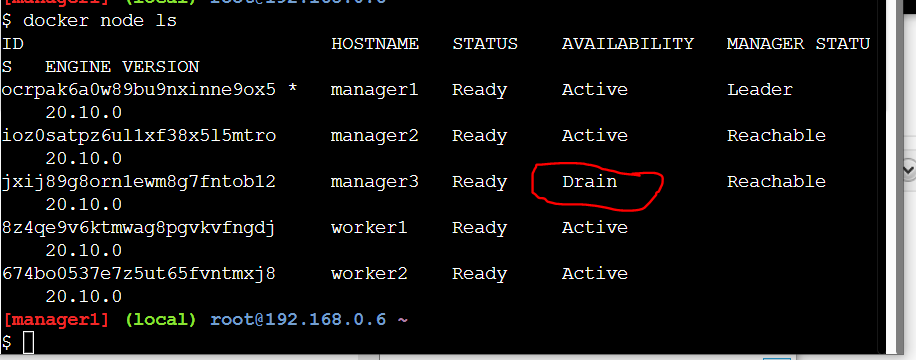
### Verify/TEST ####

#################################################

docker node ls [You will See Availability STATUS as "Drain" (USE for maintenance etc)

After Drained all existing work load will Automatically be assign to the next nodes etc

And MANAGER node will not schedule any workload to the drained node.



Remove the nodes or node Leave the swarm with the cmd below:

####################################################################################

docker swarm leave OR docker swarm leave node-name/ID [Run this on the node itself that you want to leave swarm)

###### Removed the node from the Cluster for maintenance as shown below ####Execute on MANAGER-Node

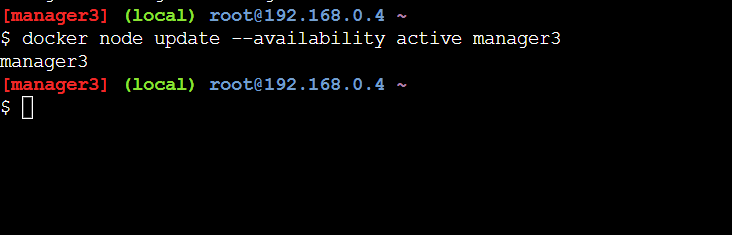
docker node rm worker-node-name OR Node ID

# docker node ls

18-A To Bring Back the node ACTIVE use below command

#######################################################################

docker node update --availability active worker-node-NAME OR ID

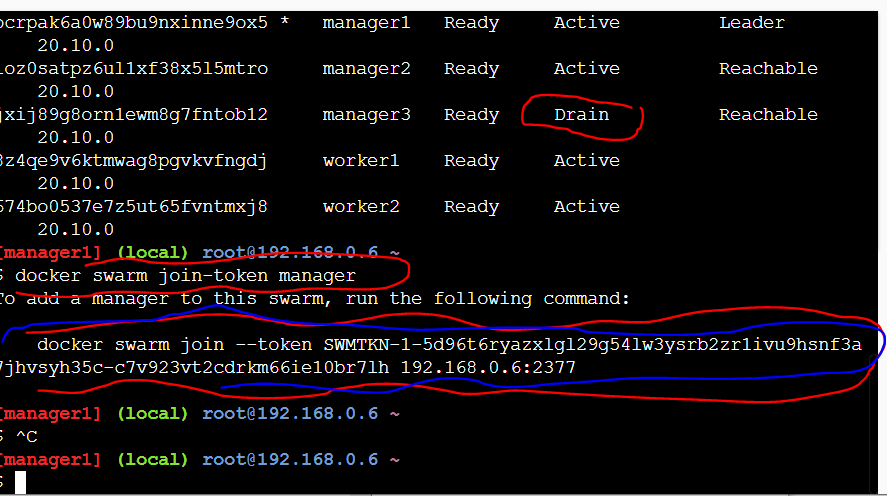


18-B To bring back the node to join the swarm use join token commands as shown below:

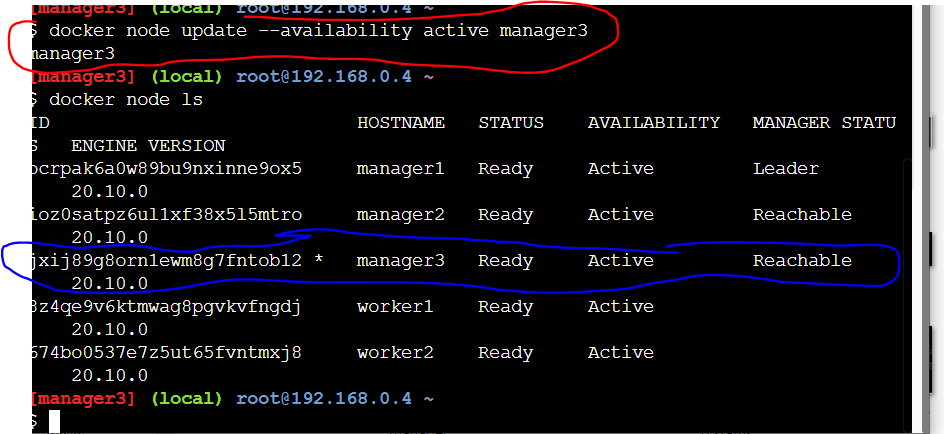
############################################################################################

docker swarm join-token worker

Copy the token command below and paste on the active node to join the docker swarm



docker node ls ( To verify all nodes status)



DEPLOYMENT / ORCHISTRATION AND MANAGER ELECTIONS WITHIN THE CLUSTER OR DOCKER-SWARM

#################################################################################################

$$$$$$$ docker API =

Orchistrator=

Allocator (Assign routing, DNS & ip adrr etc)=

Dispatcher (Assign tasks to nodes on swarm)

Scheduler (Instruct worker to run tasks)

#### VERIFY ######

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

docker service ps websvc

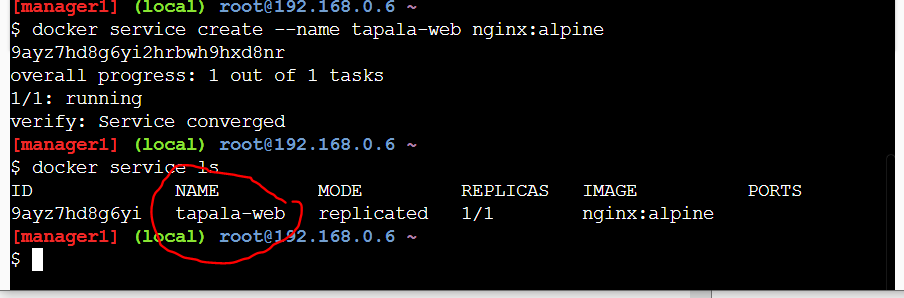
UPDATES

Listing nodes

DOCKER NODE LS

$docker service ps websvc

$docker service create –name svc-name image:version



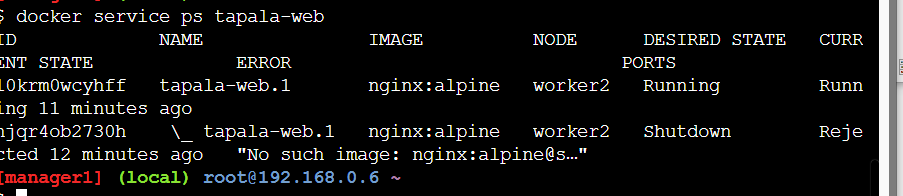
docker service logs websvc OR service-NAME

In Oder to Remove or Delete a Service in docker-swarm see cmd below:

#########################################################################################

docker service rm websvc

docker service ls [ Test to make sure the Service/SVC has been deleted ]



###########################################################################

###########################################################################

CREATING AND REPLCATING CONTAINERS IN DOCKER SWARM

###########################################################

docker service create --replicas=5 websvc

OR

docker service create --name websvc --replicas=3 httpd

docker service ls

docker service create ---name websvc --replica=3 httpd

To Create a Service with Port exposure as shown below:

#################################################################################

docker service create --name=firstservice -p 80:80 --replicas=3 httpd:alpine

NOTE: FYI; the Alpine above is the httpd version. If you need to install current version, use latest

docker service ls

#docker service ps

docker service inspect firstservice OR service-Nname

OR

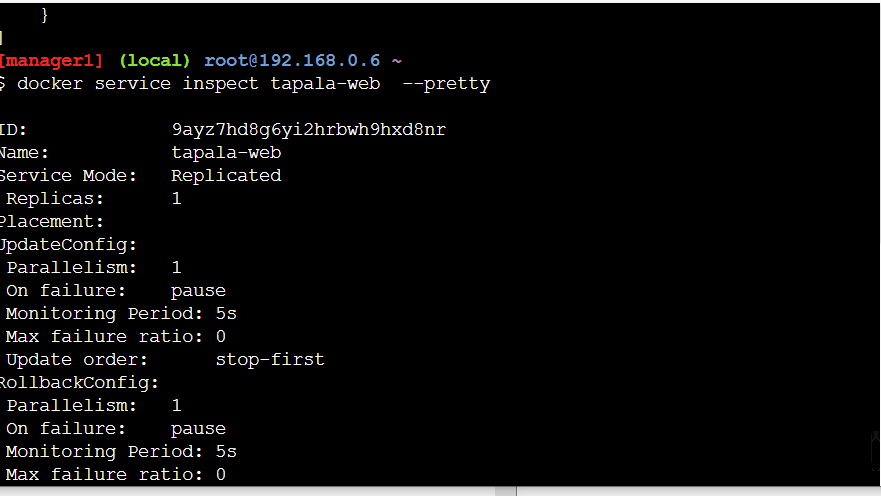
24-B To see it in a simple and pretty format use pretty cmd as shown below:

###################################################################################

docker service inspect service-name --pretty

OR

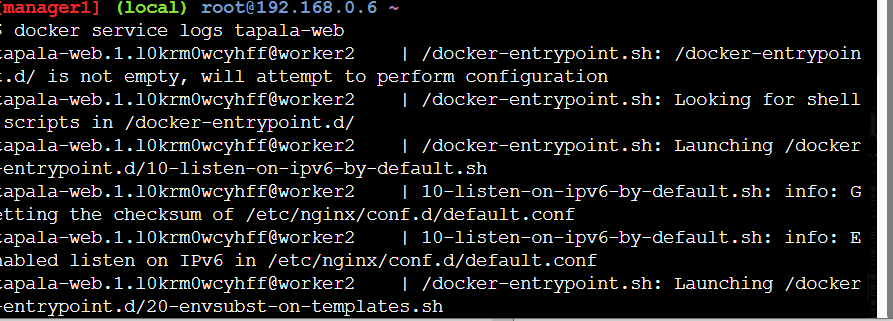
docker service inspect firstservice –pretty



Viewing logs

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

docker service logs service-name



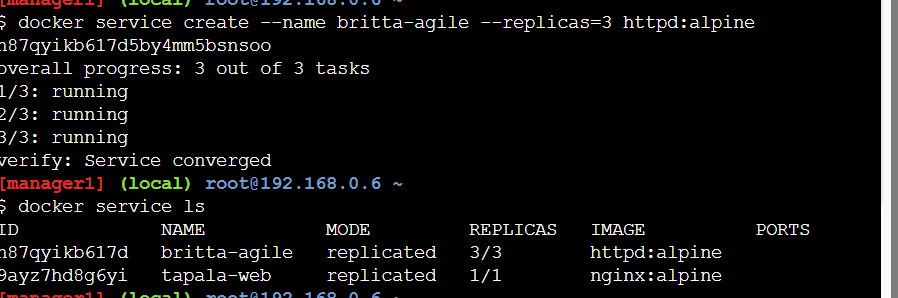
To Increase or reduce the number of Replicas use cmd below:

( We have to replicate manually since there is no AUTOSCALER in Docker-swarm unlike K8S)

###########################################################################################################

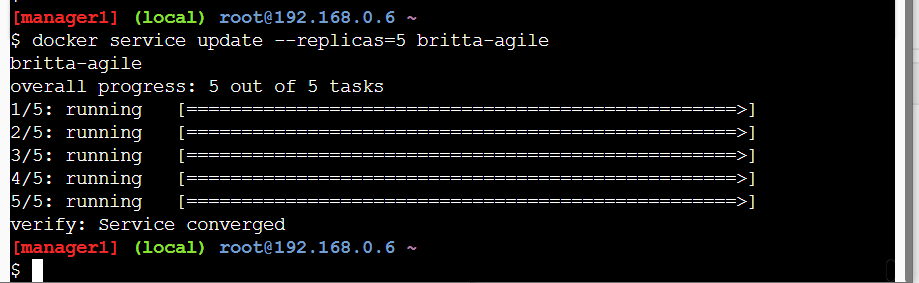
To Scale up/Down numbers of Replicas from 3 to 5 use the cmd below:

docker service ls websvc-name



docker service ps

docker service update --replicas=5 websvc-name e.g britta-agile



docker node ls

Rolling UPDATE

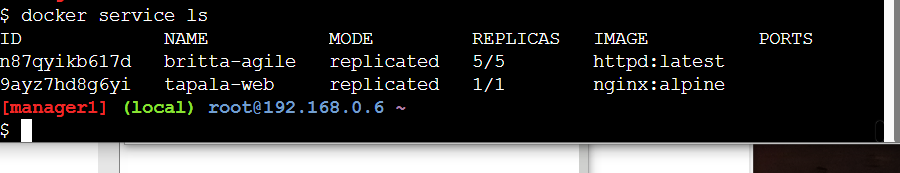
############################################################

docker service update --update-failure-action pause |rollback |continue| --image=httpd:alpine service name-here

$docker service update --update-failure-action rollback --image=httpd:alpine service-name-here

$docker service update –image=httpd:latest svc-name

docker service ls



docker service ps websvc or service-NAME

docker service update

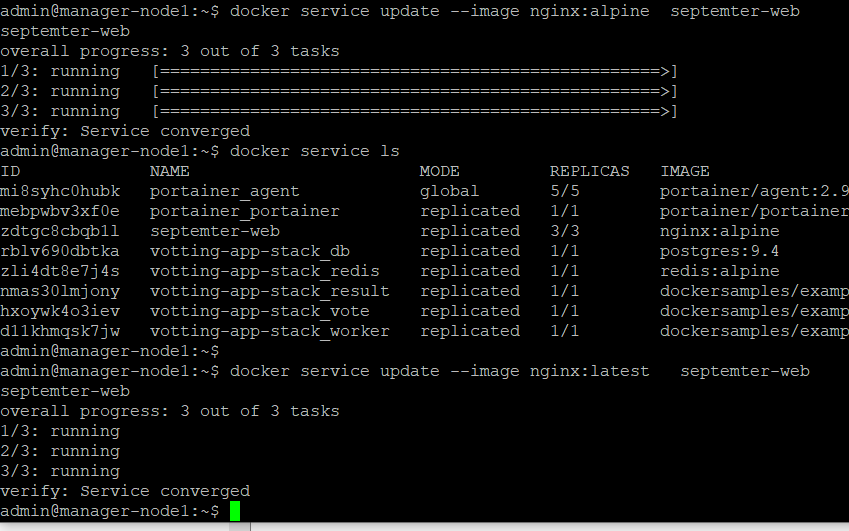
ROLLING UPDATES IN DOCKER SWARM

###########################################################################

$ docker service update --image=httpd:latest websvc or Service-Name

OR

$ docker service update --image nginx:alpine september-web



Verify the updates

docker service ls

28-A Incase of any updates Failure

Rollback Or replace any Update Option as shown below cmd:

Rolling Updates Options:

##########################################

Pause,

Rollback

Continue

delay

Parallelism (simultaneous updates to all nodes)

\*\*\*\*# docker service update --update-failure-action pause |rollback |continue websvc OR Service-Name

28-B Below cmd will delay updates for 60s before next node updates etc

docker service update --image=httpd:latest --update-sdelay 60s websvc

OR

28-C, Parallelism (simultaneous updates to all nodes); Below command will update 3 instances

at the same time and move to next 3 until all instances are all updated.

Rolling updates Options cmd as shown below: Remember by Default Rolling update is "Puase"

\*\*\*\*# docker service update --update-failure-action pause |rollback |continue websvc OR Service-Name

\*\*\*\*\*\*\*# docker service update --update-failure-action pause |rollback |continue --image=httpd:alpine websvc

#docker service update --image=httpd:latest service-name

docker service update --rollback service-name

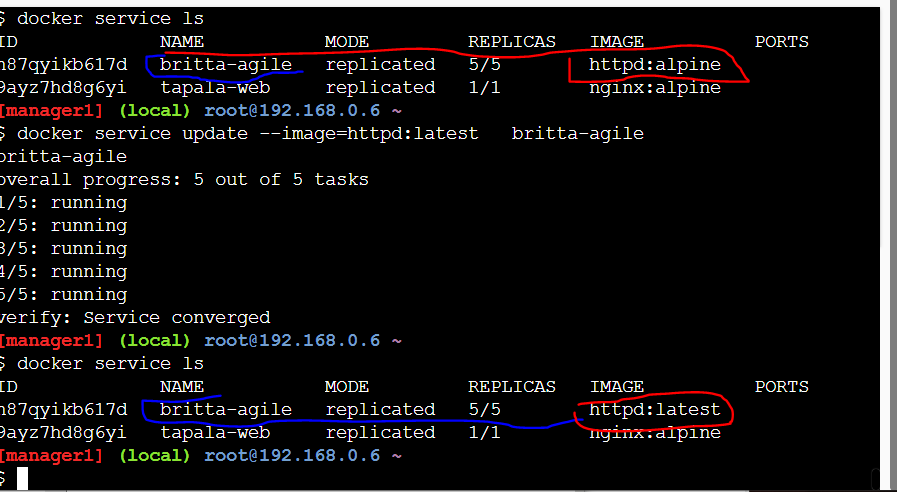
OR

29.A Rollback sample cmd as shown below:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# docker service update --rollback websvc Or Service-NAME

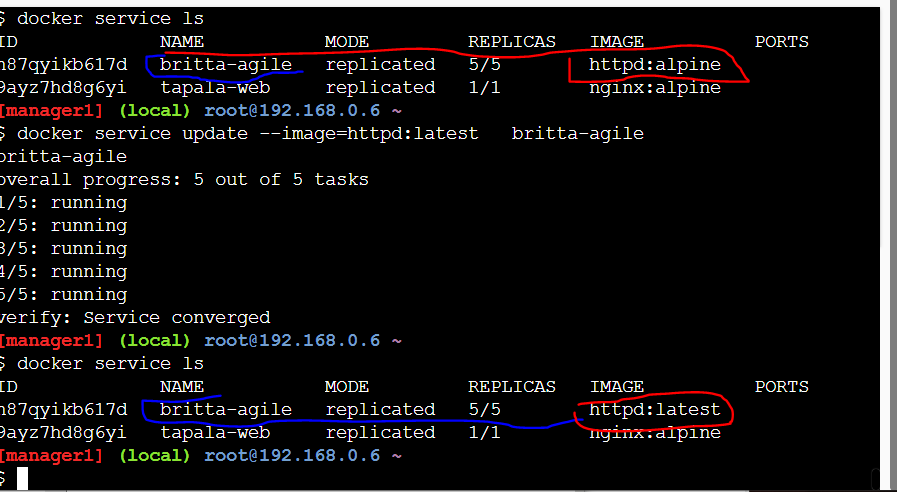
#docker service update --update-failure-action rollback --image=httpd:alpine websvc



###### Verify #####

docker service ls

docker service inspect websvc



### Replication Vs Global

############################################################################

Replica is by default while Global ensure or commonly use to deploy a specific service/application

to all nodes, example, anti-virus, aginx e.g Logs or Monitoring

docker service create --mode=global nginx [Here nginx is the service, it may be anti-virus etc)

docker service ls

docker service update --help ( For docker service cmd guides )

Rolling Update as shown below:

#docker service update --image=httpd:latest websvc

FYI s(Don’t do execute below cmd except is really needed and make sure you saved the key if not you will be locked out from your entire swarm)

DOCKER-SWARM AUTOLOCK FEATURES (Additional Security)

Prevent none-Admin from adding nodes into swarm

or even listing nodes within the locked cluster

docker swarm update --autolock=true

"After the swarm is LOCKED the unlock-key will be provided here"

TEST: # docker node ls

To unlock-key use below cmd:

docker swarm unlock

31-A. If you forgot the Unlock-key after few months etc, then use below cmd:

docker swarm unlock-key

"unlock-key will be provided here, copy & paste in host/node "

PLACEMENT STRATEGIES: Using labels & Constraints

(DECIDING WHICH APPLICATION TO BE INSTALLED IN A SPECIFIC NODE WITHIN A CLUSTER)

Below command with "!=" ( !=memory-optimized httpd ) means deploy in all but Not on stated node/application as shown below:

docker service create --constraint=node.labels.type!=memory-optimized httpd

docker service create --constraint=node.role==worker web

docker node ls

docker node update --label-add type=cpu-optimized worker or worker-node-Name here

In order to Verify (to See the Labels etc) use cmd below:

docker node inspect worker-node-Name --pretty

docker service create --constraint=node.labels.type==cpu-optimized nginx

OR

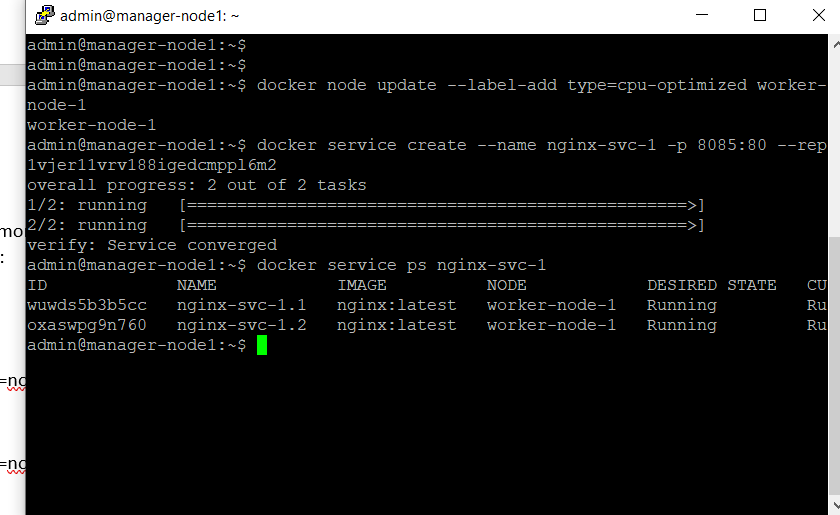
docker service create --name nginx\_svc -p 80:80 --constraint=node.labels.type==cpu-optimized nginx

OR with numbers of Replicas as desire as shown below:

docker service create --name nginx\_svc -p 80:80 --replicas=2 --constraint=node.labels.type==cpu-optimized nginx

To Check or Verify

docker service ls OR docker service ps nginx-svc or svc\_NAME



\*\*\* DOCKER SWARM STACK (Using Docker-compose.yml)

Docker stack (use to deploy multiple applications)

Unlike single application with just docker service create or docker-compose up

command,

Example to deploy stack as shown below:

$ docker stack deploy --compose-file docker-compose.yml

33-A To check stack status use below cmd :

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

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

#docker stack ls

docker stack services

#docker stack ps

# docker stack rm [To remove the services etc]

33-B. \*\*\* Creating a docker stack; see below: \*\*\*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

$mkdir voting-app-stack

$cd voting-app-stack

#ls

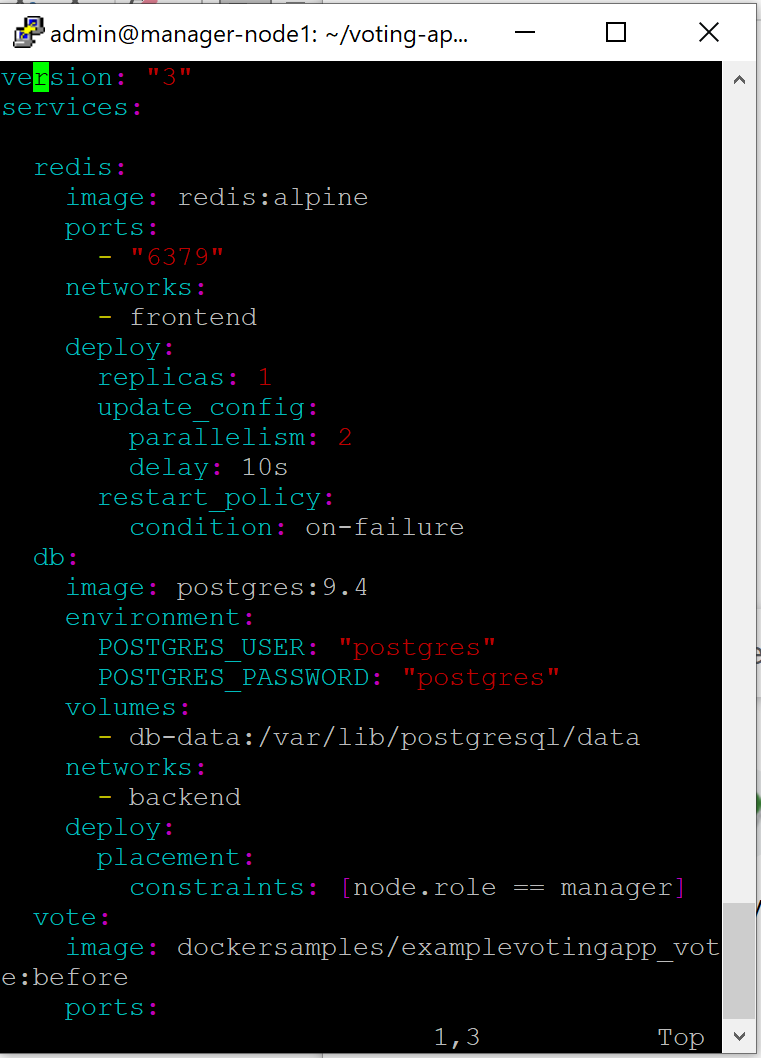
# Create stack-file as below:

$vi docker-stack.yml

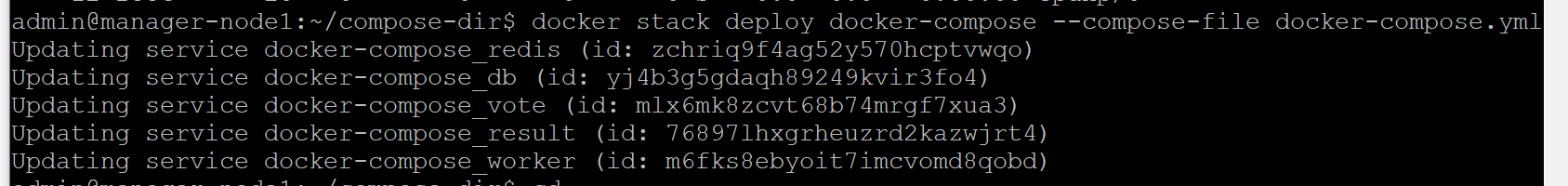
Editing your docker compose-files for stack deployment using roles

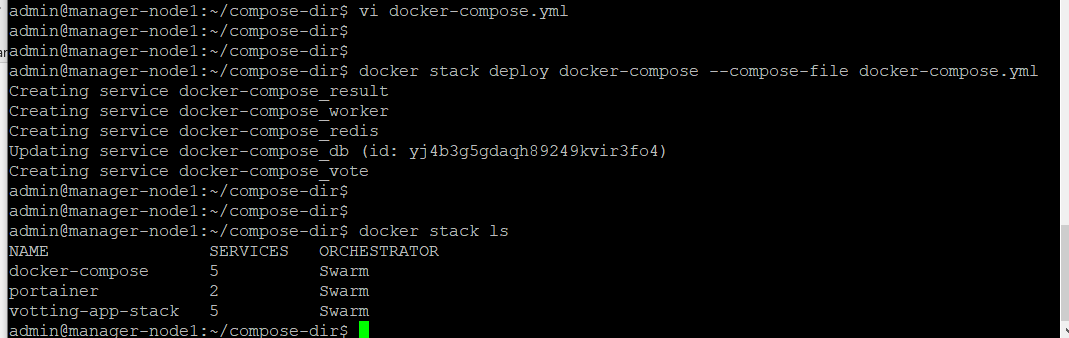
as shown below

$vi docker-compose.yml



$docker stack deploy docker-compose –compose-file docker-compose.yml





Docker-compose | Docker stack

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

|

docker-compose updocker-compose up docker-compose.yml |docker stack deploy --compose-file

docker-compose.yml

|docker stack ls

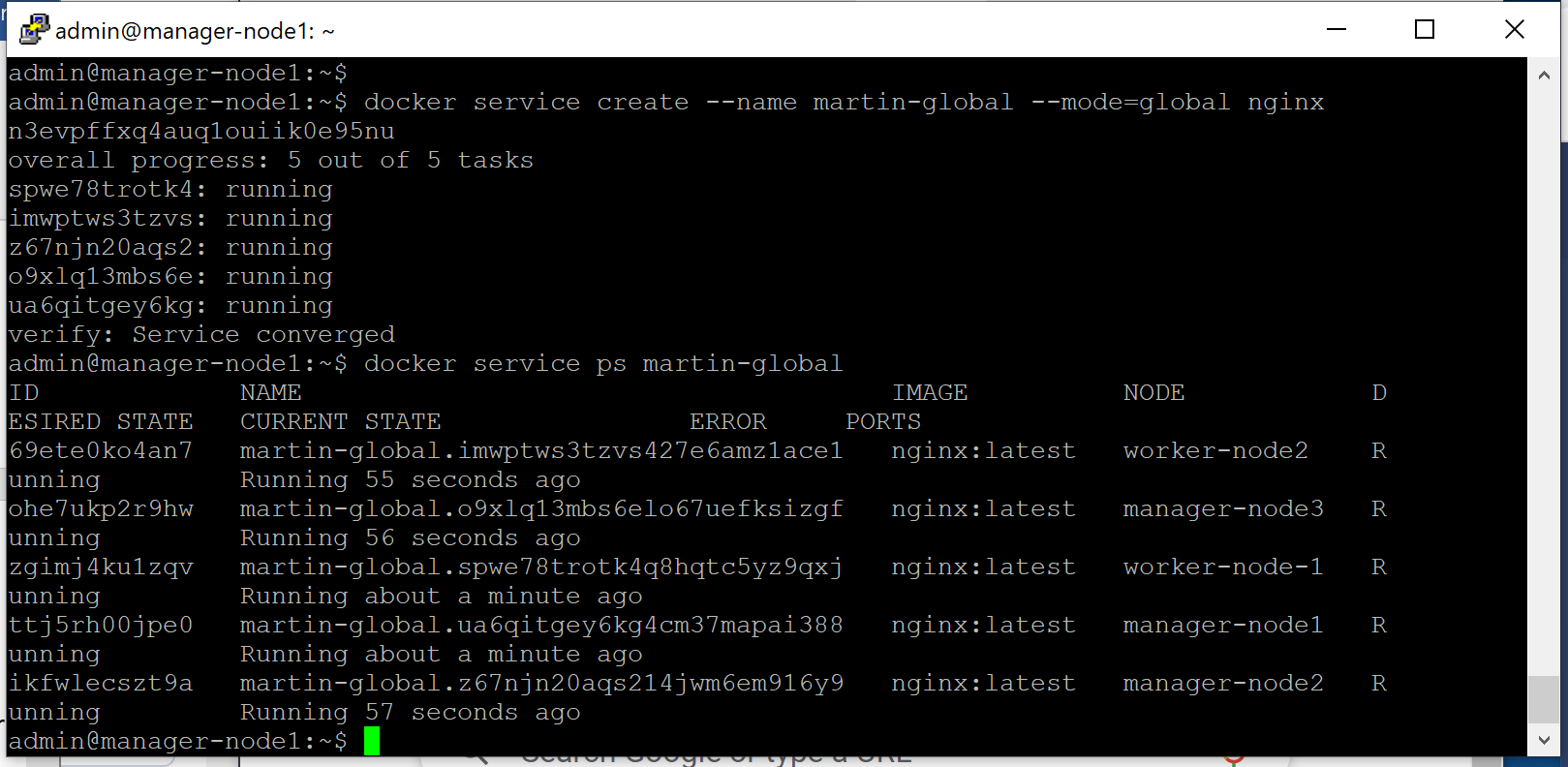
|docker stack ps

| docker stack services

|docker stack rm

Global mode deployment to ensure the service is deployed to all nodes in a swarm

$ docker service create --name martin-global --mode=global nginx



Docker mounting for Persistent data/Volume (Volume papping)

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

$docker volume create data-vol

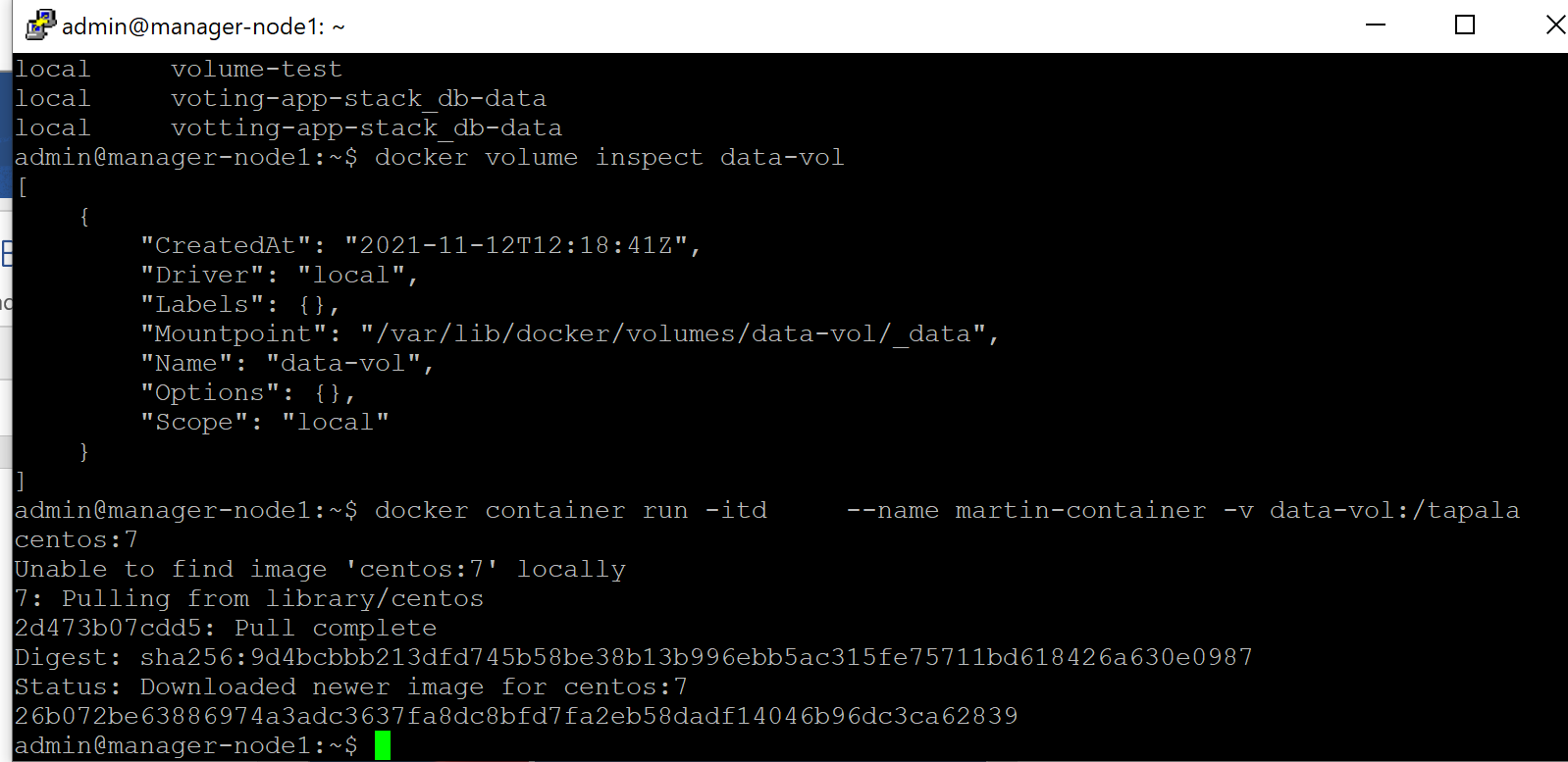
$docker volume ls

$docker volume inspect data-vol Or Volume-Name

$

$docker container run -itd --name martin-container -v data-vol:/tapala centos:7

#docker container run -d --name=wp-mysql --mount /data:var/lib/mysql -p 3360:3360 --network my-net mysql:5.7



map a directory outside the container host to a directory inside the container as shown below:

---->>>>>>>> map a directory (/opt/datadir) outside container to /var/lib/mysql inside host container using -v or mount option etc

All data will be store in the external volume /opt/datadir even if the container(mysql) is deleted. See below

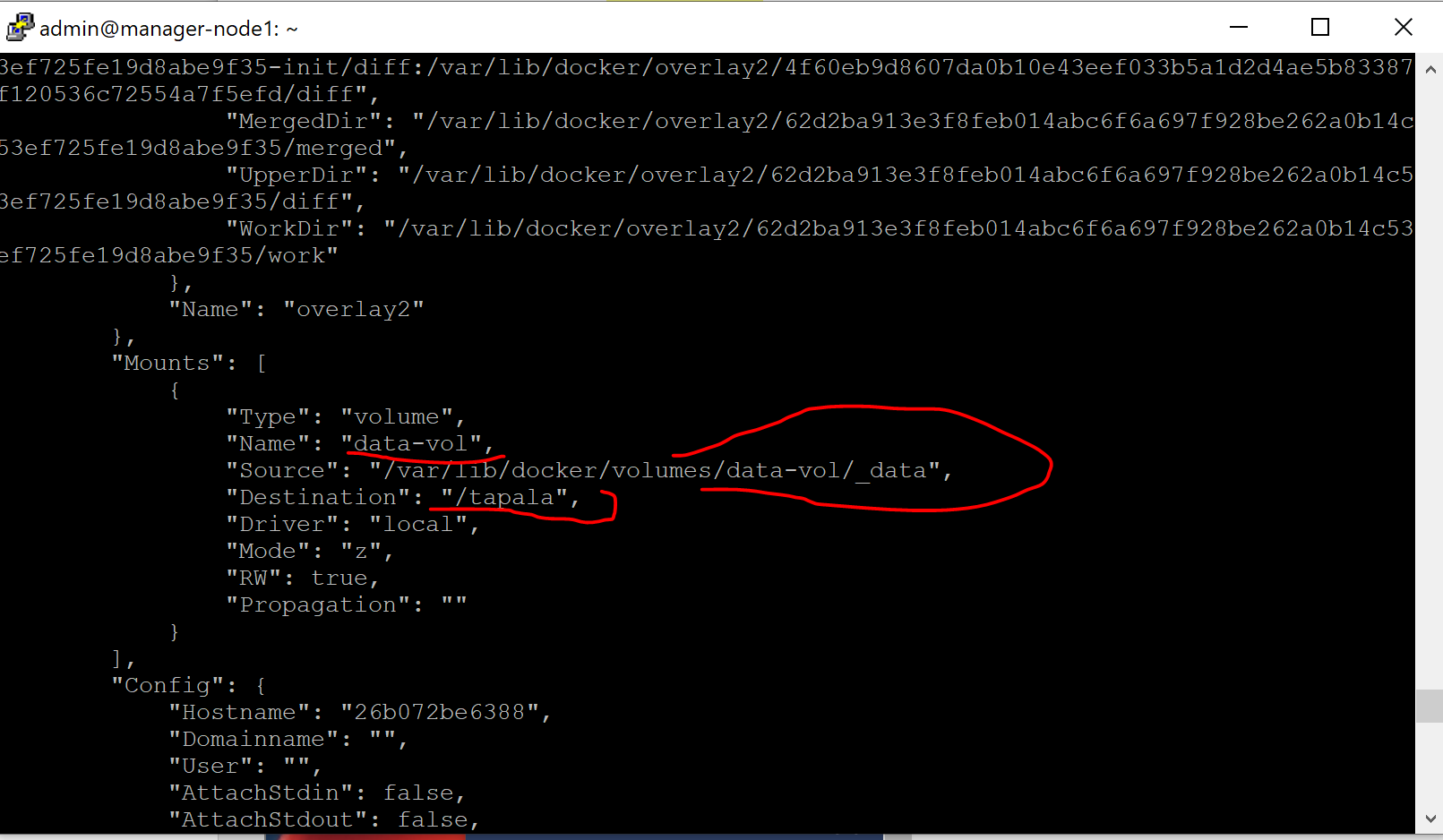
$sudo mkdir /opt/datadir

$ docker run -v /opt/datadir:/var/lib/mysql mysql

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

New created directory outside container host: /opt/datadir

$docker container inspect martin-container



Mapping docker volume to one location via NFS shares

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

##### Setup of NFS shares for Docker-swarms as shown below: ###

Installation steps on NFS server setup:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

yum install nfs-utils nfs-utils-lib

systemctl enable nfs-server.service

systemctl start nfs-server.service

systemctl status nfs-server

Create a folder and make the changes in permission and exports file

mkdir /root/nfs

chown -R nfsnobody:nfsnobody /root/nfs

chmod 755 /root/nfs

vi /etc/exports

/root/nfs 192.168.1XX.XXX(rw,sync,no\_root\_squash,no\_subtree\_check)

exportfs -a

On client :

yum install nfs-utils

mkdir -p /mount/share

mount 192.168.145.144:/root/nfs /root/mount/share

For Unmount:

umount 192.168.145.144:/root/nfs /root/mount/share

df -h

Configure the firewall:

firewall-cmd --permanent --zone=public --add-service=nfs

firewall-cmd --reload

34-A After NFS Server is setup , See Steps to setup NFS shares in Docker-Swarm

##################################################################################

How to install NFS in Docker-Swarm

Pre-requisite

Update the file with your node ip address

vi /etc/exports

/root/nfs 192.168.1xx.xxx(rw,sync,no\_root\_squash,no\_subtree\_check)

/root/nfs 192.168.1xx.1xx(rw,sync,no\_root\_squash,no\_subtree\_check)

/root/nfs 192.168.xxx.1xx(rw,sync,no\_root\_squash,no\_subtree\_check)

exportfs -a

On Docker nodes install nfs utils

yum install nfs-utils

Add firewall exceptions

firewall-cmd --permanent --zone=public --add-service=nfs

firewall-cmd --reload

Create NFS docker volume on all nodes

docker volume create --driver local --name db\_data --opt type=nfs4 --opt device=:/root/nfs/data --opt o=addr=192.168.xxx.1xx,rw,nolock

Create Docker service with NFS volume

docker service create --name db --network mynetwork --mount type=volume,source=db\_data,target=/var/lib/postgresql/data -e POSTGRES\_PASSWORD=password --replicas 2 postgres:9.4

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Installing Postgres in docker-swarm (Ref Dockerhub)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

$docker pull postgres:alpine

$ docker run --name postgresql-name -p 5432:5432 -e POSTGRES\_PASSWORD=Password -d postgres:alpine

$docker ps [To Test for postgres installation/status]

expose postgres port 5432:5432 to the outside world etc

docker exec -it postgres-name bash [To get inside postgress container]

Given root access to psql

#psql --help

psql -U postgres

#ls

\du [to see the users in postgres]

\l [to list current and new created database etc]

How to create database, cmd as shown below

# Create database database-Name;

create database testDB1;

#\l [To list data and new created database]

\c database database-name [to connect to the database]

\d [to see any relations details within the pgsql database within the swarm cluster]

End of Postgres:alpine installation in Docker-swarm

\*\*\* Install Portainer with Docker Swarm on Linux for Docker GUI \*\*\*

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

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

To get started, you will need:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The latest version of Docker installed and working Swarm mode enabled and working, including the

overlay network for the swarm service communicationsudo access on the manager node of your swarm clusterBy default,

Portainer will expose the UI over port 9443 and expose a TCP tunnel server over port 8000. The latter is optional

and is only required if you plan to use the Edge compute features with Edge agents.The manager

and worker nodes must be able to communicate with each other over port 9001.

Portainer Deployment

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

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

Portainer can be directly deployed as a service in your Docker cluster.

Note that this method will automatically deploy a single instance of the Portainer Server,

and deploy the Portainer Agent as a global service on every node in your cluster.

First, retrieve the stack YML manifest:

curl -L https://downloads.portainer.io/portainer-agent-stack.yml -o portainer-agent-stack.yml

Then use the downloaded YML manifest to deploy your stack as shown below:

# docker stack deploy -c portainer-agent-stack.yml portainer

Portainer Server and the Agents have now been installed.

You can check to see whether the Portainer Server and

Agent containers have started by running docker ps:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

root@manager01:~# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

59ee466f6b15 portainer/agent:latest "./agent" About a minute ago Up About a minute portainer\_agent.xbb8k6r7j1tk9gozjku7e43wr.5sa6b3e8cl6hyu0snlt387sgv

2db7dd4bfba0 portainer/portainer-ce:latest "/portainer -H tcp:/…" About a minute ago Up About a minute 8000/tcp, 9443/tcp portainer\_portainer.1.gpuvu3pqmt1m19zxfo44v7izx

Logging In to Portainer (Docker-swarm GUI)

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

##############################

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Now that the installation is complete, you can log into your Portainer Server instance

by opening a web browser and going to:

https://localhost:9443

OR

https://localhost ip Addr :9000

FYI: ASk Your Admin for Portainer ID (Docker GUI) and Password etc

Replace localhost with the relevant IP address or FQDN if needed, and adjust the port if you changed it earlier.

You will be presented with the initial setup page for Portainer Server.

Additional Reference as shown below:

https://docs.docker.com/get-started/

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* END OF DOCKER-SWARM INSTALLATION \*\*\*\*\*\*\*\*\*\*\*