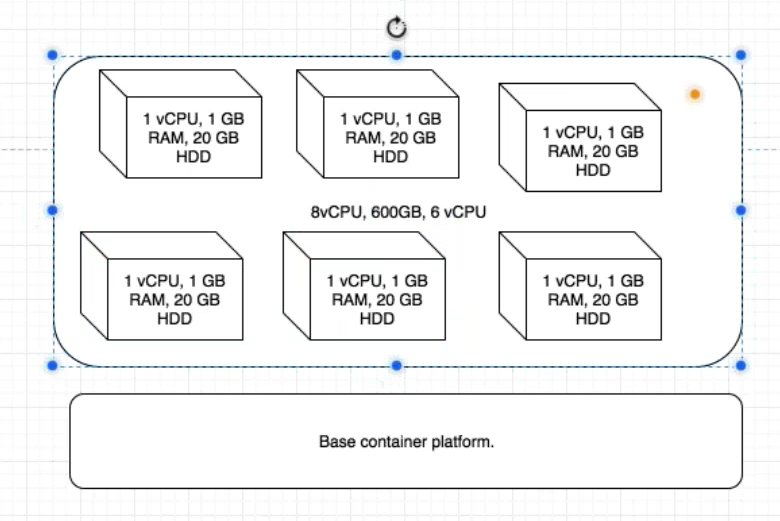
Dockers

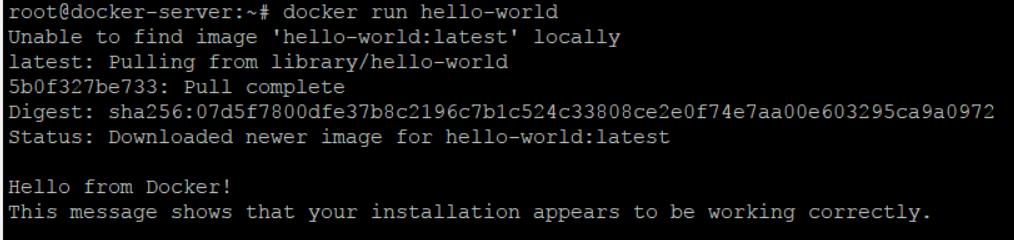
Docker is based on container technology it is similar to vm but there are 2 major short comings in Vm which docker overcome

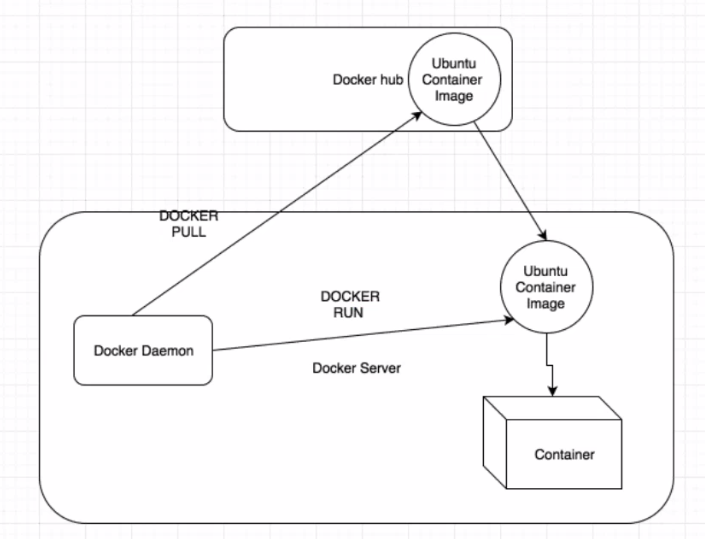
1. VM is fixed RAM and it resources cannot be utilized.it means if VM machine hardware is 40-50% utilized then rest of the hardware would be total waste but docket take the hardware direct from host and give it back if not utilised.
2. Lift and shift of vm infrastructure to cloud is quite problematic and not straight forward.

## *Dockers facts – We can define as many containers as we want but the aggregate hardware utilization should not be 90% (set 90 to be on safer side other you can go up to 99%)of the base platform*



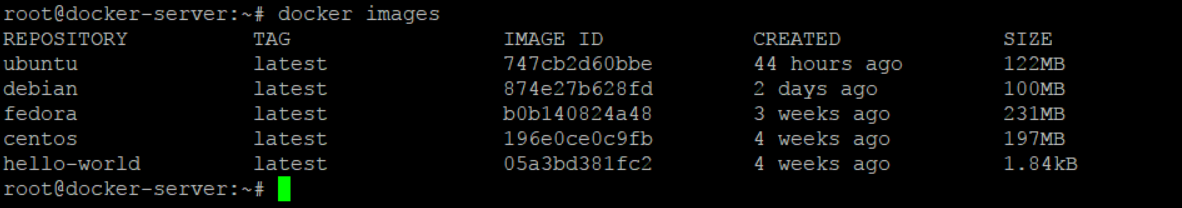
## Docker installation steps

1. Once the ubantu machine is created, update it with *apt-get update –y*
2. Now install some prerequisite for docker like below
3. apt-get install apt-transport-https ca-certificates curl software-properties-common –y
4. Now download the docker key
5. wget <https://download.docker.com/linux/ubuntu/gpg>
6. Now add the gpg key with: *apt-key add gpg*
7. Now add the below repository in last line in: *vim /etc/apt/sources.list*
8. deb [arch=amd64] https://download.docker.com/linux/ubuntu xenial stable
9. Now do: apt-get update
10. Now install the docker community edition with: *apt-get install docker-ce –y*
11. Docker installed now and checked the docker version with: docker –version
12. Now do a connectivity check with docker hub as below
13. docker run hello-world
14. 



## Docker images and container working

Create few images with the help of docker pull centos/debian command like below

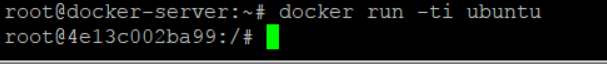


Now create a container with the help of image Ubuntu in interactive mode

docker run -ti Ubuntu

the above command will create the container and go inside it

always do the apt update inside a new container as a first thing



Important – The container and the base machine share the same pool of resources which means what ever the capacity of base machine(RAM, HD), container will also have the same.

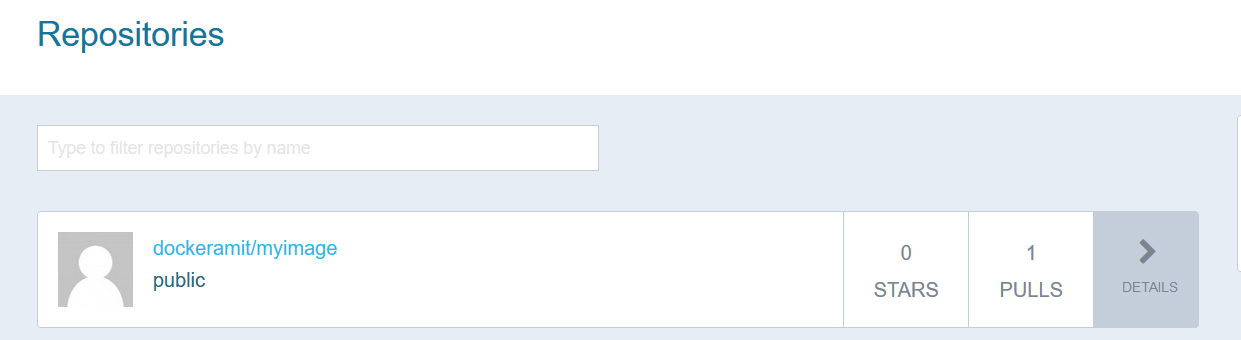
The is no pid for the docker containers, so you cannot kill a container from outside like base machine. The only way is to do it from inside the container. Only the docker deamon PID (dockerd) is exposed to base machine.

## Commit the container and create image out of it

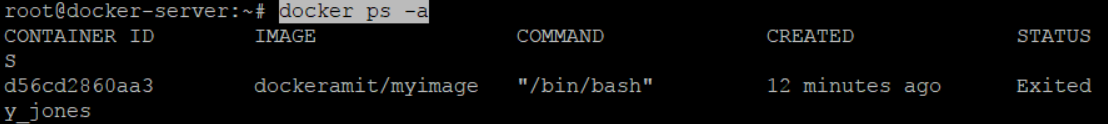
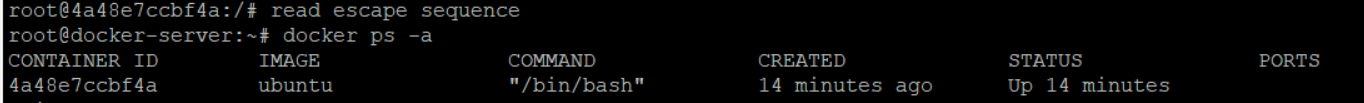
First exit from the container with the exit command and from base machine now check the image id using docker ps –a

1. Now commit the container image using command
2. docker commit -m "created my container" -a "Amit" 998aa9892882 dockeramit/myimage
3. Note – here dockeramit is the userid for hub docker account
4. Now login to the dockerhub using docker login command and push this image to docker hub cloud.

## Pushing and pulling the image from docker hub cloud

1. docker push dockeramit/myimage
2. 
3. Now remove the image from command (optional just for leaning purpose)
4. docker rmi dockeramit/myimage
5. Now when image is removed from base machine, you can pull it from the docker hub account(which we pushed above) with below command
6. docker pull dockeramit/myimage
7. Now you can run this image with command docker run -ti dockeramit/myimage

## Docker Commands and Usage

1. To list the Images : docker images -a
2. Docker current running containers : docker ps –a
3. 
4. To remove the docker images: docker rmi Ubuntu
5. To remove all the images at once: docker rmi -f $(docker images -q)
6. To remove docker containers at once : docker rm -f $(docker ps -aq)
7. Create container out of image in background : docker run -d Ubuntu
8. To create an container from image : docker run -ti Ubuntu (note this will take you inside container so you will have to exit it)
9. To create a container in background and in running state: docker run -ti -d Ubuntu
10. 
11. If container is using high CPU or anyother hardware then to release the hardware memory we can pause it : docker pause 4a48e7ccbf4a
12. To unpause : docker unpause 4a48e7ccbf4a
13. To start a stopped or exited(as exit command inside the container stops it) container: docker start ecd37a8f0233
14. To go inside the already running container : docker attach ecd37a8f0233
15. To exit container without stopping it : clt + p + q
16. 
17. To create a container with customised name : docker run -ti -d --name amit\_container Ubuntu

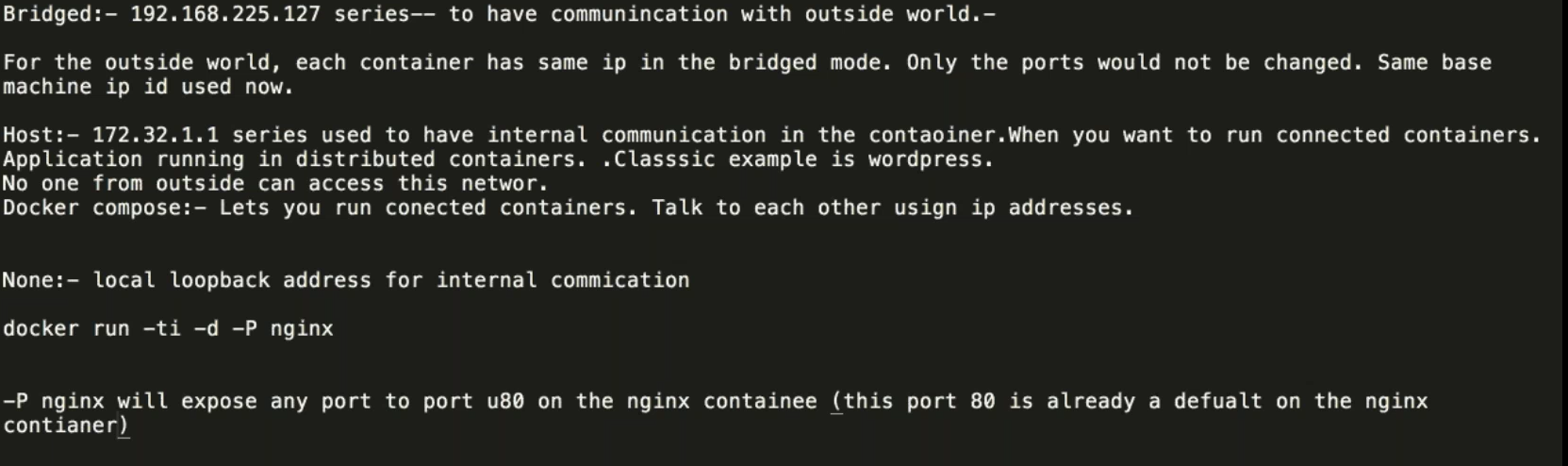
## Docker Networking

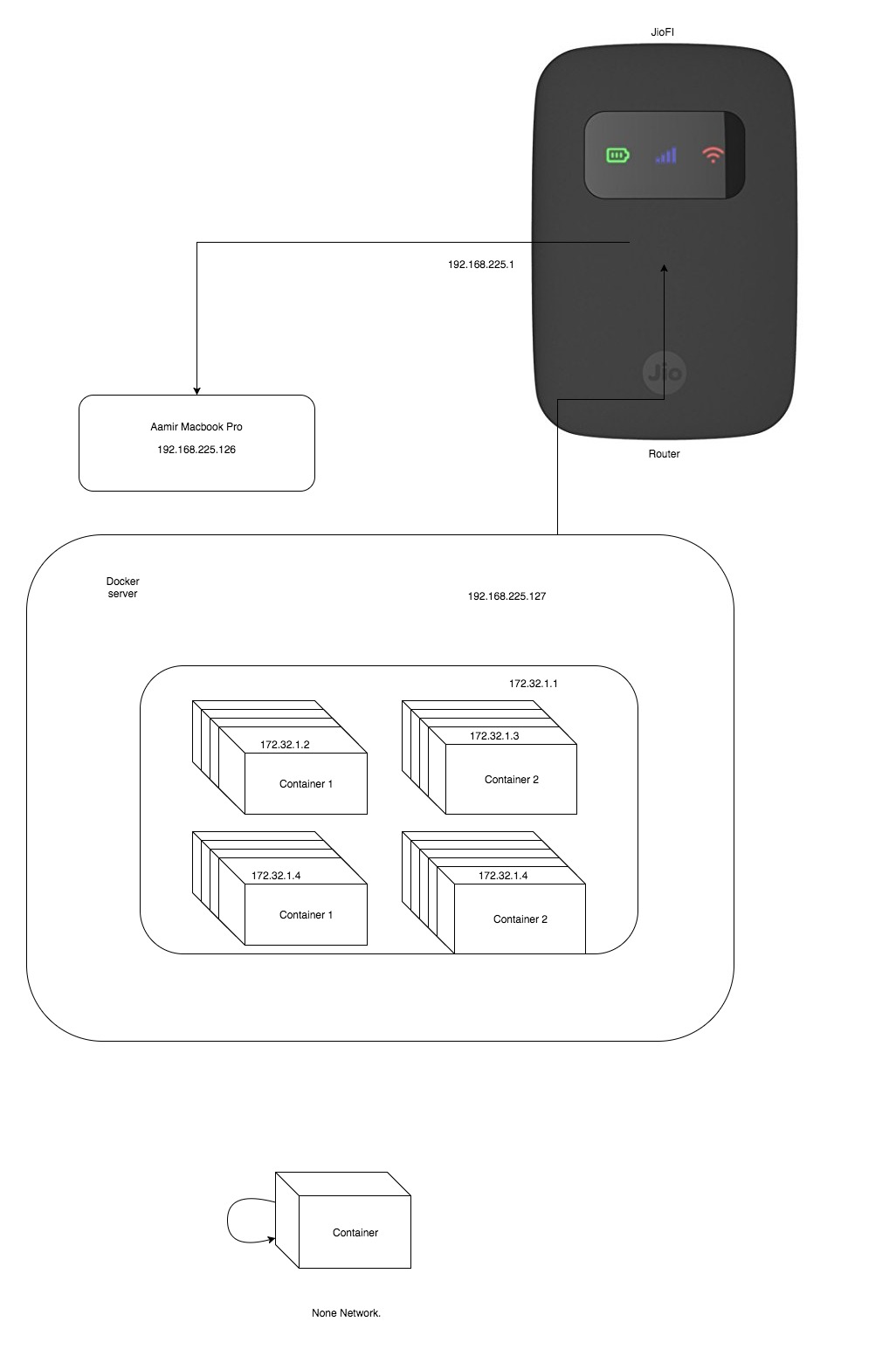
There is 3 types of networking

Bridged

HostOnly

None

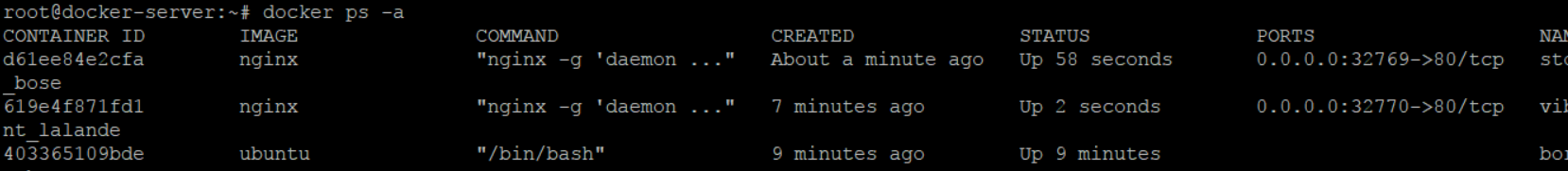




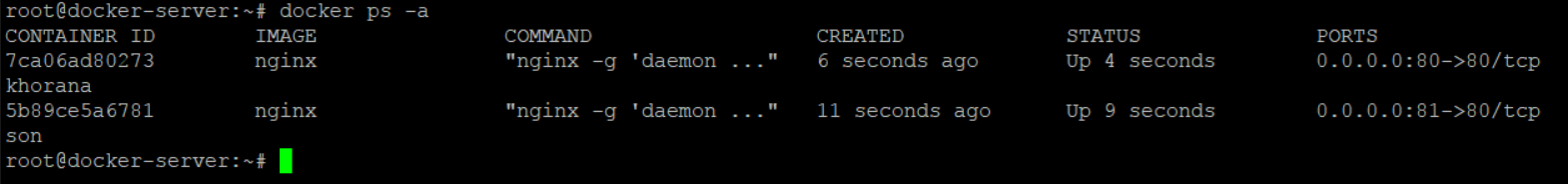
# Ports

Docker ports

**Free Ports (Useful for local development)** – **docker run –P nginx** here docker assigned random port to the nginx container(and keep on incrementing it incase you run multiple) and would expose that to the docker base machine and this can be check using docker ps –a .



**Binding ports** - **docker run -d -p 81:80 nginx** here everything remains same just that port is assigned by user. You can assigned multiple ports like this docker run -d -p 80:80 nginx



# Volumes

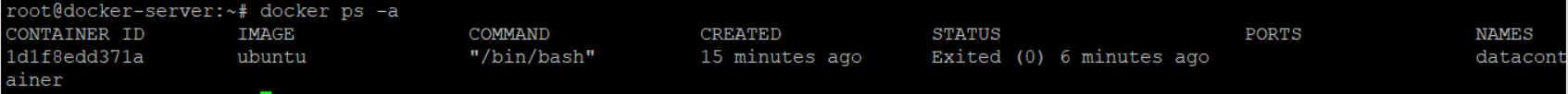
There are 2 types of folder volume creation in docker and one of them is called data container

### Data Containers

There are 2 types of folder volume creation in docker and one of them is called data container. In this we create a directory and we expose it to all the container on that base machine. So any container which is created, will have that mounted folder and all of its global data(which means all the data from all the containers in this mount persist and do not lost even if the containers are destroyed). The data will be lost only if main data container is destroyed.

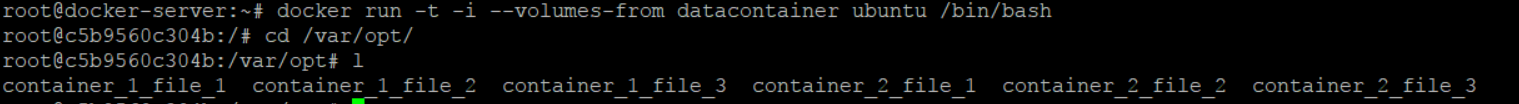
Command to create data container

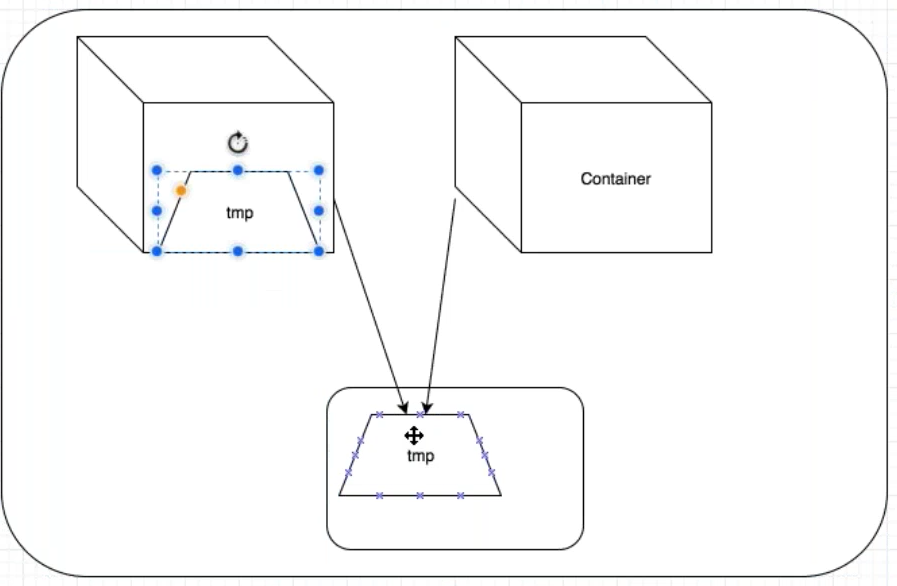
docker create -v /var/opt --name datacontainer ubuntu /bin/bash



Now other container can be created and mounted to this container like below

docker run -t -i --volumes-from datacontainer ubuntu /bin/bash





### Persist in Base machine

First create a folder inside the base machine (say mkdir nginxlogs) and then run the below command.

**To create a nginx container and to have log from /var/log/nginx persisted in base machine**

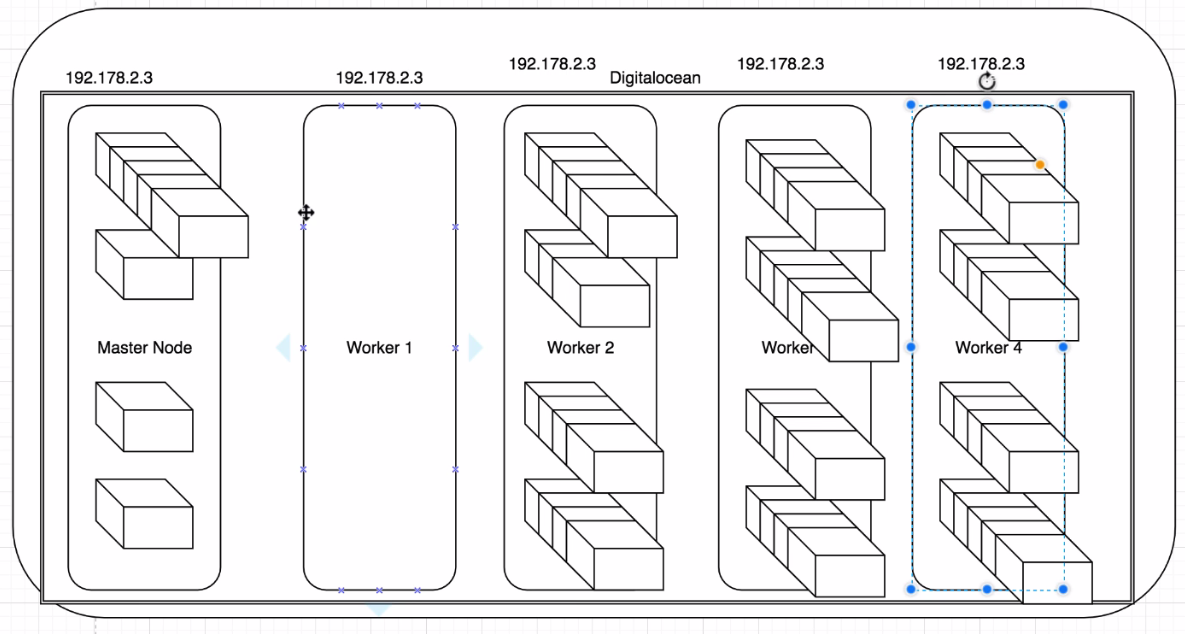
docker run -d -v /root/nginxlogs/:/var/log/nginx -p 5000:80 -i nginx

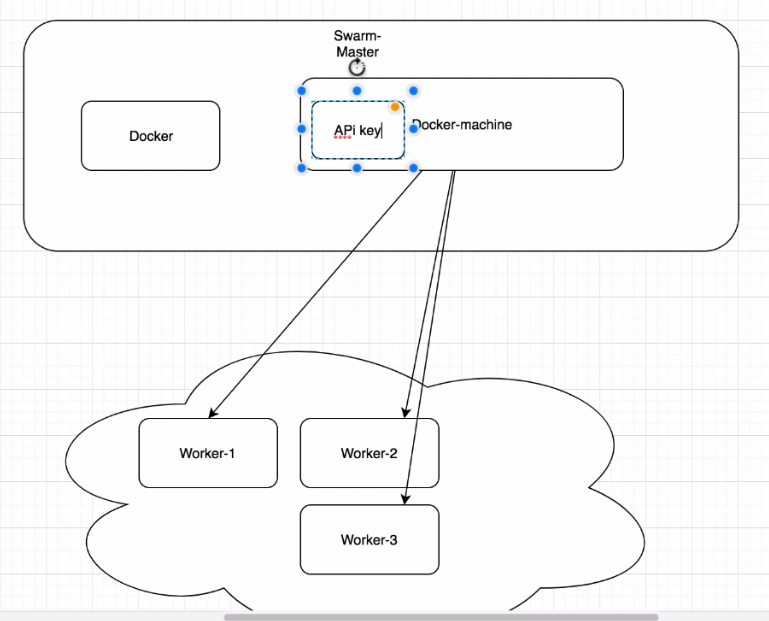
**To create a normal Ubuntu container and have the log persist in base machine**

docker run -d -ti -v /root/nginxlogs/:/var/log/nginx ubuntu /bin/bash

Docker Swarm cluster

In the swarm cluster, any container can be accessed using any machine(master or worker) IP address which means this whole cluster works as single piece of machine.

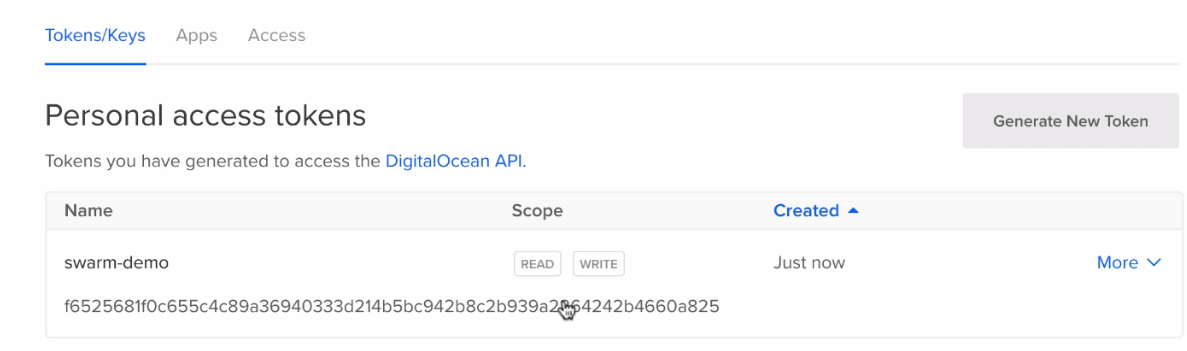
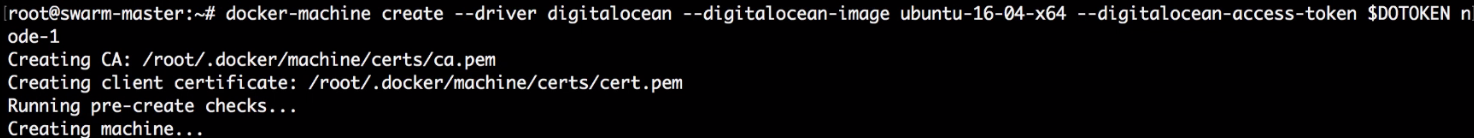
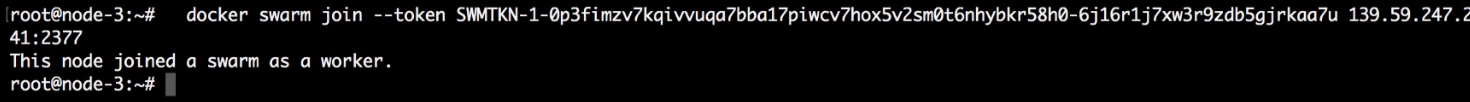
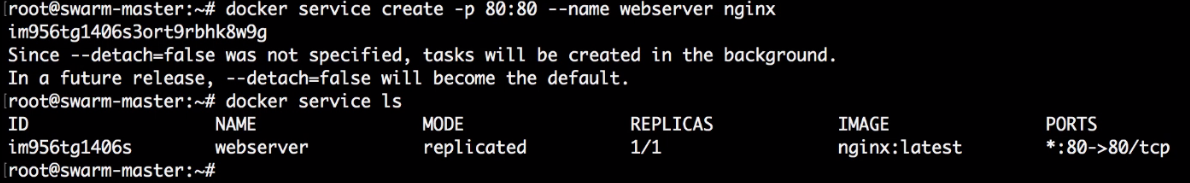
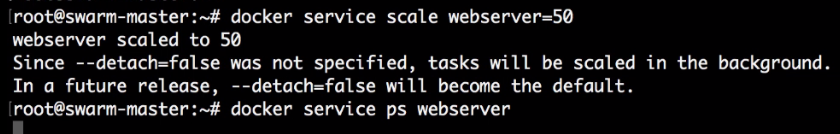




# Swarm Master

Swarm master server need to have the docker and docker master installed in it.

Follow the below steps

1. Install the docker on the swarm master server
2. Install the docker master on the server now by following the below steps
3. curl -L https://github.com/docker/machine/releases/download/v0.12.2/docker-machine-`uname -s`-`uname -m` >/tmp/docker-machine &&
4. chmod +x /tmp/docker-machine &&
5. sudo cp /tmp/docker-machine /usr/local/bin/docker-machine
6. now check the docker-machine –version to check if it working.
7. Now docker machine will need the API key so that it could talk to cloud server (digital ocean, AWS) for getting the machine created automatically.
8. Go to the digital ocean cloud and create a key like below
9. 
10. Now copy this key (as it will be visible only once) and save it in environment variable on master server like below.
11. 
12. Now go ahead to create a node one machine with the help of docker-machine
13. 
14. Once this is completed, create node 2 and node 3 as well.
15. Initialled the Swarm cluster, Once all the 3 node machines are created
16. Docker swarm init –advertise-addr ip\_adress\_of\_swarm\_server
17. 
18. Now since cluster is initialled , now join the other machines to this cluster.
19. Now ssh to the other machines like below and run the above one time command to bring them in cluster
20. 
21. Now exit from this node 1 and ssh to node 2 and node 3 and join them as well.
22. 
23. Now check the cluster node with docker node ls command.
24. Now create an nginx server on the master server with below command.
25. Docker service create –p 80:80 –name webserver nginx
26. 
27. The above command means only 1 container is running with nginx and this will be available across all the cluster machines(check by launching the IP addresses of all)
28. Check status with docker service ps webserver
29. Now scale this webserver from 1 to say 50 containers like below
30. Docker service scale webserver=50
31. 
32. Check with docker service ps webserver and you will see all of them are distributed equally.