Play-with-docker.com

Login with docker

Start

Create a new intance

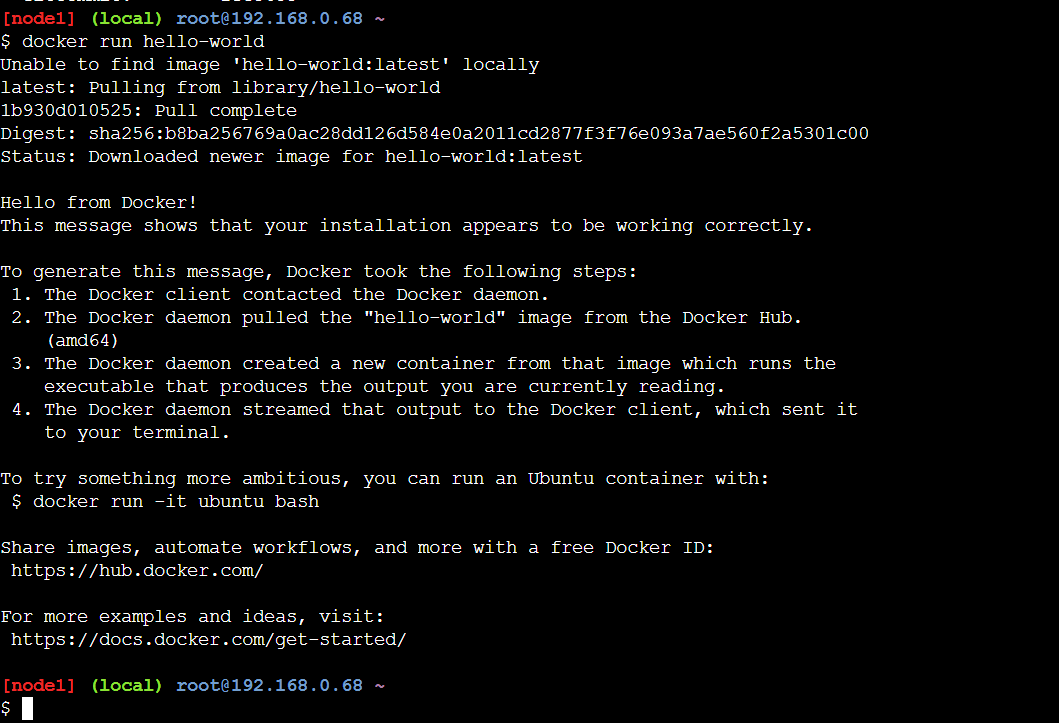
Alt+enter=fullscreen

$ docker version

Docker version

$docker run hello-world

It does not locally have the image so it downloads the image.



Hello from docker

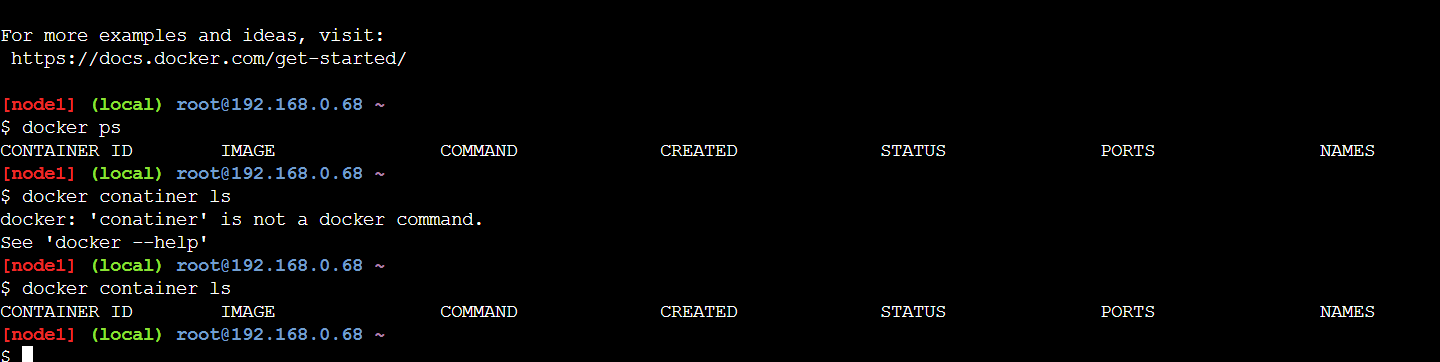
If received this message then the hello-world docker have run successfully.

$docker ps

Gives a list of running containers on our machine.

$docker containers ls

Equivalent to docker ps, but this is more conventional and newer.



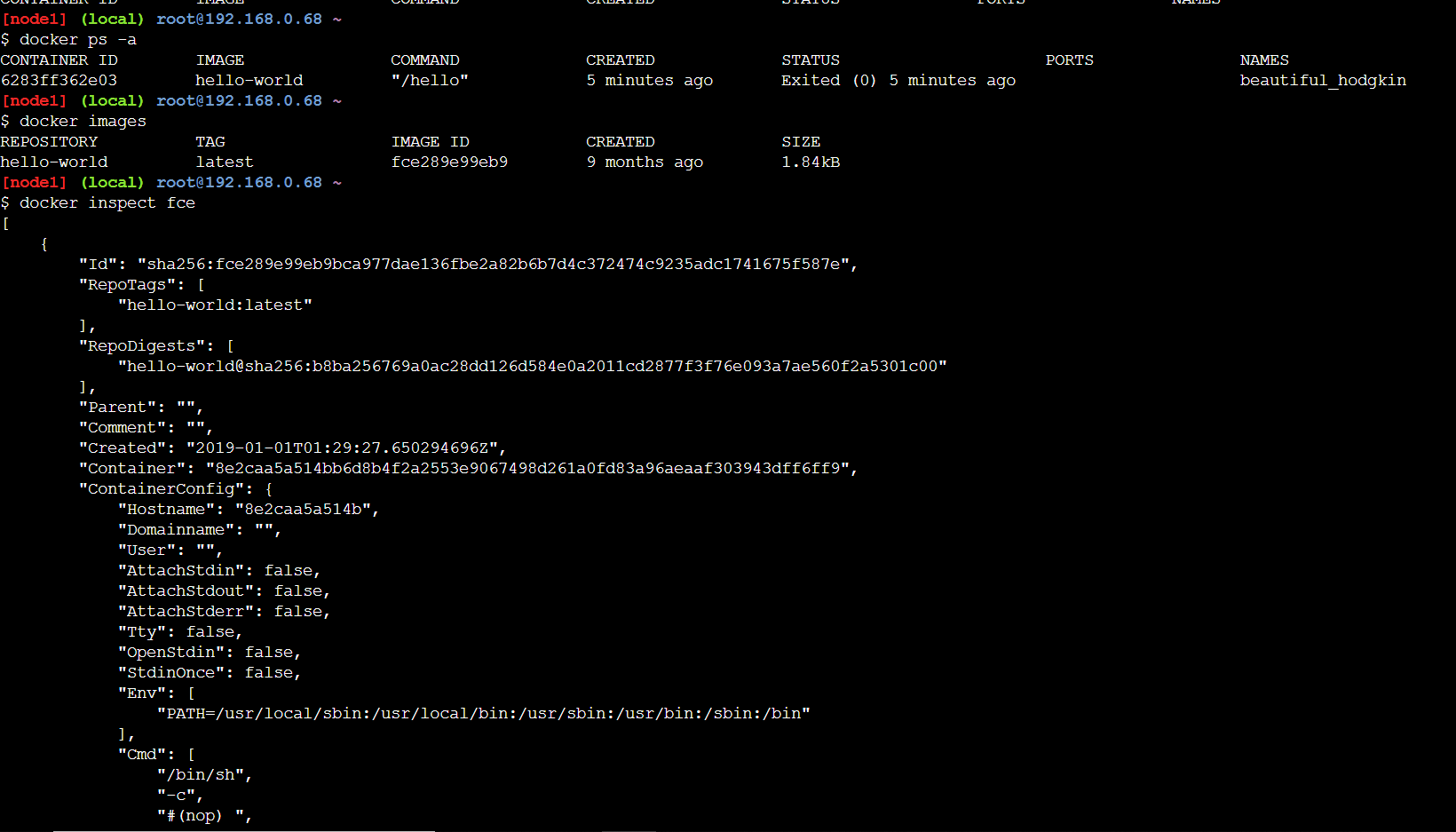
$docker ps -a

If we do not give a docker name, it will be assigned automatically.

$docker images

It lists out all the images that we have downloaded from the history.

$docker inspect fce “here fce is the first three letters of the image id”

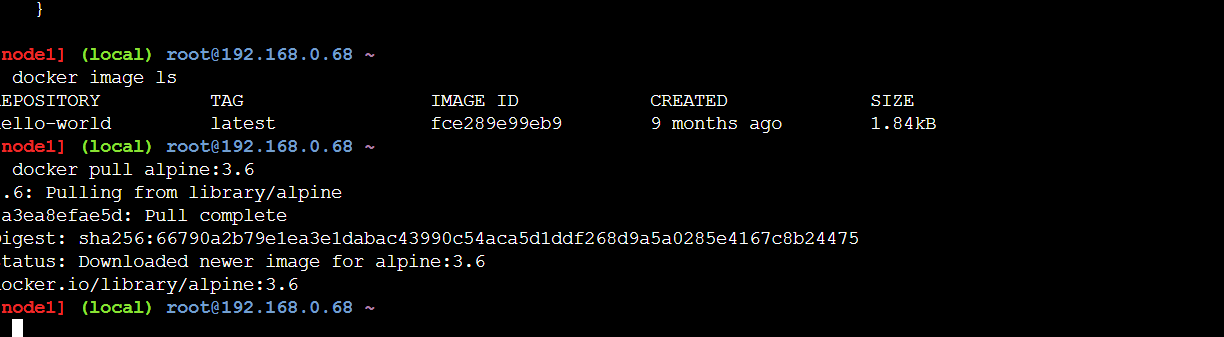


$docker image ls

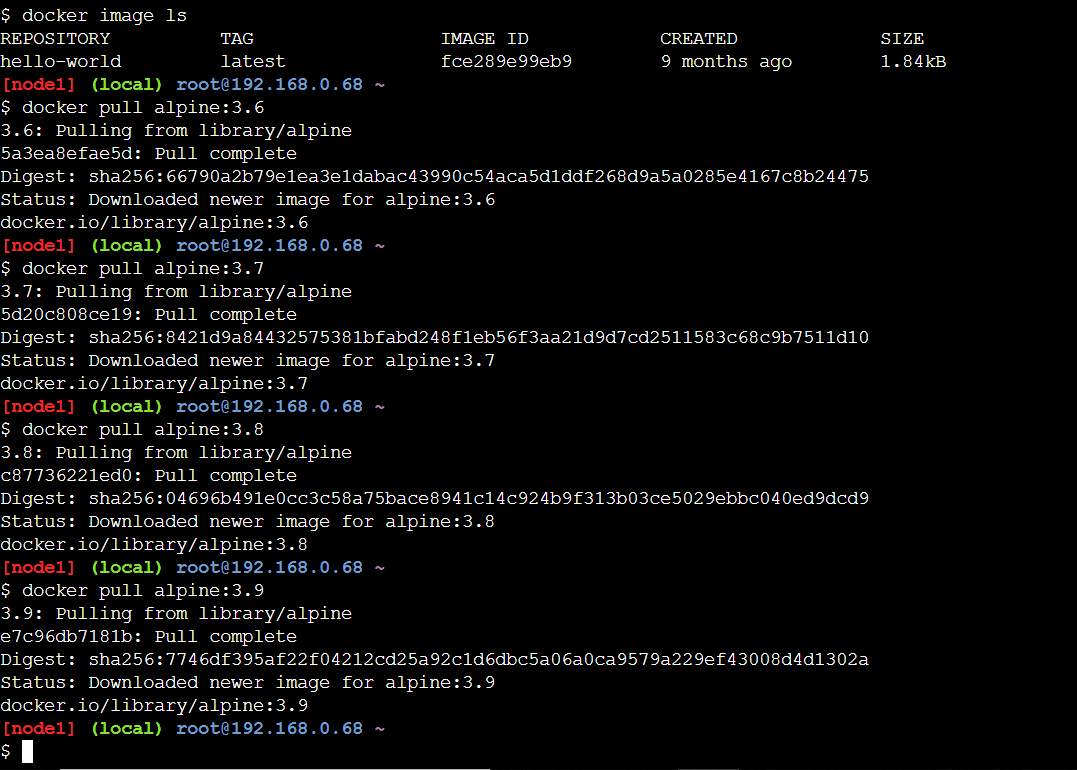
New command ; same result as docker images.

$docker pull alpine:3.6

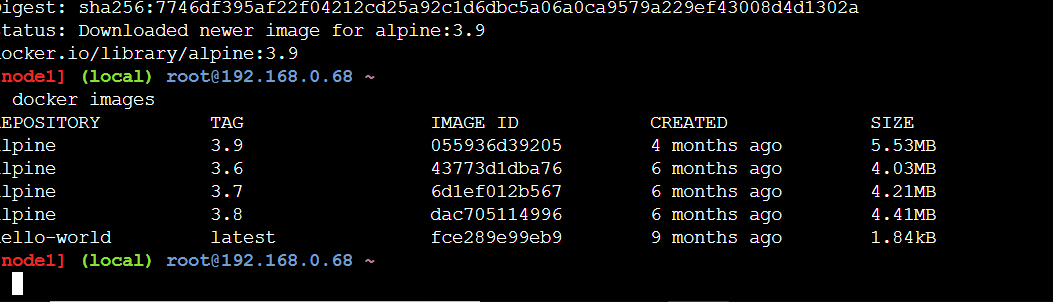
For pulling some images to our file system from the docker registry.

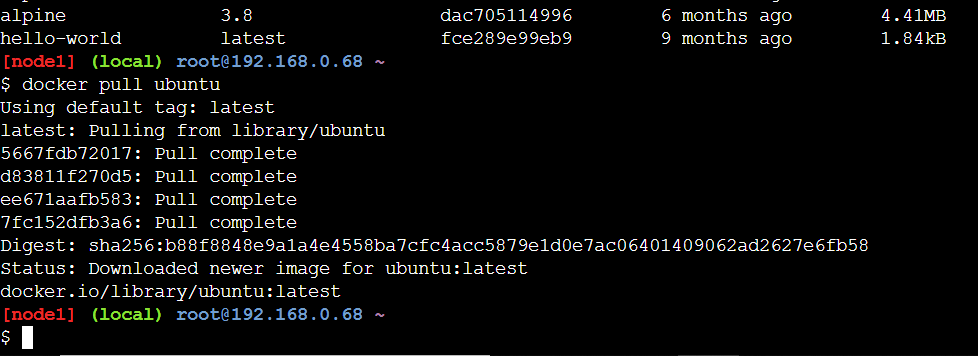


Pull more images,



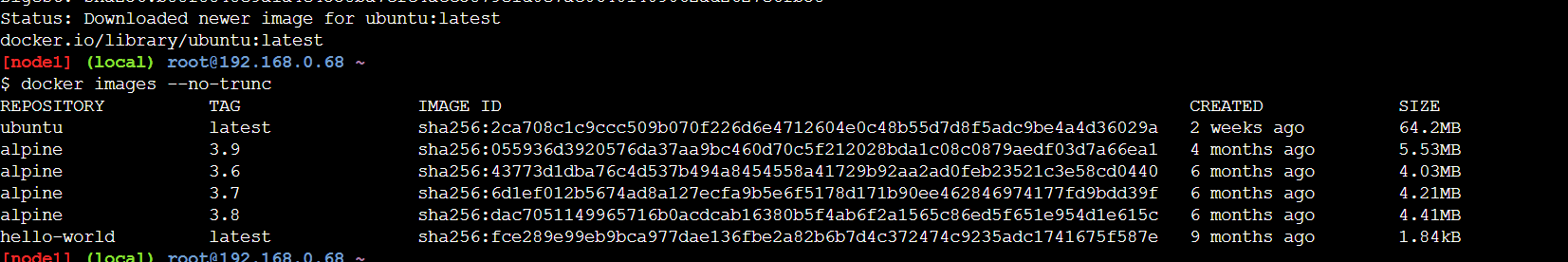
Now when we list out the images, every image is shown with an image tag.





To know the complete sha or the complete image id,

$docker images –no-trunc

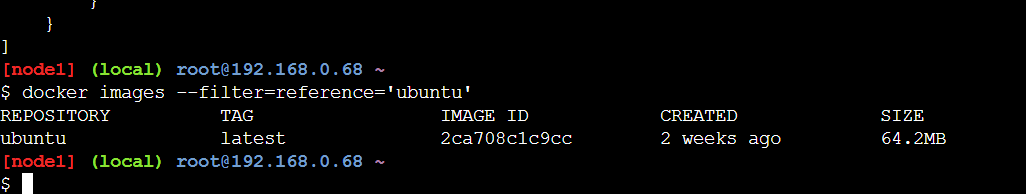


Even ubuntu is made up of many layers we can see that,

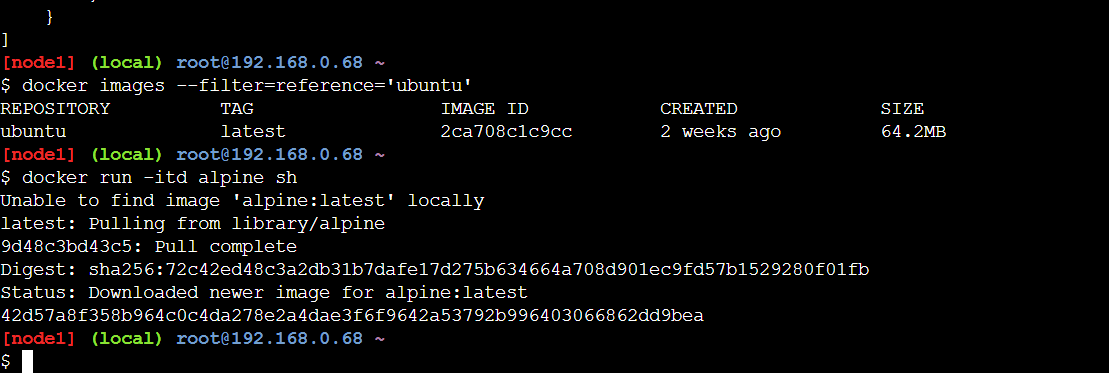
$docker inspect ubuntu

$docker images –filter=reference=’ubuntu’

List out images with filter.



$docker run -itd alpine sh



This will run a sh command inside that image. Alpine is a flavor of an image and it will run as a container.

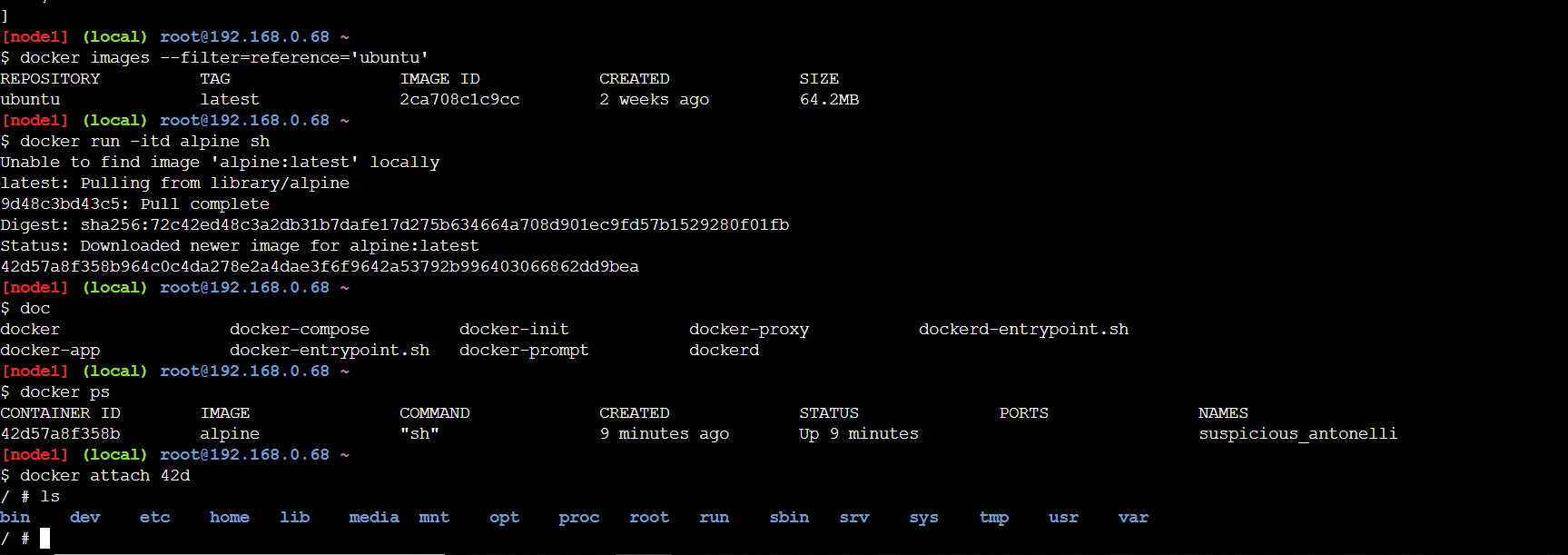
D in the command means detached mode that is the container is running in the background.

If we just used it in the command without d, then It will straight away go inside the container and run the shell.

$docker attach 42d

42d id the first three letters of the container id.

On running ls, it gives all the basic directories and command.



Attach is used to attach to the conatainer.

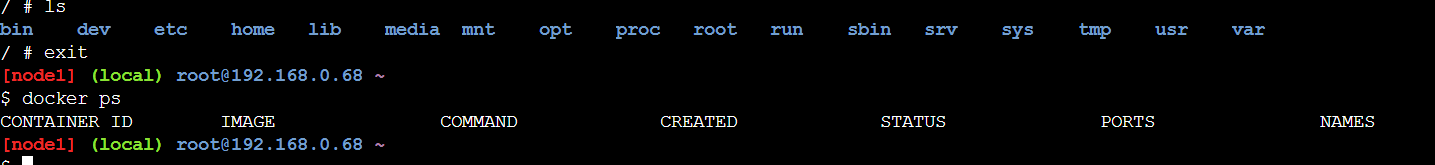
/#exit

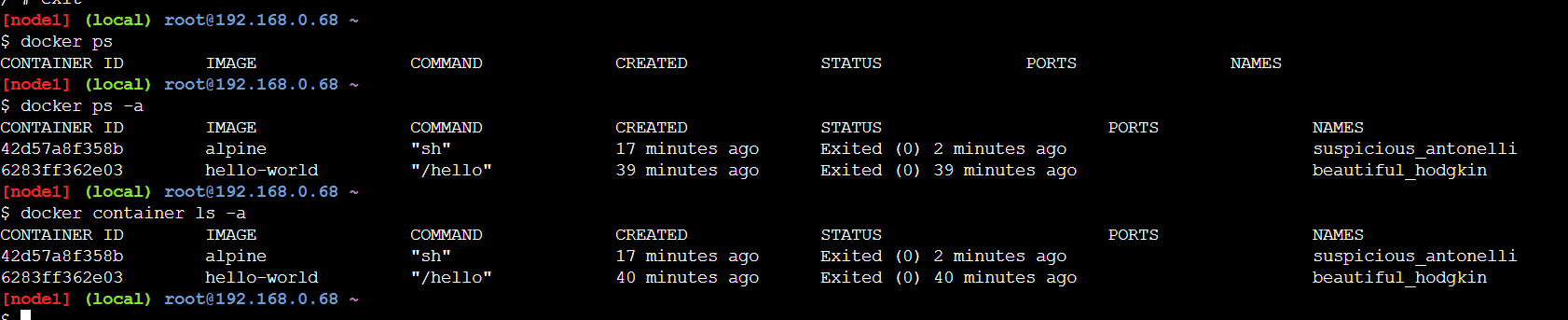
It goes back to the terminal.

$ docker ps

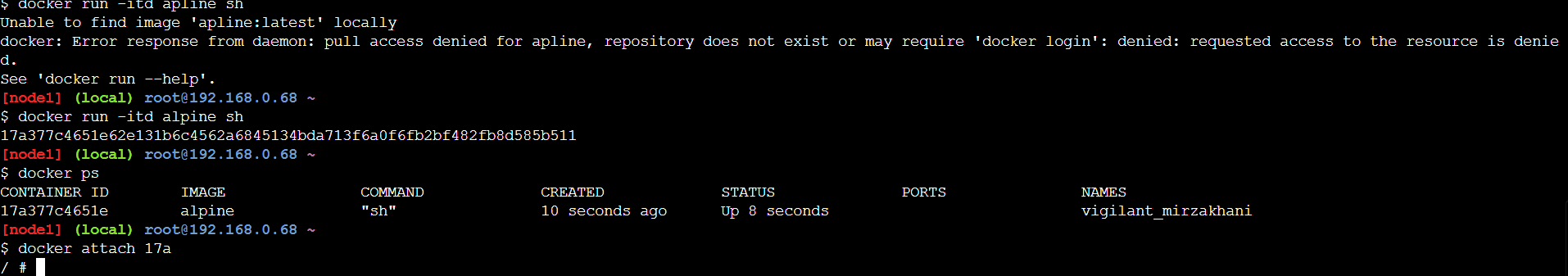
Nothing to be shown, as the container dies, it had the sh command, we did exit and the container finished.

$docker ps -a





Attach to the container again,



Ctrl+pq=container running as well as we r out to the terminal again.

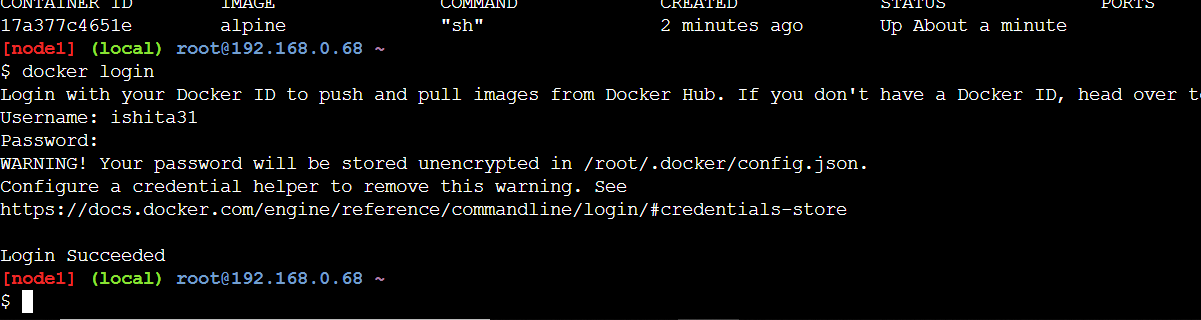


Go to hub.docker.com and search alpine.

It shows all the images from where we have downloaded and all the downloads that have been done.

Now do docker login:

$docker login

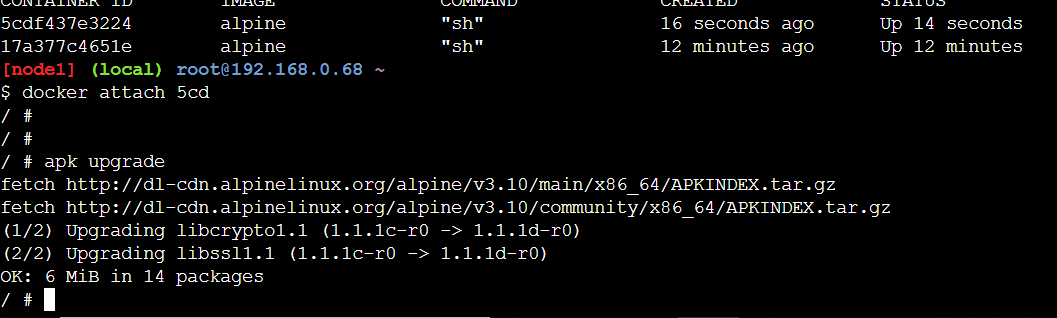


$docker run -itd alpine sh

$docker ps

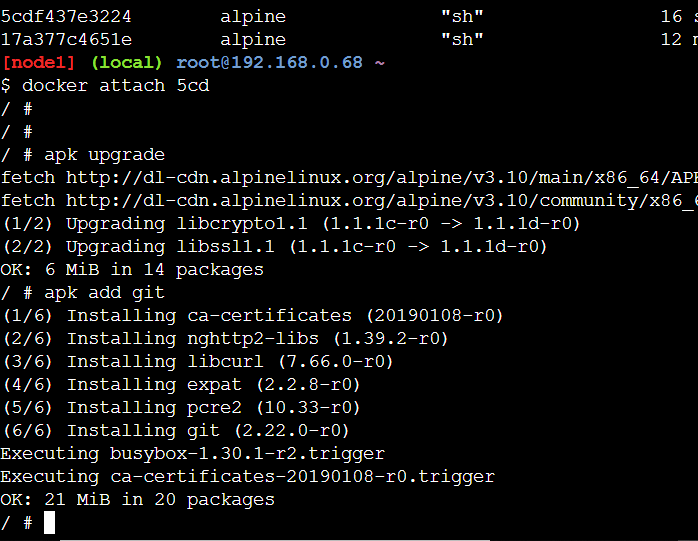
$docker attach 5cd

/# apk upgrade



/# apk add git

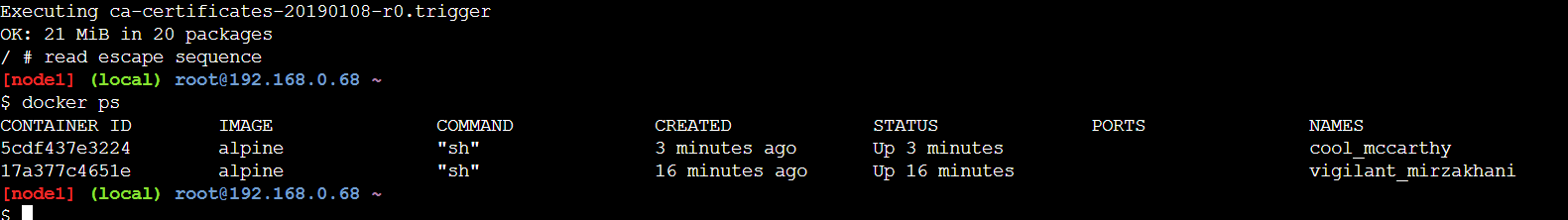
To get attached to git



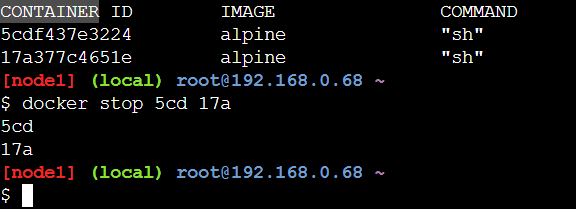
Ctrl+p+q

Do not exit

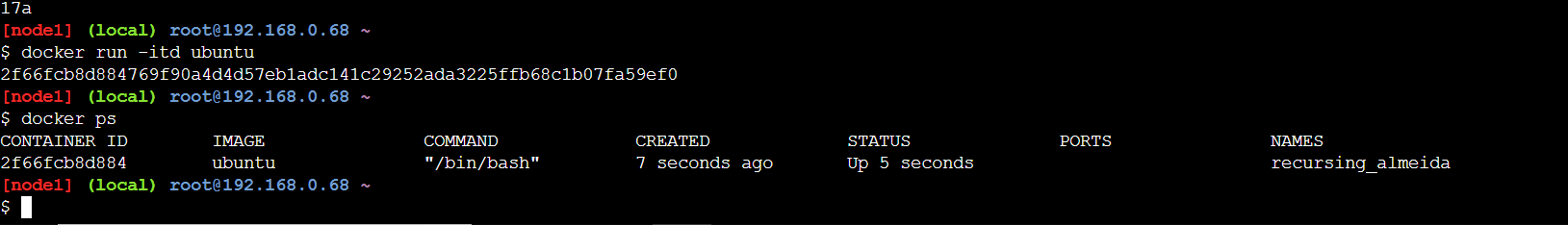
Container should be up and running



To stop both all the conatiners, simple add three letters od the container ids separated by space.

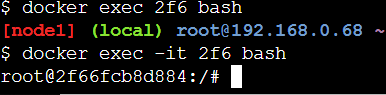


$docker rm all two ids.



Usually to go inside a container, we don’t use attach. We use:

$ docker exec -it first three letters of containerid bash





 If we use [docker attach](https://docs.docker.com/engine/reference/commandline/attach/), we can use only one instance of shell.  
So if we want to open new terminal with new instance of container's shell, we just need to run [docker exec](https://docs.docker.com/engine/reference/commandline/exec/)

$vi Dockerfile

Press i to write inside the docker file.

Write :

FROM alpine

RUN apk upgrade

RUN apk add git

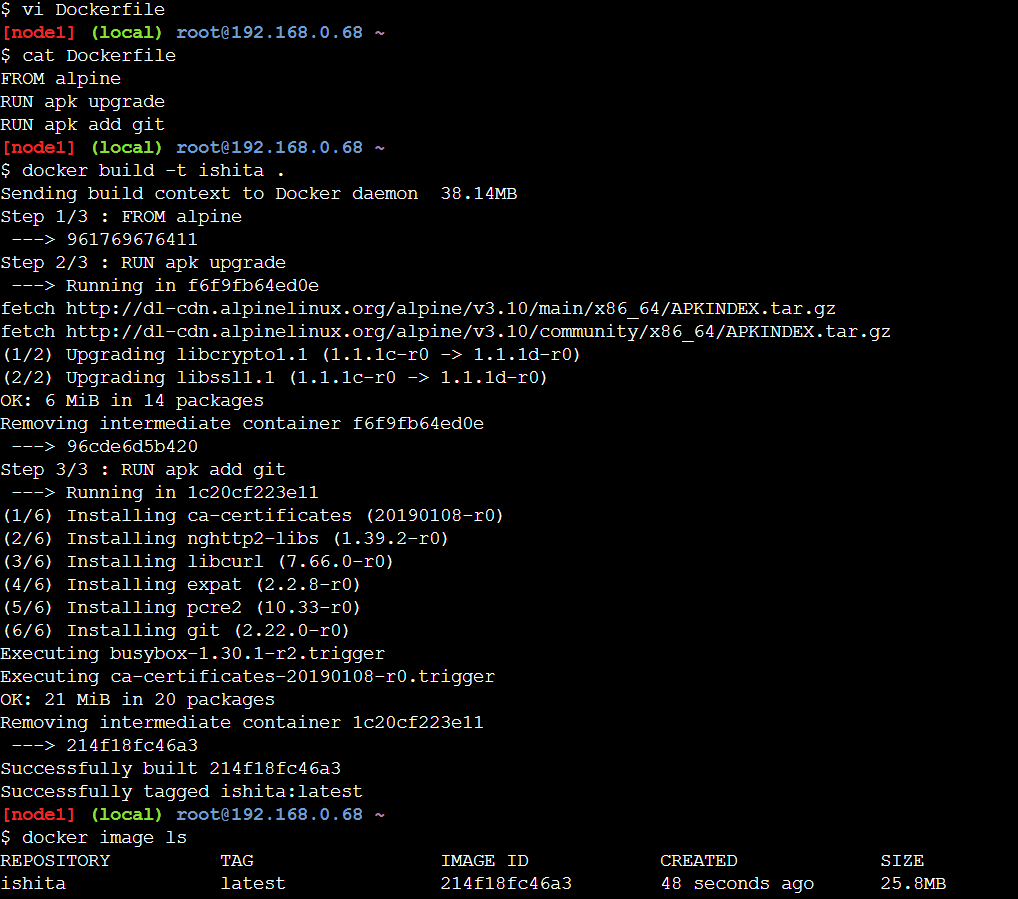
Save using esc :wq

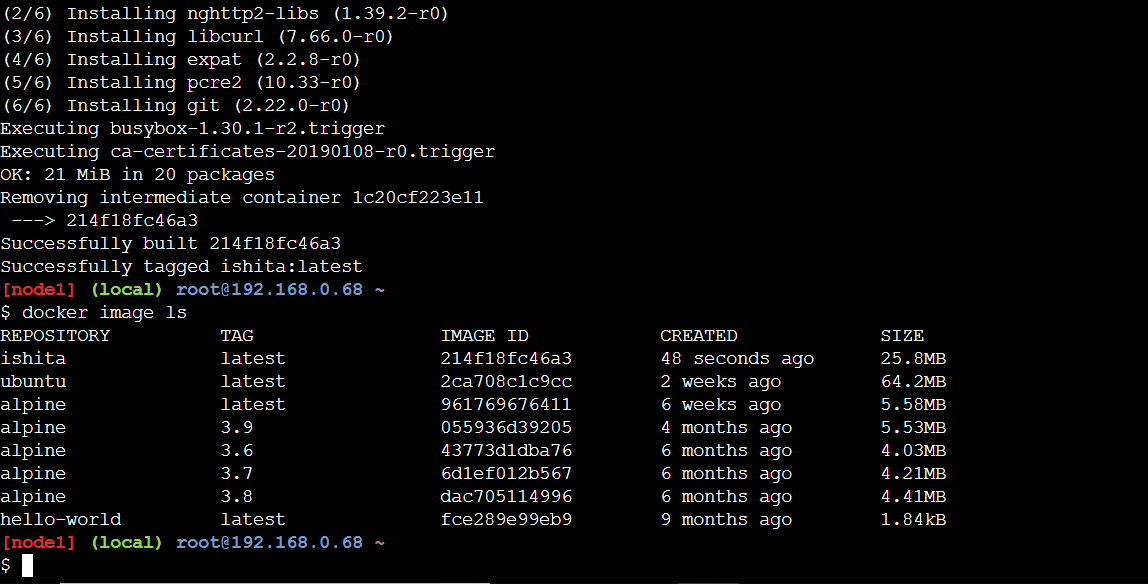
Then cat dockerfile

$docker build -t Ishita .

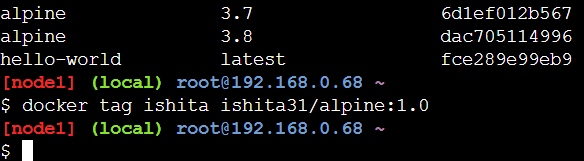
$ docker image ls

An image will be created.

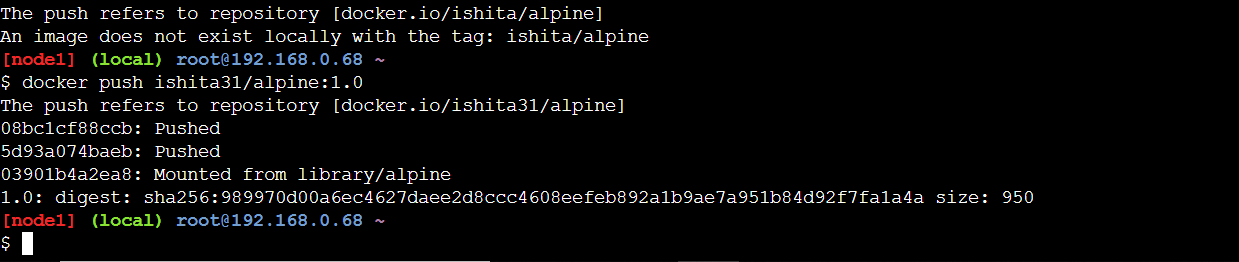


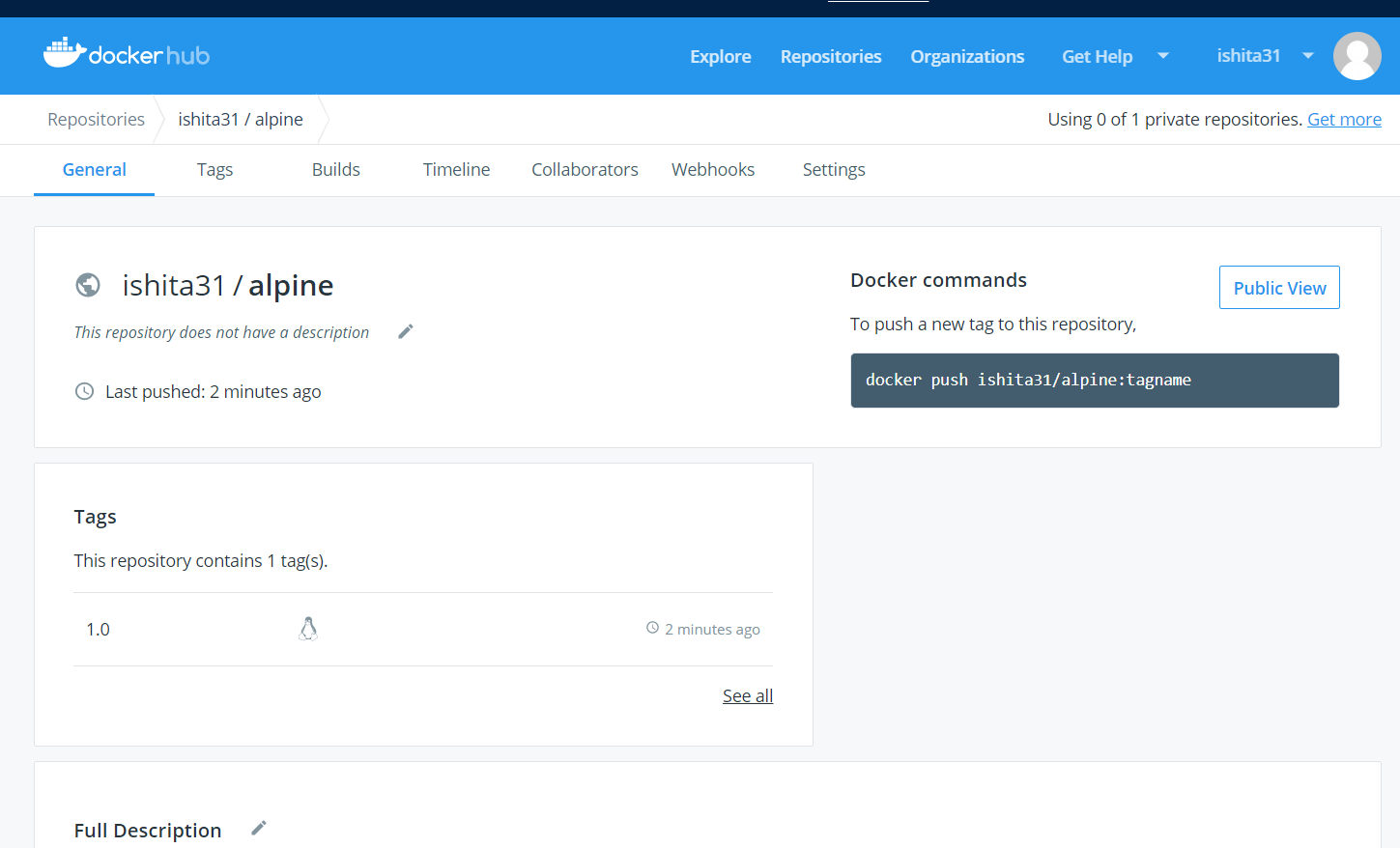


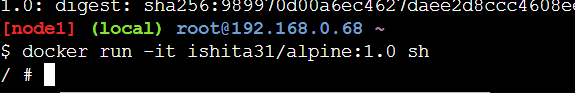
$docker tag imagename dockerid/alpine:1.0



$docker push imagename/alpine:1.0







$docker run -it -v /root:/tmp alpine sh

/# cd /tmp

/tmp # ls

Dockerfile

/tmp # mkdir Ishita

/tmp # exit

$ls

Output: Dockerfile Ishita

$docker network ls

$docker network create Ishita

$docker network ls

$cd /var/lib/docker/

$cd cont

$cd co