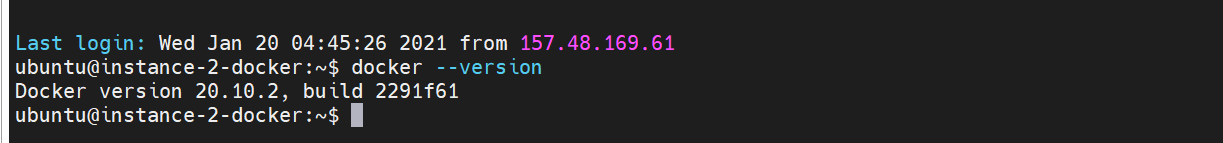
***DOCKER ASSIGNMENT***

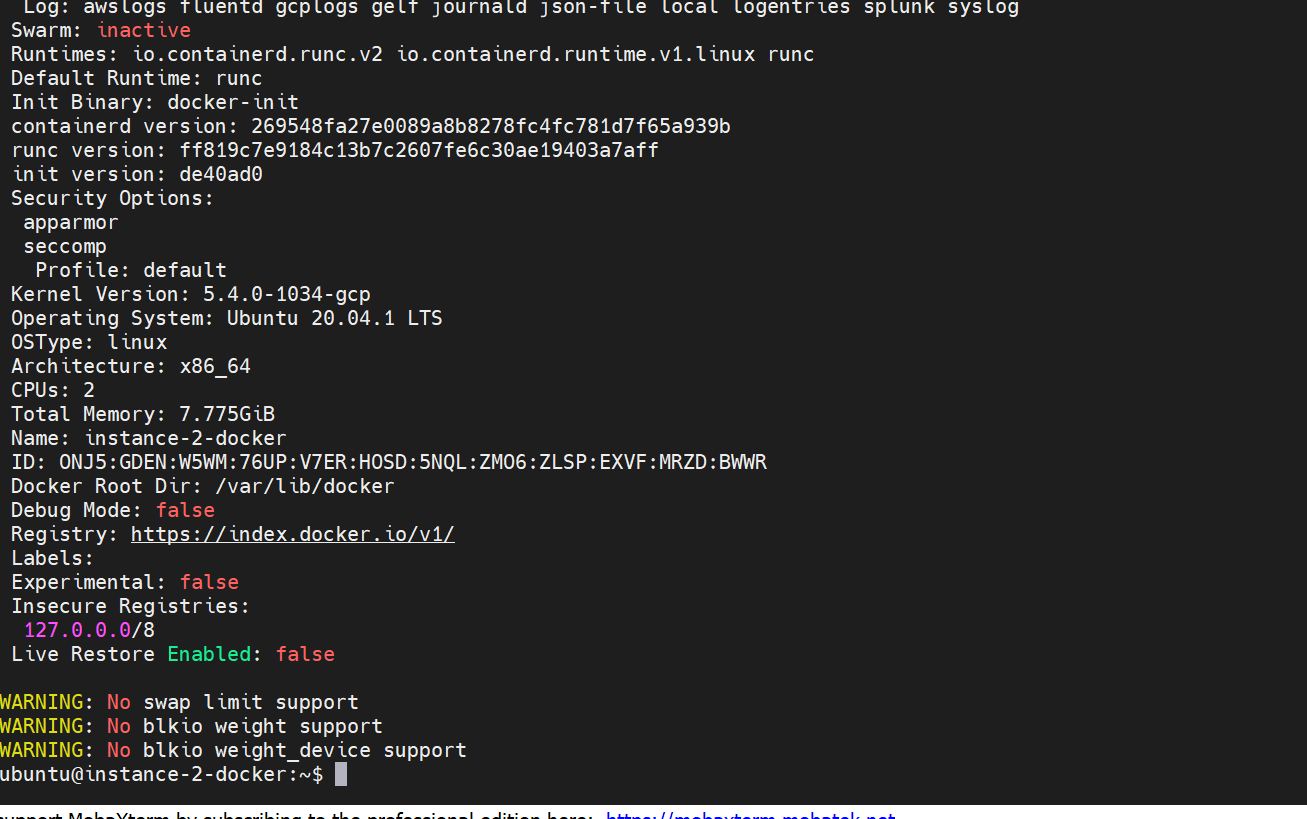
1. Verify Docker Version

Docker –version



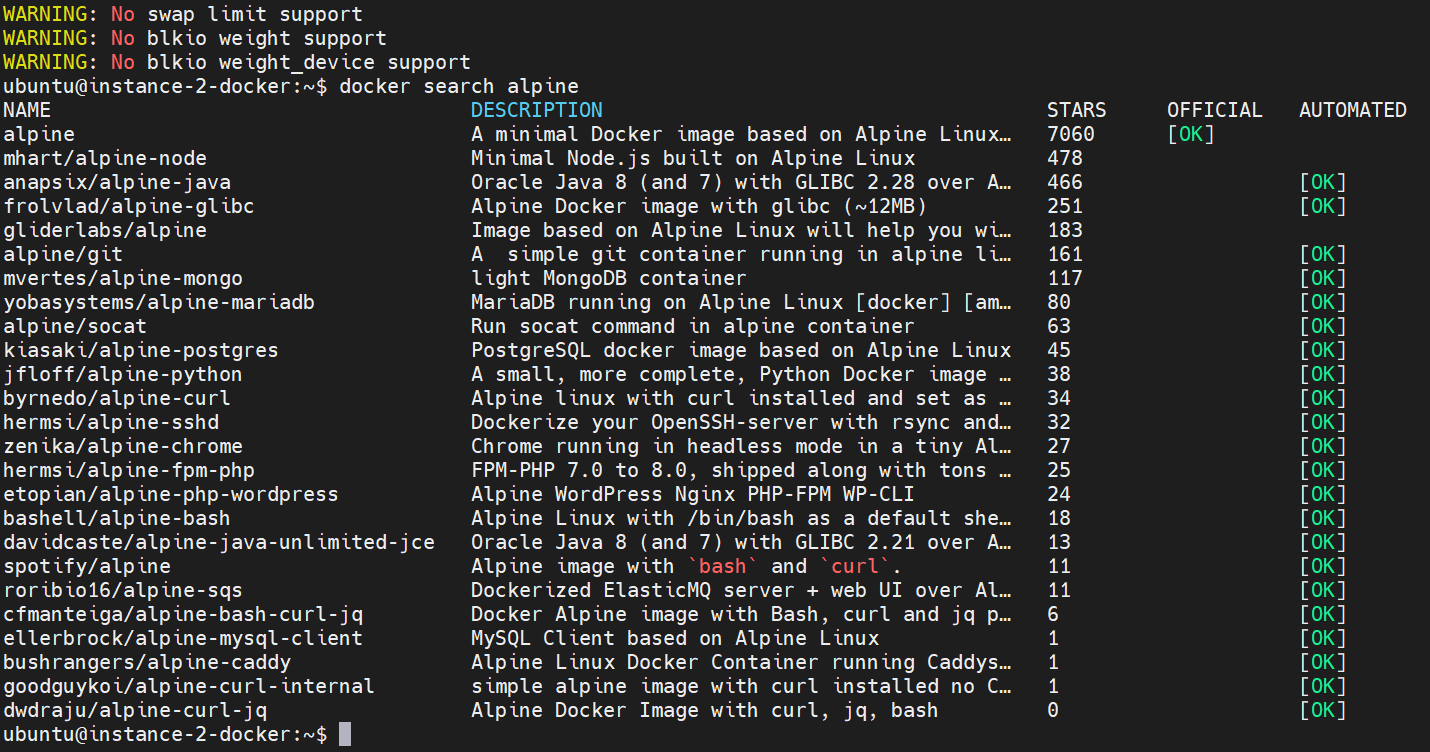
1. View system wide Information

Docker info



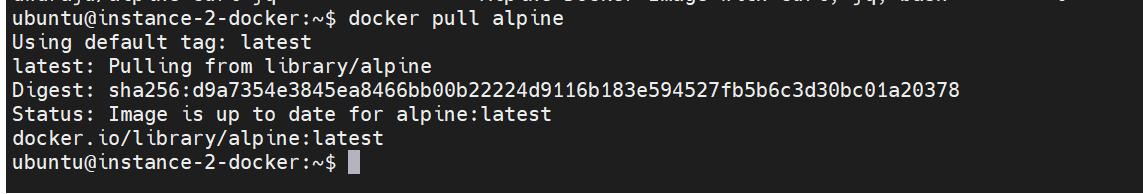
1. Search Docker Images in its Registry Server – alpine

Docker search alpine



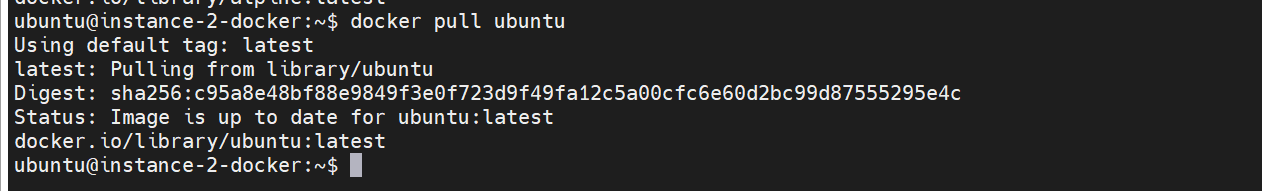
1. Download the latest version of Image alpine

Docker pull alpine



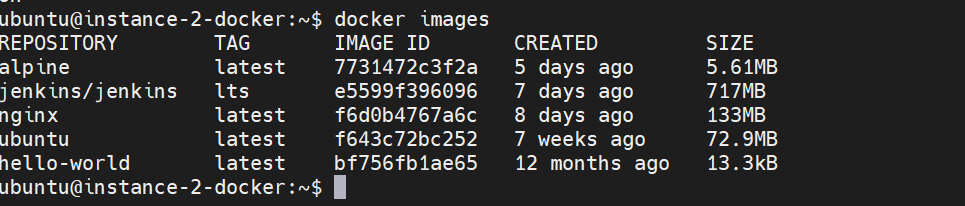
1. Download Ubuntu:14.04 Docker Image

Docker pull Ubuntu



1. List all Downloaded Docker Container Images

Docker images



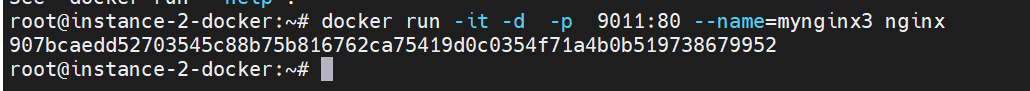
7. spinup a docker container for nginx and name it as mynginx1 and

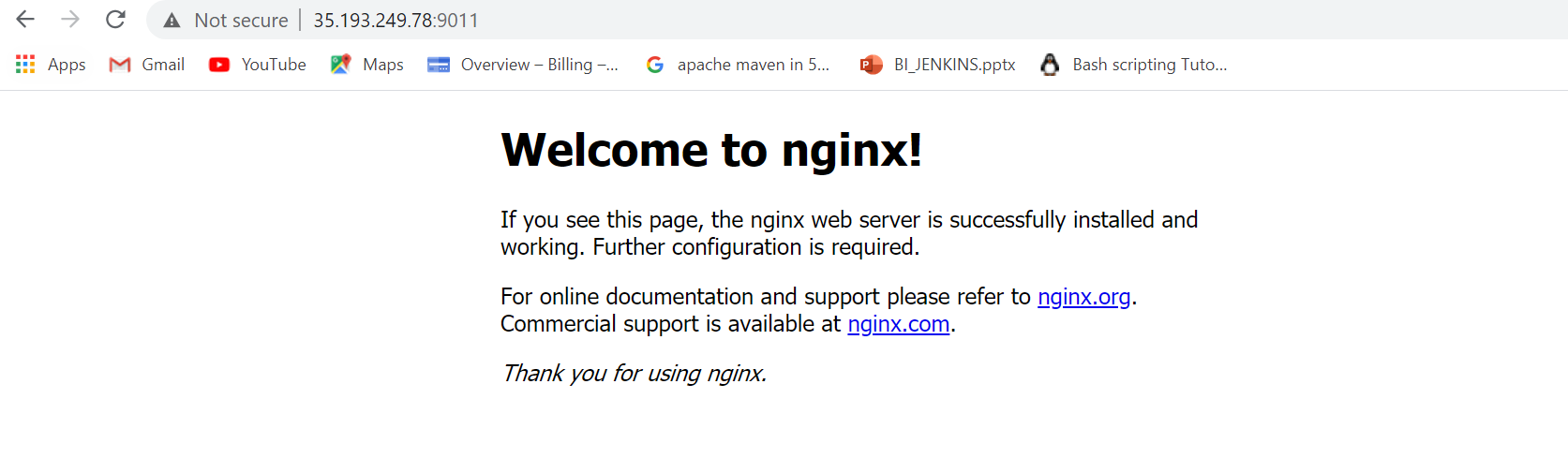
   set a rule in such a way that if any request comes to 9010 port on docker host then that request will be redirected to mynginx1 container on 80 port.



8. spinup a docker container in detached mode for nginx and name it as mynginx2 and

   set a rule in such a way that if any request comes to 9011 port on docker host then that request will be redirected to mynginx2 container on 80 port and should

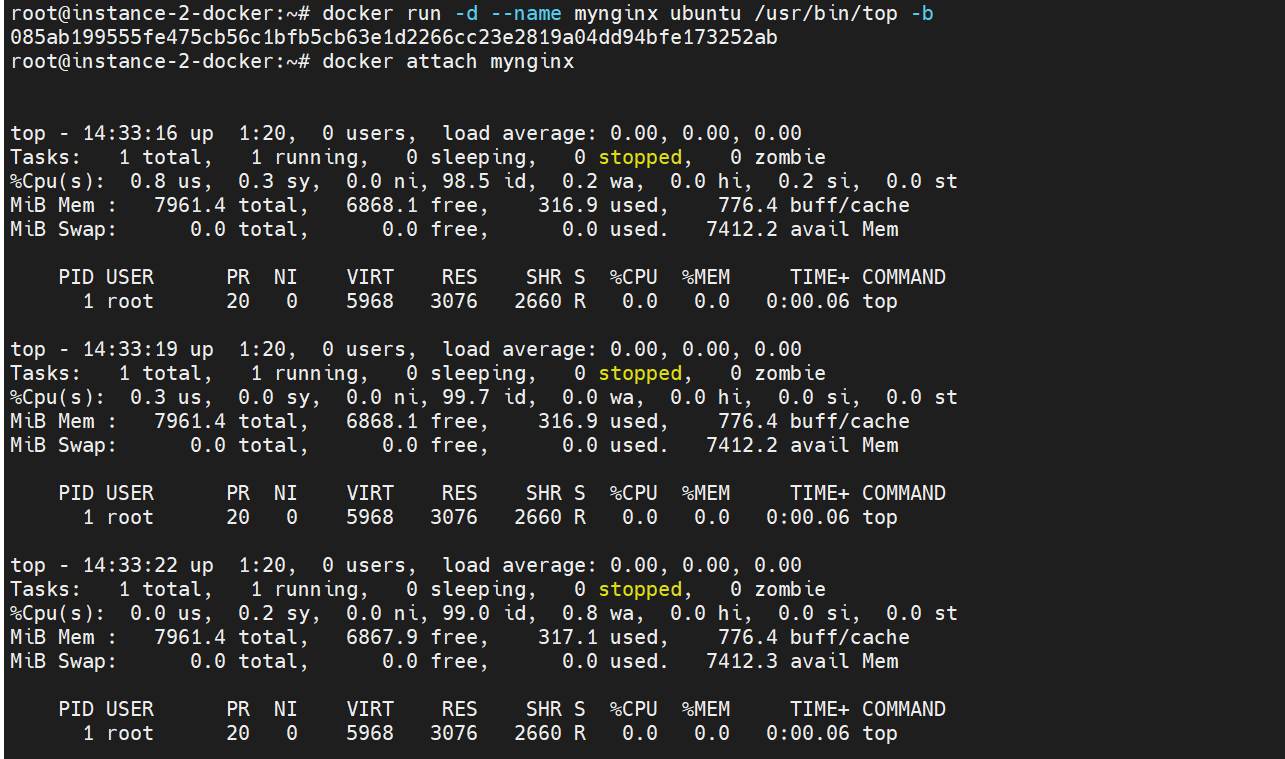




9. In the above example we have provisioned a container in detach mode, we can have its console using ‘docker attach’ command.

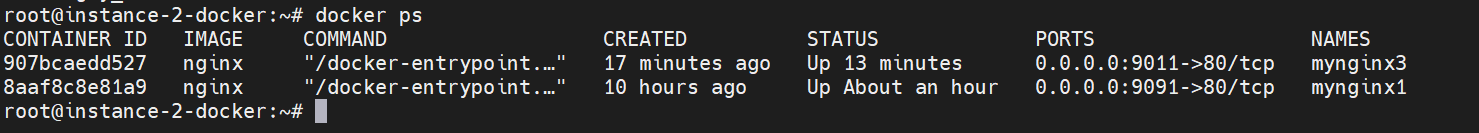
Docker run –d –name myngnix Ubuntu /usr/bin/top –b

Docker attach mynginx



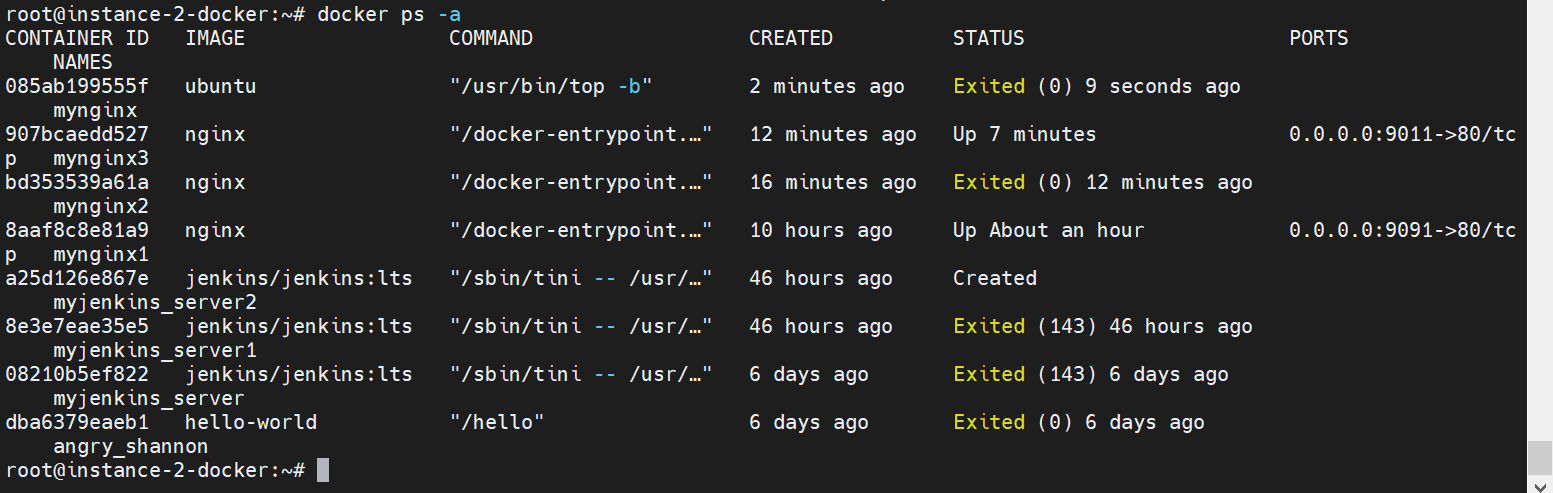
10.List all the running Containers.

Docker ps



11.List all the containers including stopped /shutdown containers.

Docker ps -a



12.pull a image named myalpine1 for alpine image and do following actions:

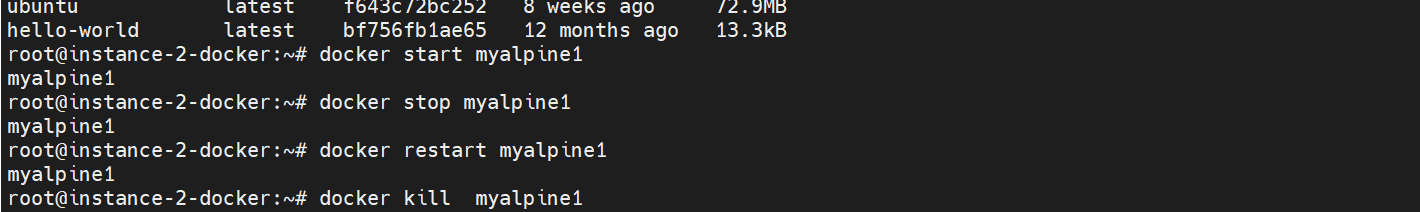
docker run --name=myalpine1 alpine

a) start the container

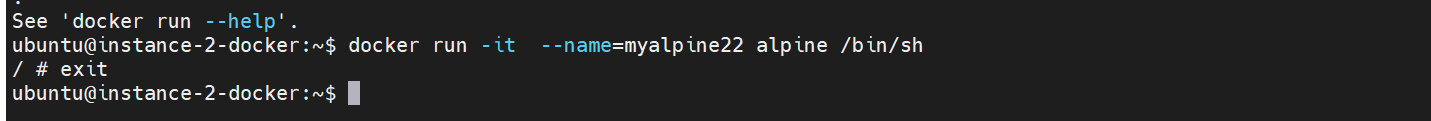
b) stop the container

c) restart the container

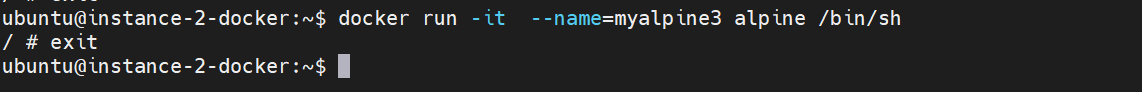
d) kill the container



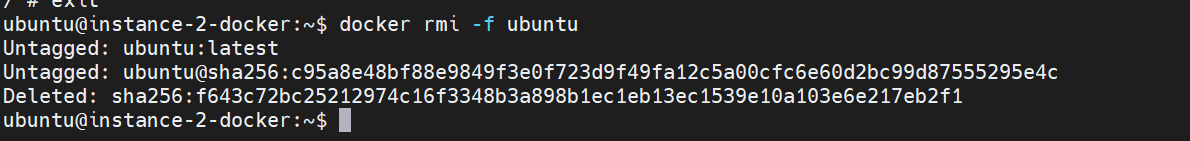
13. Remove a container name myapline2



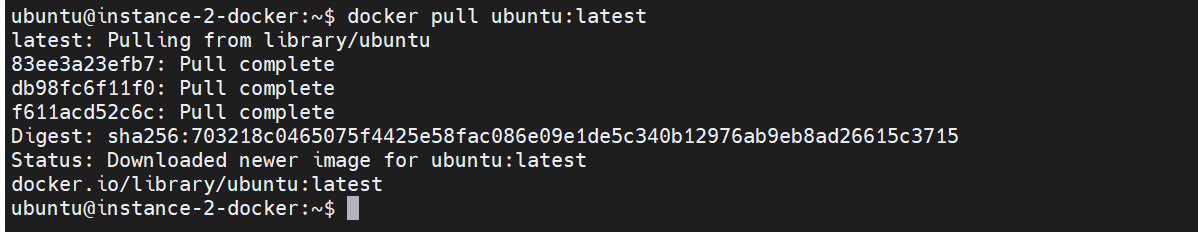
14. Remove a running container forcefully named myalpine3



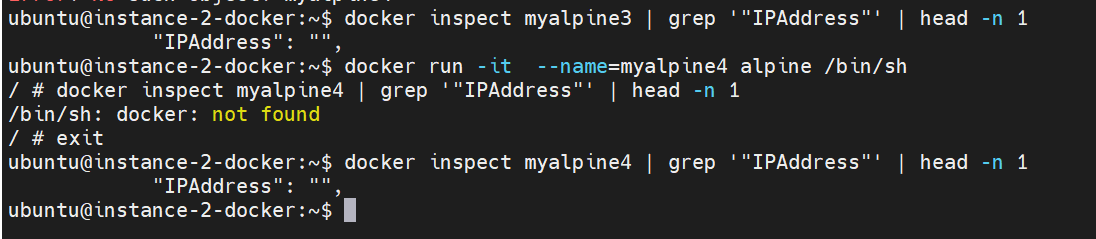
15. Delete the image ‘Ubuntu:14.04’ which you have earlier pulled/downloaded.



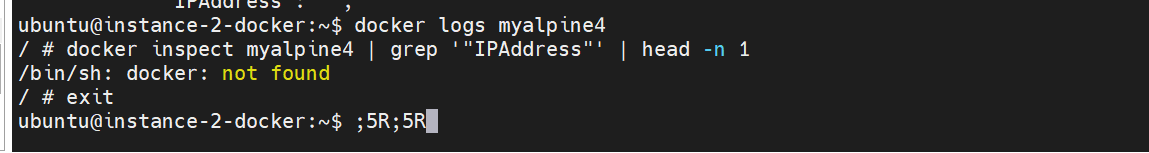
16. Download the ubuntu latest image and check the history



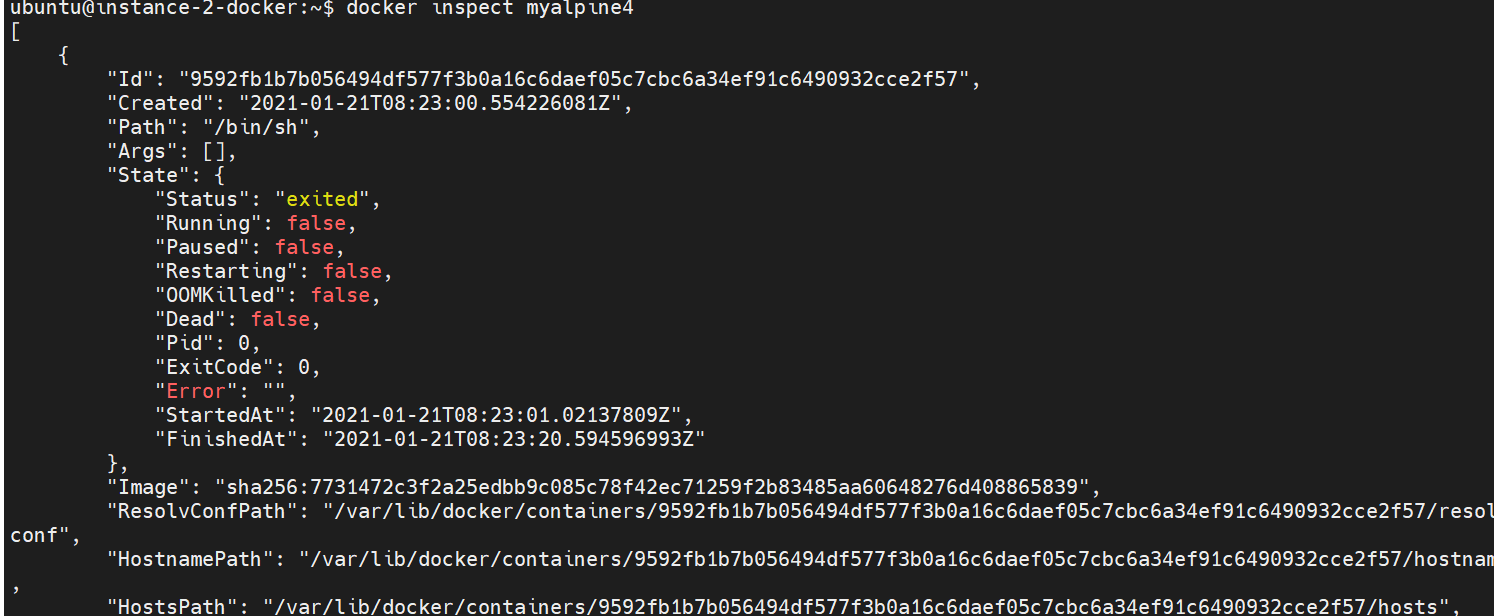
17. Display Container IP address for myalpine4



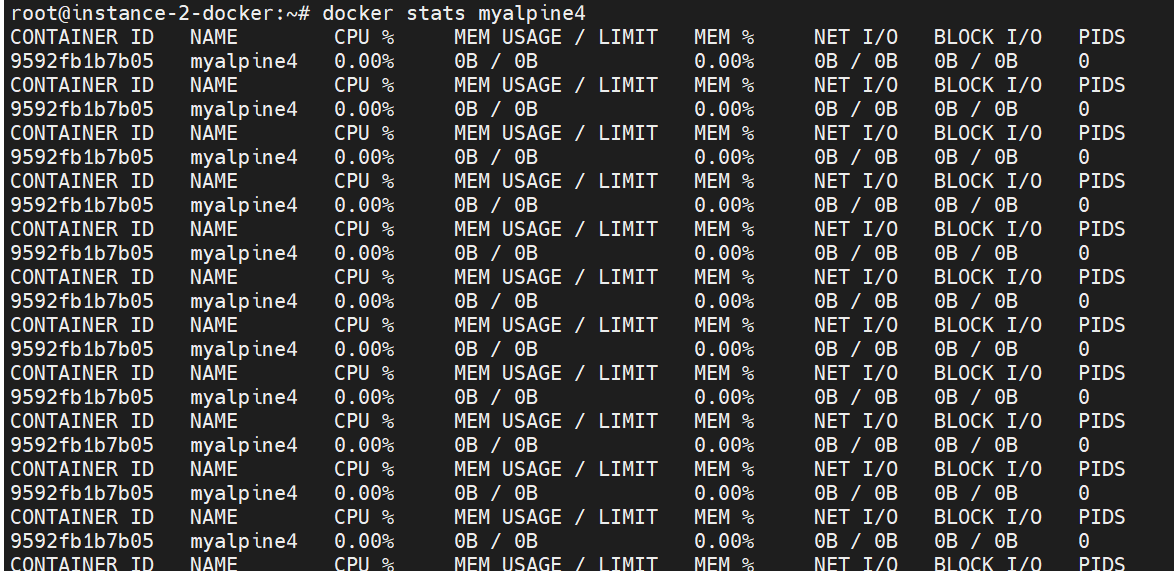
18. Display the logs for container myalpine4

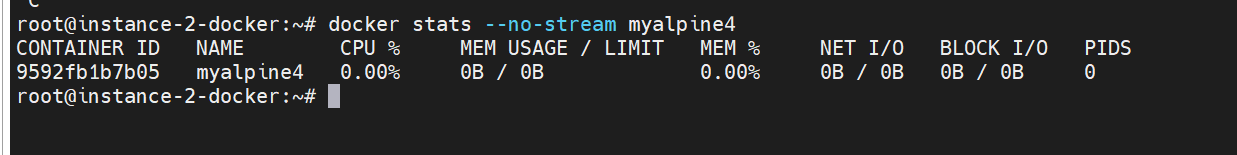


19. Inspect the container myalpine4

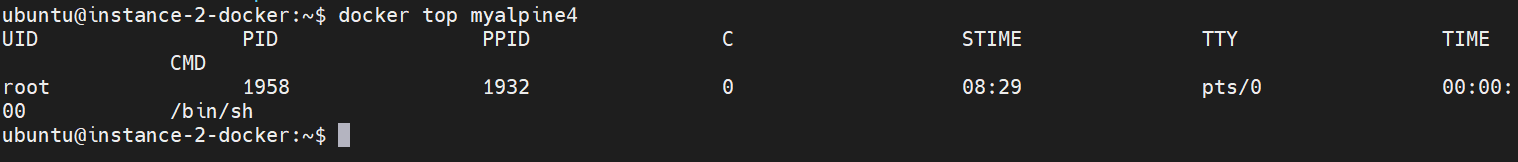


20. Display the stats for myalpine4 (live streaming /non-live streaming)



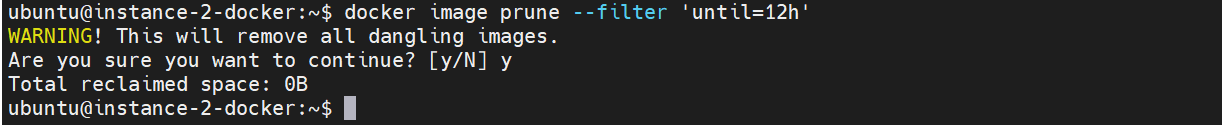


21. Display the running processes of a container with ‘docker top‘ command.



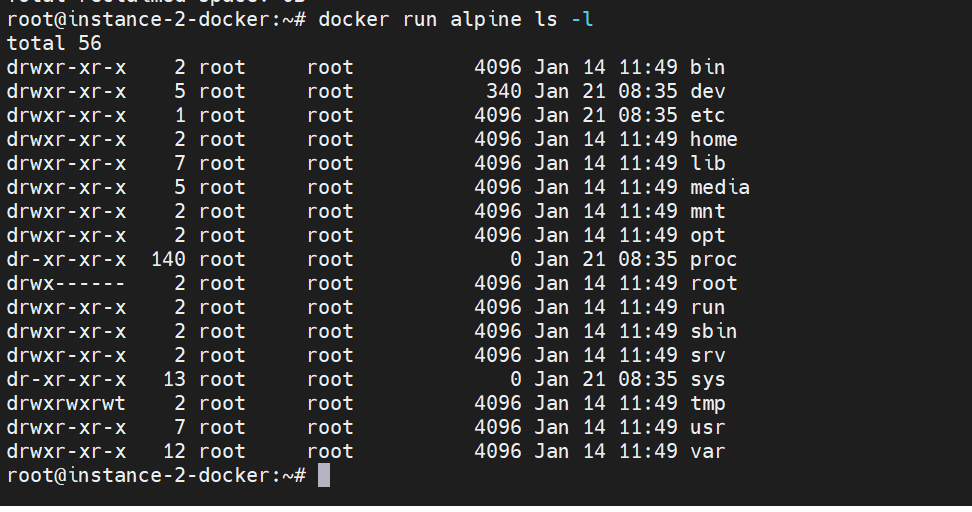
22. Remove all images that are created more than 12 hours ago

docker stats --no-stream myalpine4



23. Spinup containers and execute the below

docker run alpine ls –l



docker run alpine echo "Hello"

docker run alpine /bin/sh

docker run -it alpine /bin/sh

/ # date

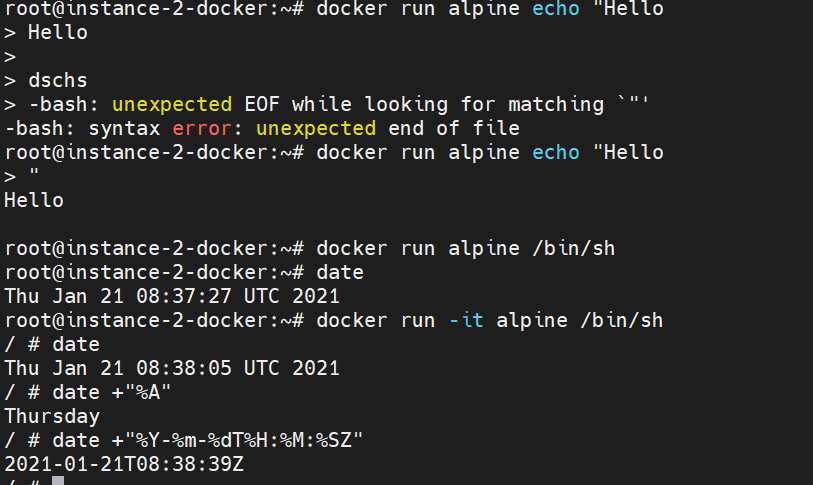
Sat May 30 18:57:24 UTC 2015

/ # date +"%A"

Saturday

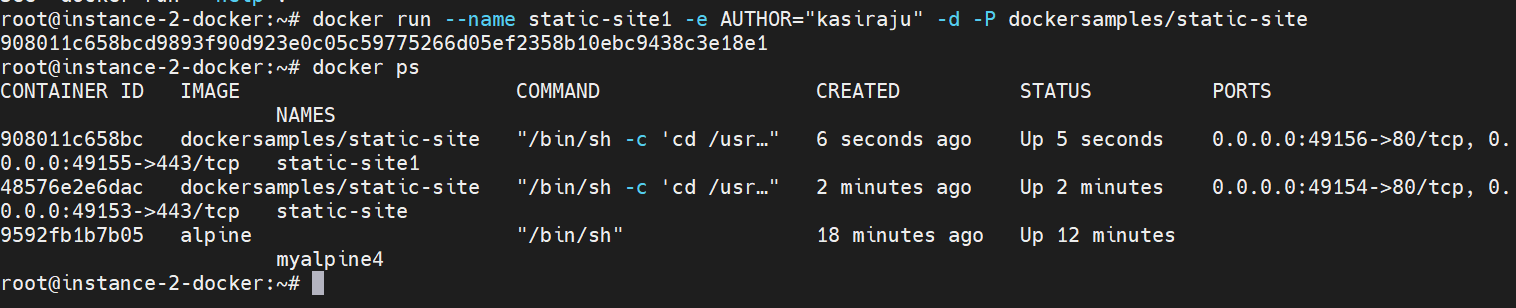
/ # date +"%Y-%m-%dT%H:%M:%SZ"

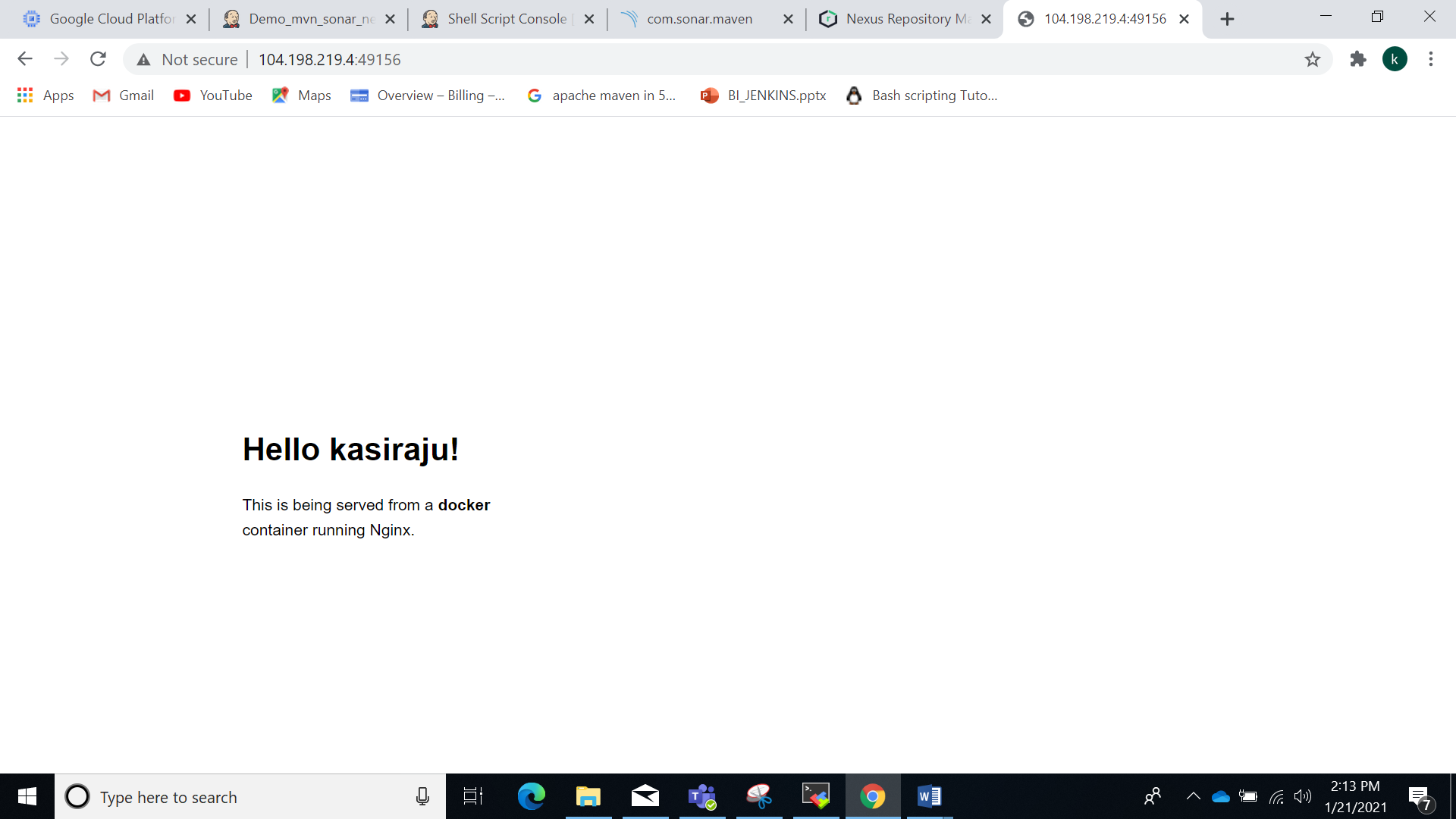
2015-05-30T19:00:38Z



24. Execute the below and publish the ports on which its running.

docker run --name static-site -e AUTHOR="kasiraju" -d -P dockersamples/static-site

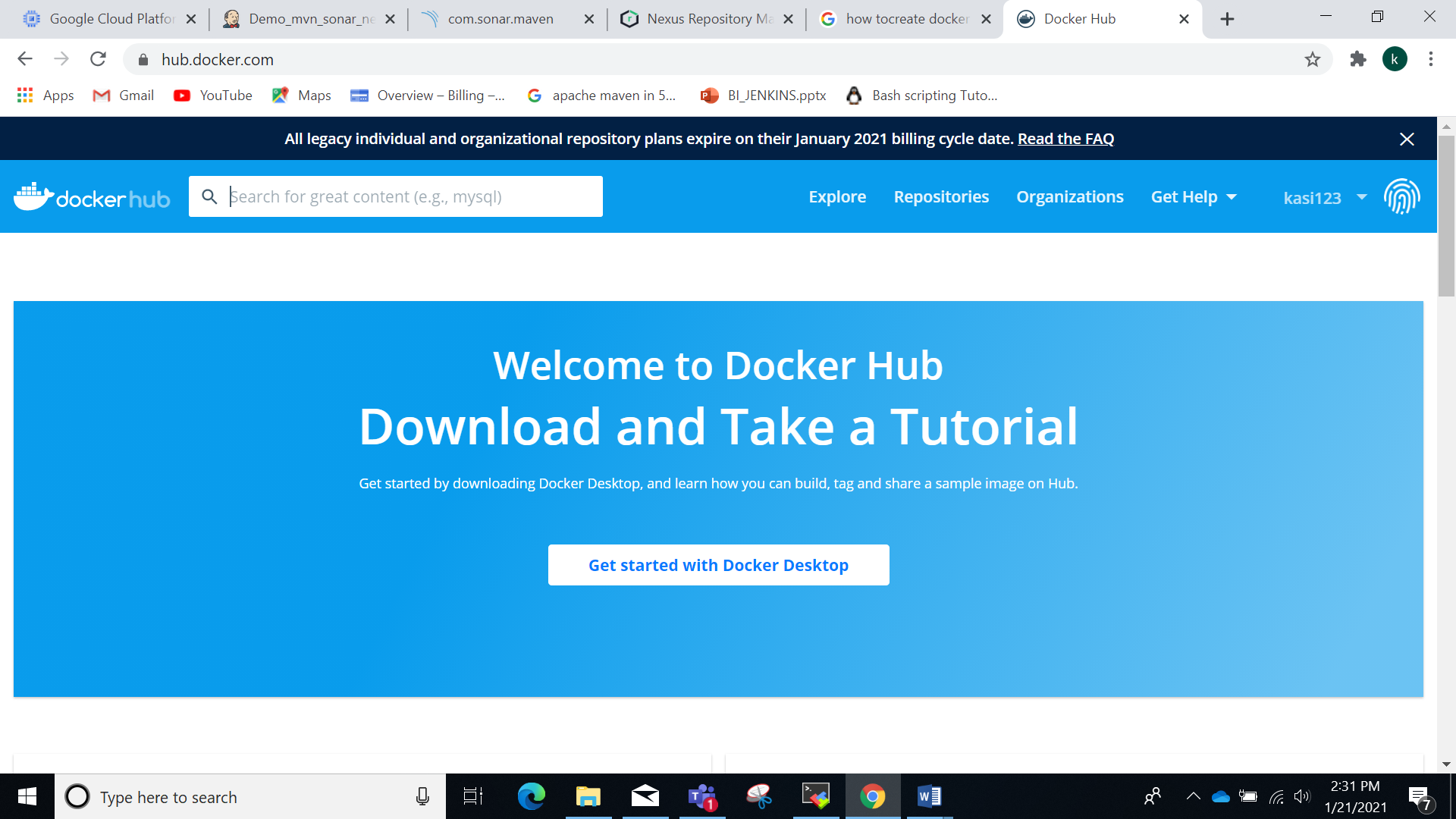




25. Stop all running containers with a single command.



26. Create a docker hub account prior to exercise 27.



27. Build a custom docker image :

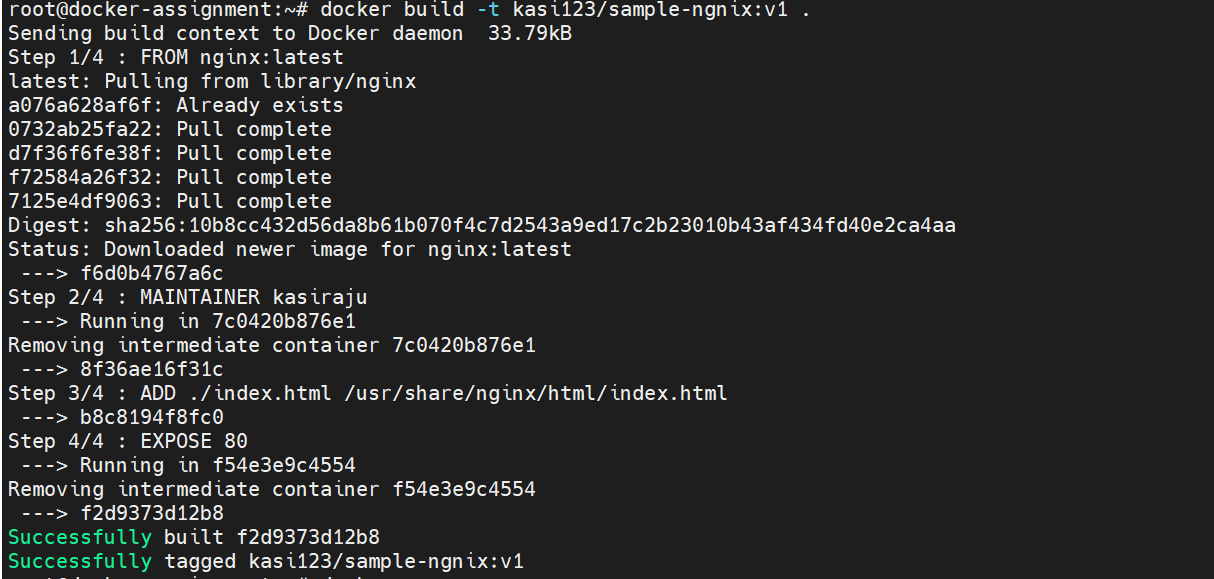
a) create Dockerfile :

FROM nginx:latest

MAINTAINER kasiraju

ADD ./index.html /usr/share/nginx/html/index.html

EXPOSE 80

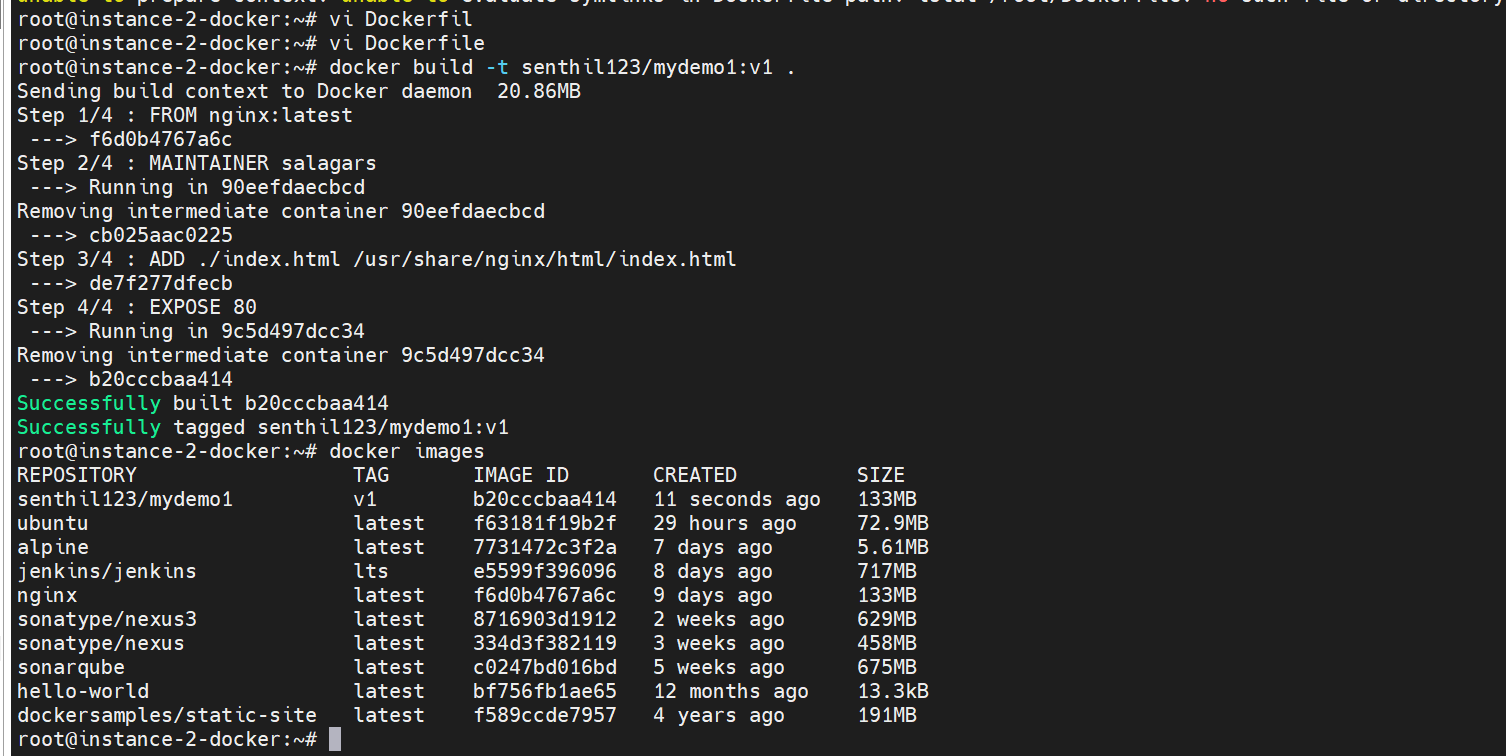


b) create index.html :

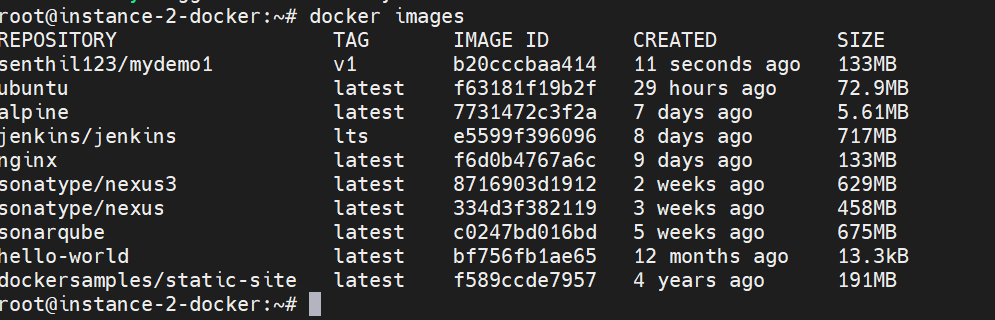
<h1> welcome to Dockerizing apps! <h1>

c) create the custom image and observe the intermediate layers.

$ docker build -t Kasi123/mydemo1:v1 .

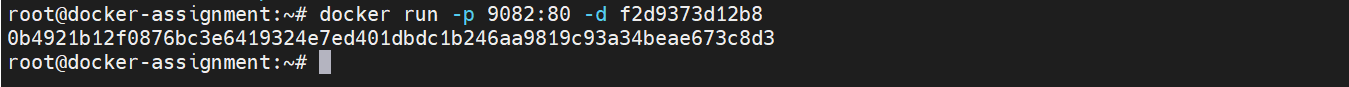


d) $ docker images

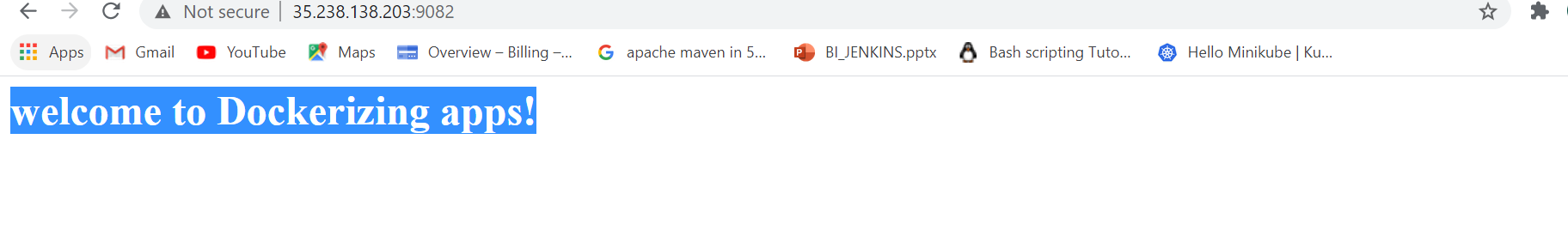


e) $ docker run -p 9082:80 -d image-id

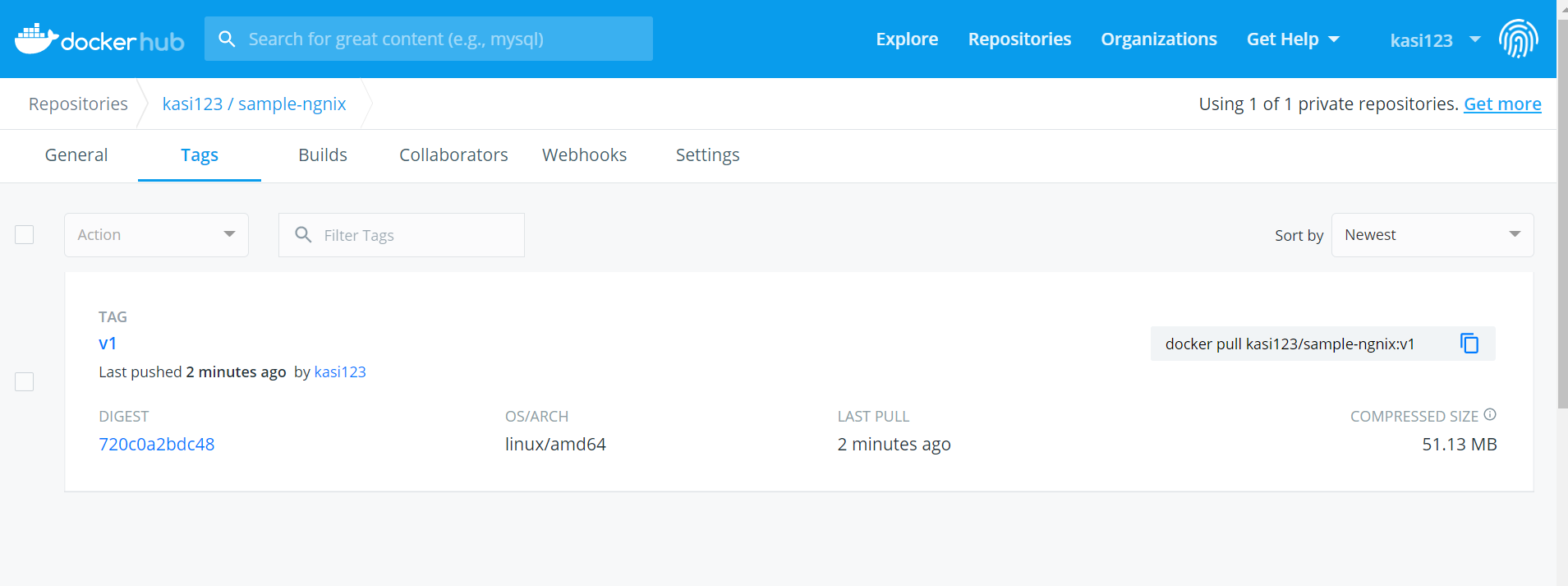
curl 127.0.0.1:9082 or curl private\_ip:9082



f) run the application on browser



g) push the custom image to your docker hub repository



Docker login –username=kasi123

@December5@

