***Docker Container***

# Introduction

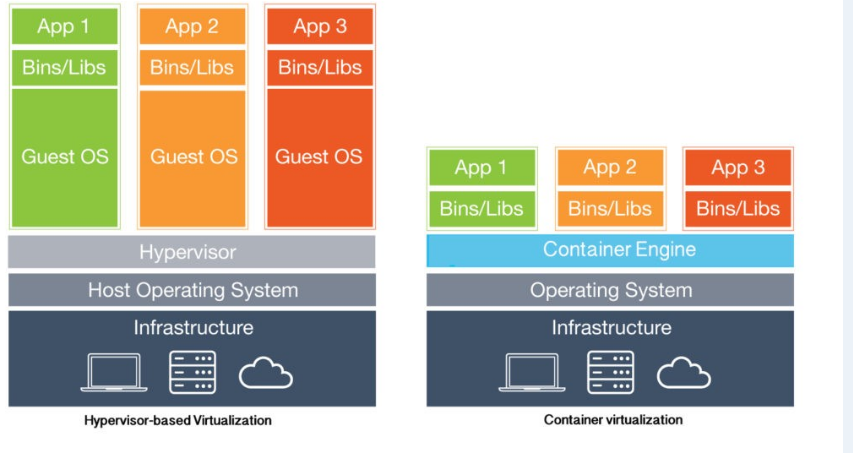
***What is Virtualisation?***

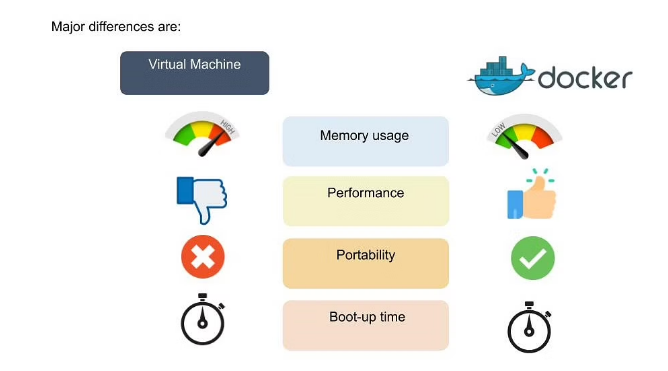
Virtualization is the fundamental technology that powers cloud computing. Virtualization is software that manipulates hardware, while cloud computing refers to a service that results from that manipulation. You can’t have cloud computing without virtualization.

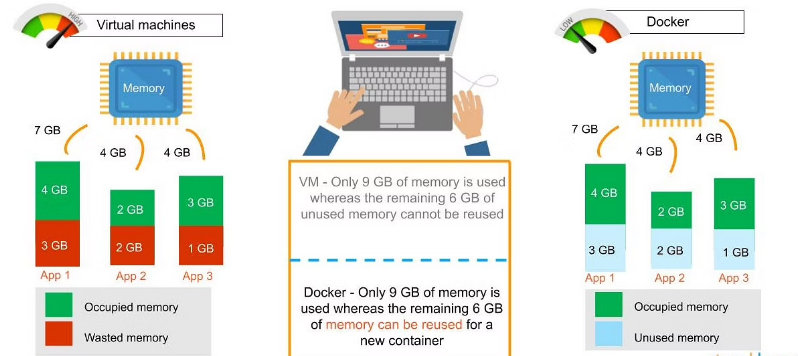
***What Is Container?***

Containers are lightweight software packages that contain all the dependencies required to execute the contained software application. These dependencies include things like system libraries, external third-party code packages, and other operating system level applications. The dependencies included in a container exist in stack levels that are higher than the operating system.

***Virtual Machine vs Containers?***

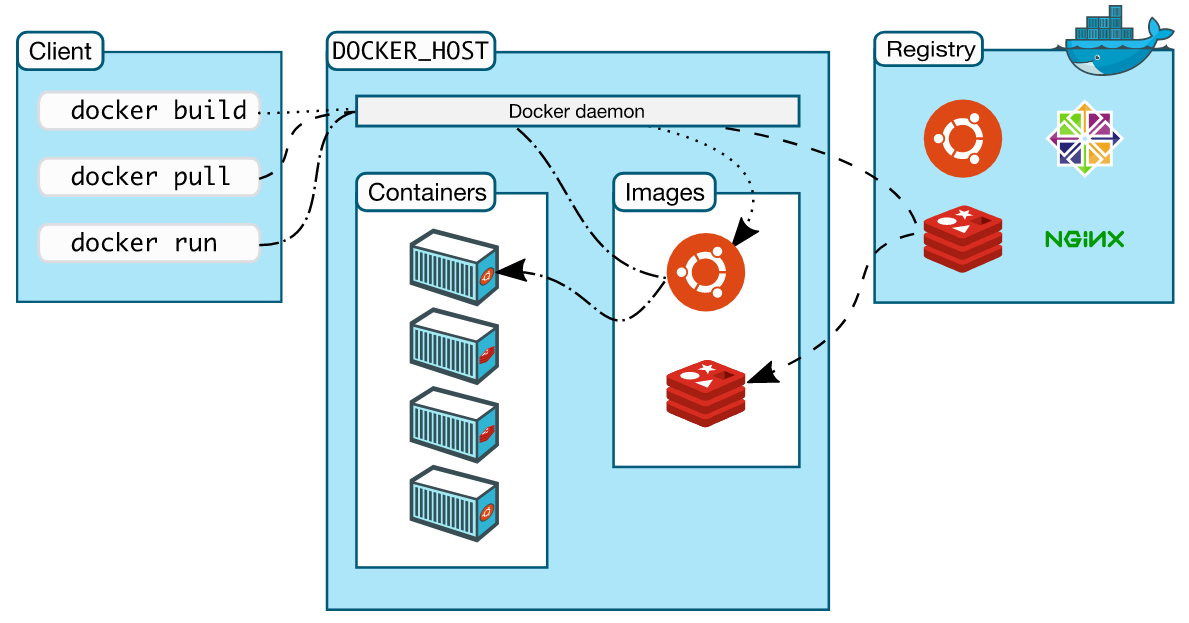






***Docker Components?***

Docker uses a ***client-server*** architecture. The Docker ***client*** talks to the Docker ***daemon***, which does the heavy lifting of building, running, and distributing your Docker containers. The Docker client and daemon *can* run on the same system, or you can connect a Docker client to a remote Docker daemon



***The Docker daemon***

The Docker daemon (dockerd) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes. A daemon can also communicate with other daemons to manage Docker services.

***The Docker client***

The Docker client (docker) is the primary way that many Docker users interact with Docker. When you use commands such as docker run, the client sends these commands to dockerd, which carries them out. The docker command uses the Docker API. The Docker client can communicate with more than one daemon.

***Docker registries***

A Docker registry stores Docker images. Docker Hub is a public registry that anyone can use, and Docker is configured to look for images on Docker Hub by default. You can even run your own private registry.

When you use the docker pull or docker run commands, the required images are pulled from your configured registry. When you use the docker push command, your image is pushed to your configured registry.

# Installation

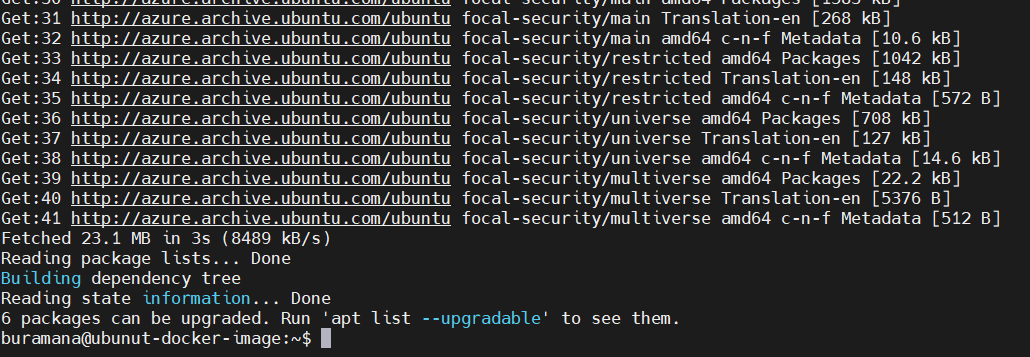
## On ubuntu:

If you want install latest version of docker run the below commands

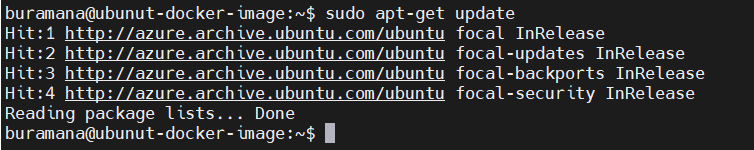
To avoid permission issues while executing commands run ***“sudo su”***

**apt update** - to update the repositories

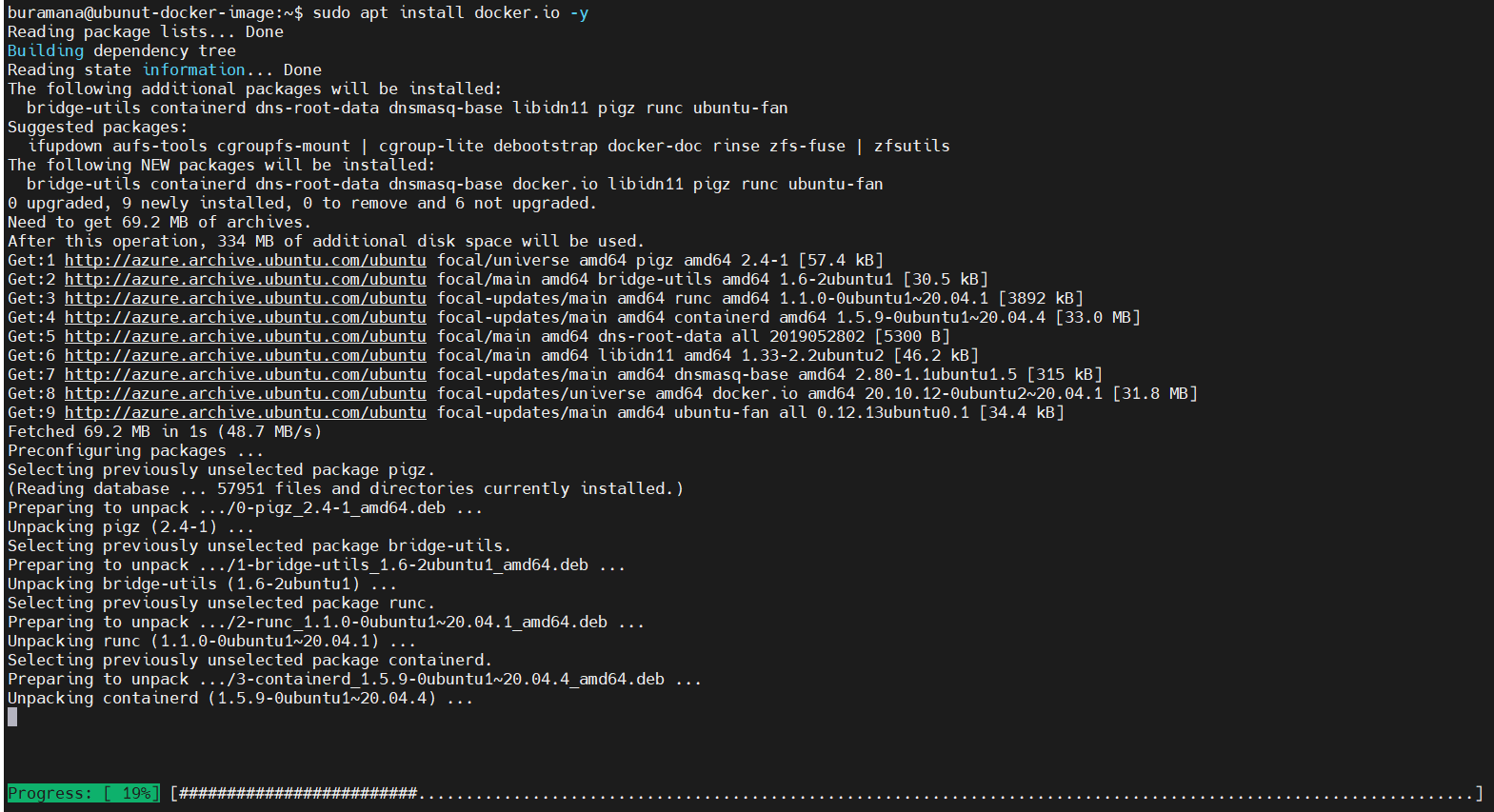


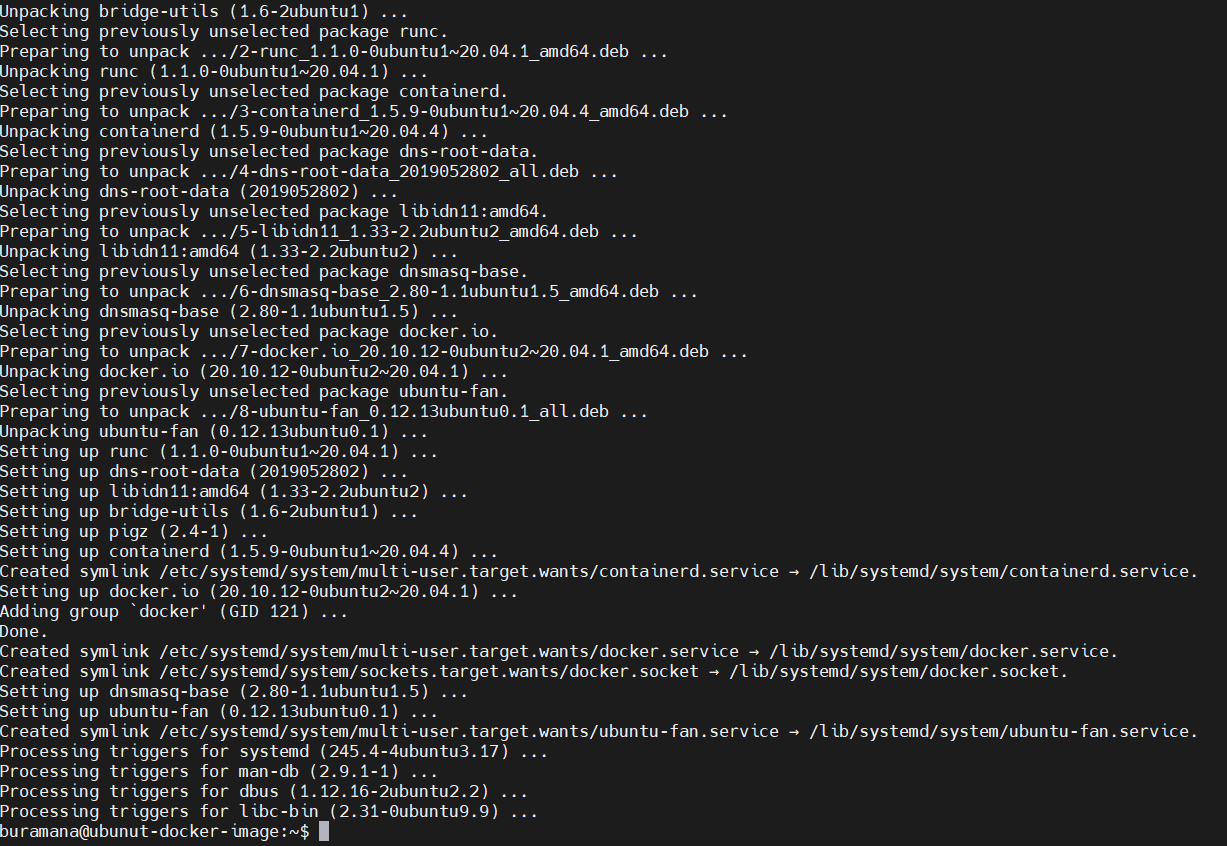


**apt-get update –** to update apt-get repo as well

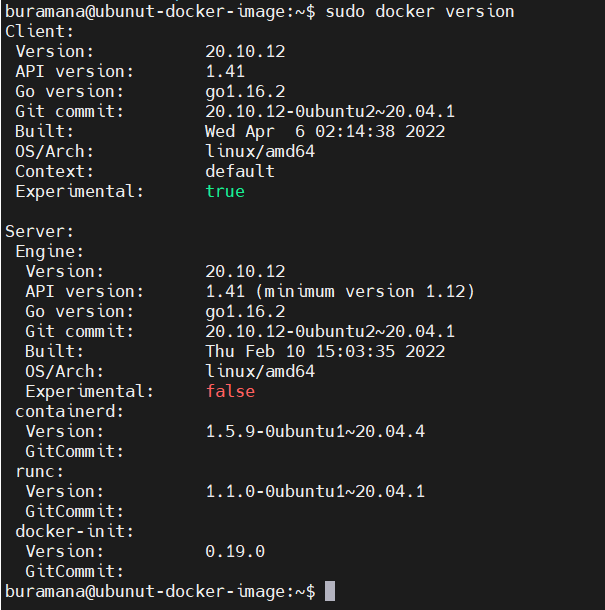


**apt install docker.io -y** – To install docker engine

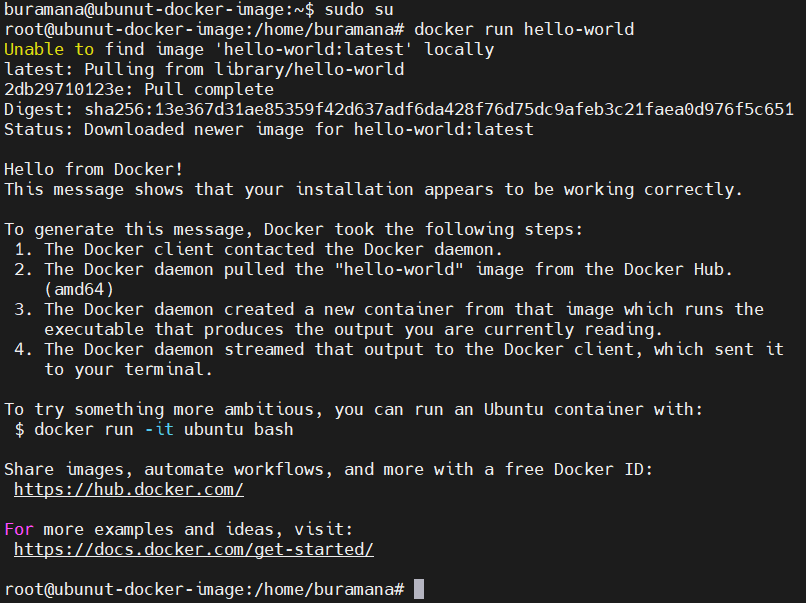




***docker version*** – to verify version of docker engine



***docker run hello-world*** – run simple hello-world container to make sure docker is working properly.



## On Windows

|  |  |
| --- | --- |
| **Steps** | **screenshot** |
| Login to the windows machine and download the Docker desktop here is the url [Docker Desktop](https://desktop.docker.com/win/main/amd64/Docker%20Desktop%20Installer.exe?utm_source=docker&utm_medium=webreferral&utm_campaign=dd-smartbutton&utm_location=header)  Once download completed, file will be available in Downloads folder or default download location. |  |
| Double click on the installer to start the Docker installation. |  |
| It will asks to create icon on desktop. Click OK. |  |
| Wait till installation completed. Installation succeeded screen click on close. Installation is completed. |  |
| After installation to start it double click on Docker Desktop icon on your machine’s desktop. |  |
| It will open start page, check the accept the terms and then click on Accept button. Ignore any warning windows or messages. |  |
| Now it will restart the docker service |  |
| Once service got restarted. Getting started page will open. Either you start tutorial by click on start or click on skip tutorial to go main window. |  |
| Here I am click on Skip tutorial.  Here is the Home page. |  |
| Here we can see the running containers, if any. |  |
| In Images page , will display the local images and remote repository if logged to your private registry |  |
| In Volumes page,it will list if any volumes created. |  |
| Dev Environments will allow us to share our work with others |  |
| We can also use the power shell as command line to execute commands.  *docker version*  *docker --version* |  |
| Open services, to look at the docker service |  |

# Docker Commands

Graphical user interface, text, table

Description automatically generated

docker create <imagename>:<tag> - this command will just create container but not started.

docker create --name first-container httpd:latest



It will create the container with the name “first-container” & image name is httpd with latest tag and it will return the Container ID.

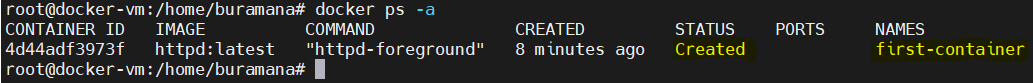
docker ps - To list the running containers

docker ps



docker ps -a – To list all containers

docker ps -a



docker start <container name or id> - to start the container and it will container id as output.

docker start first-container



Now, let check the running containers. Execute “docker ps” and observer the output.



docker stop <container name or Id> - to stop the running container

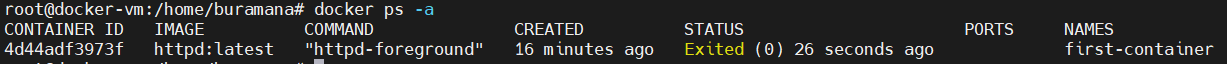
docker stop first-container



After stopping, the running container list will show empty



And all container list will have exited an running. But in this example we have only one that exited.

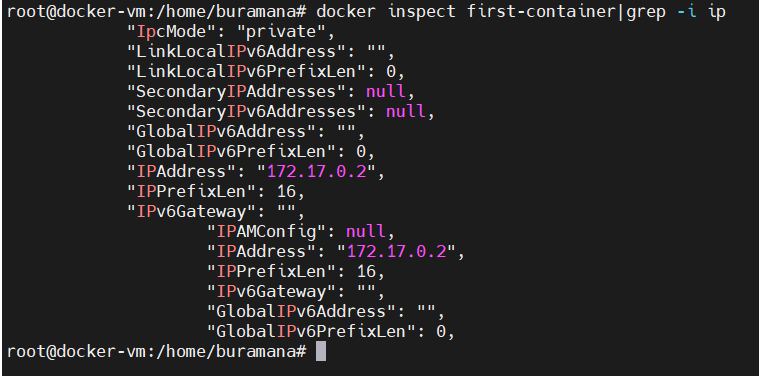


Till this we have seen creating, start and stop the containers.

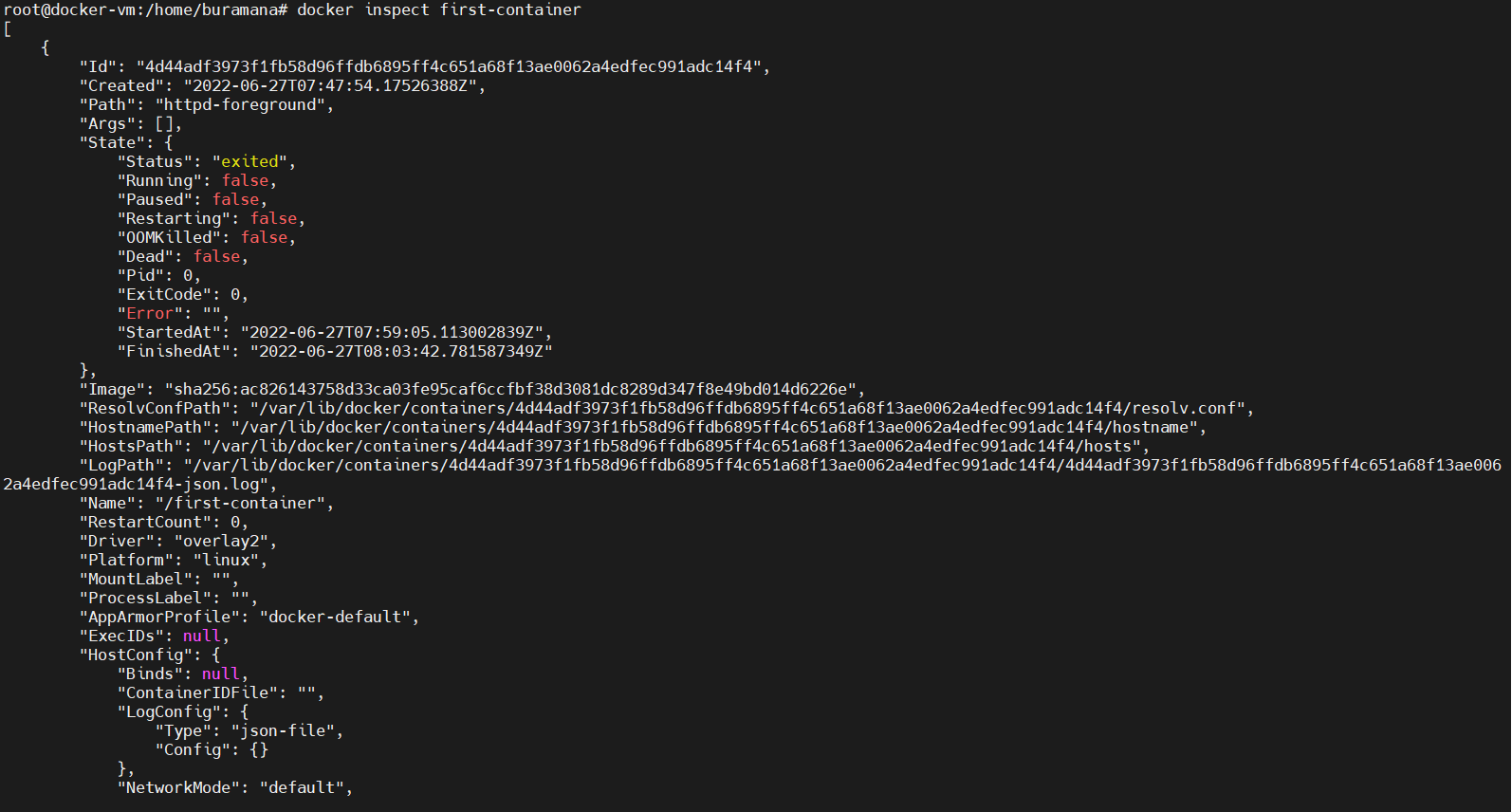
docker inspect <container-name or ID> - to get the container information

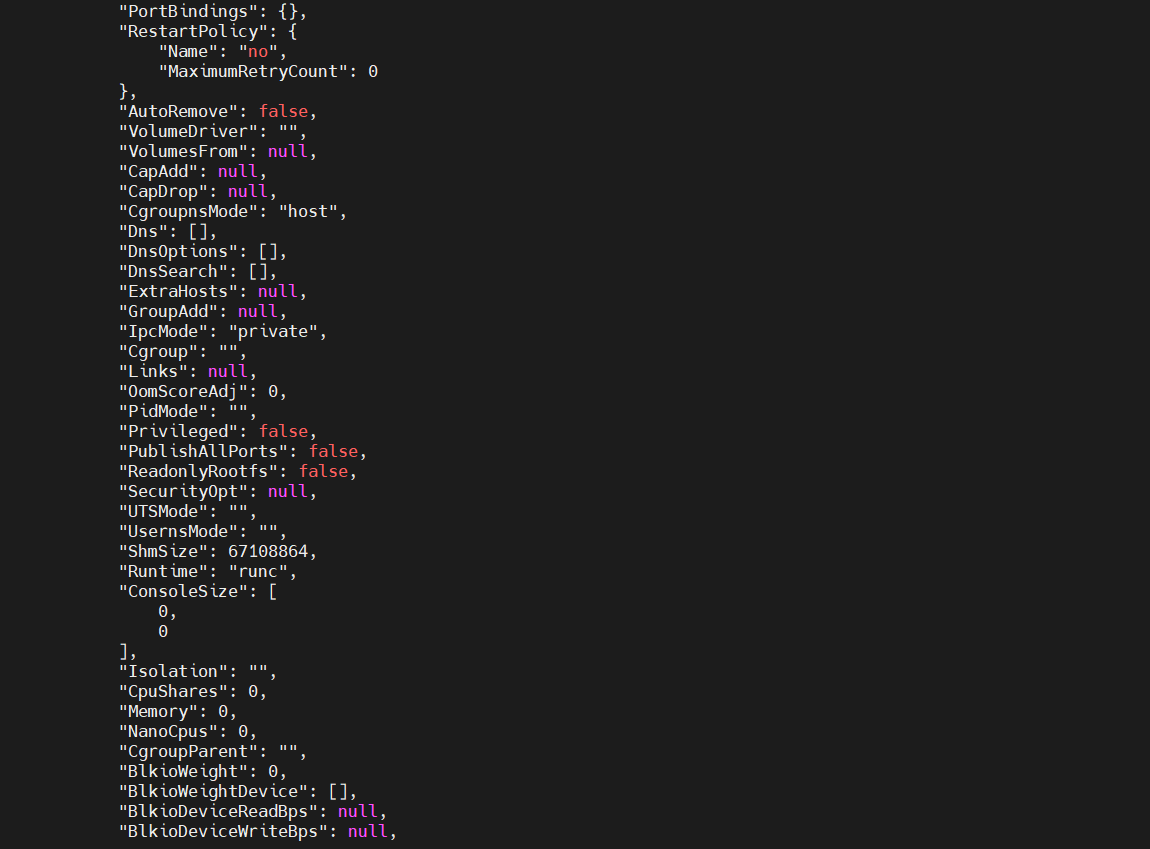
docker inspect first-container

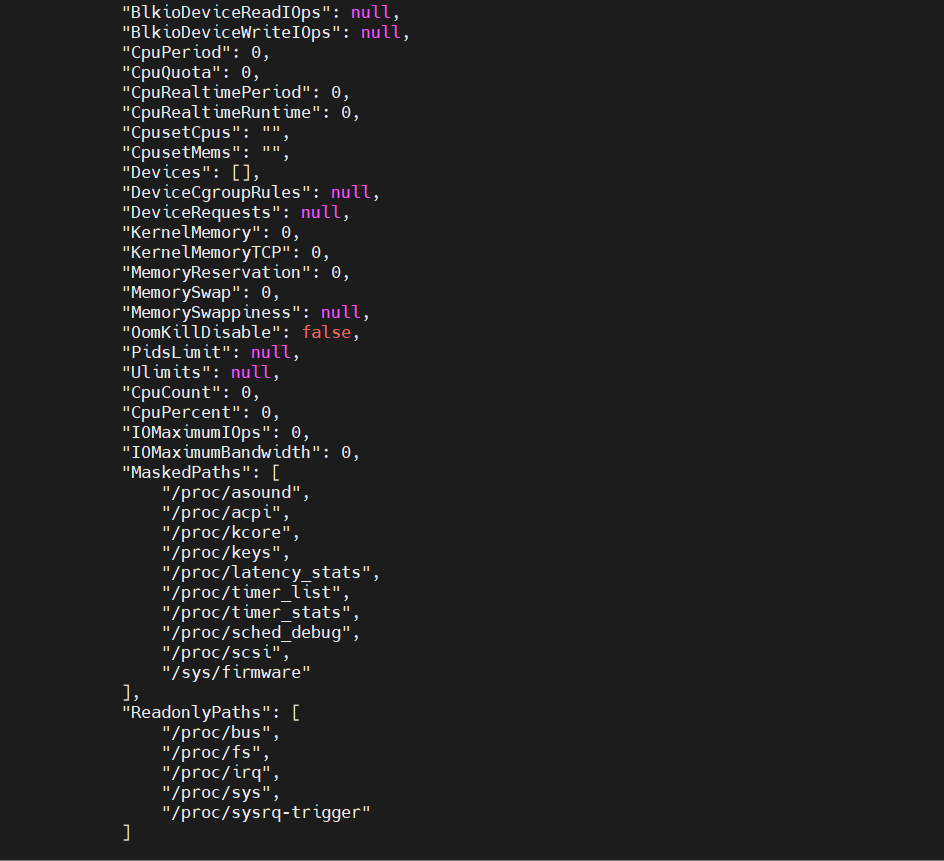
docker inspect first-container|grep -I ip

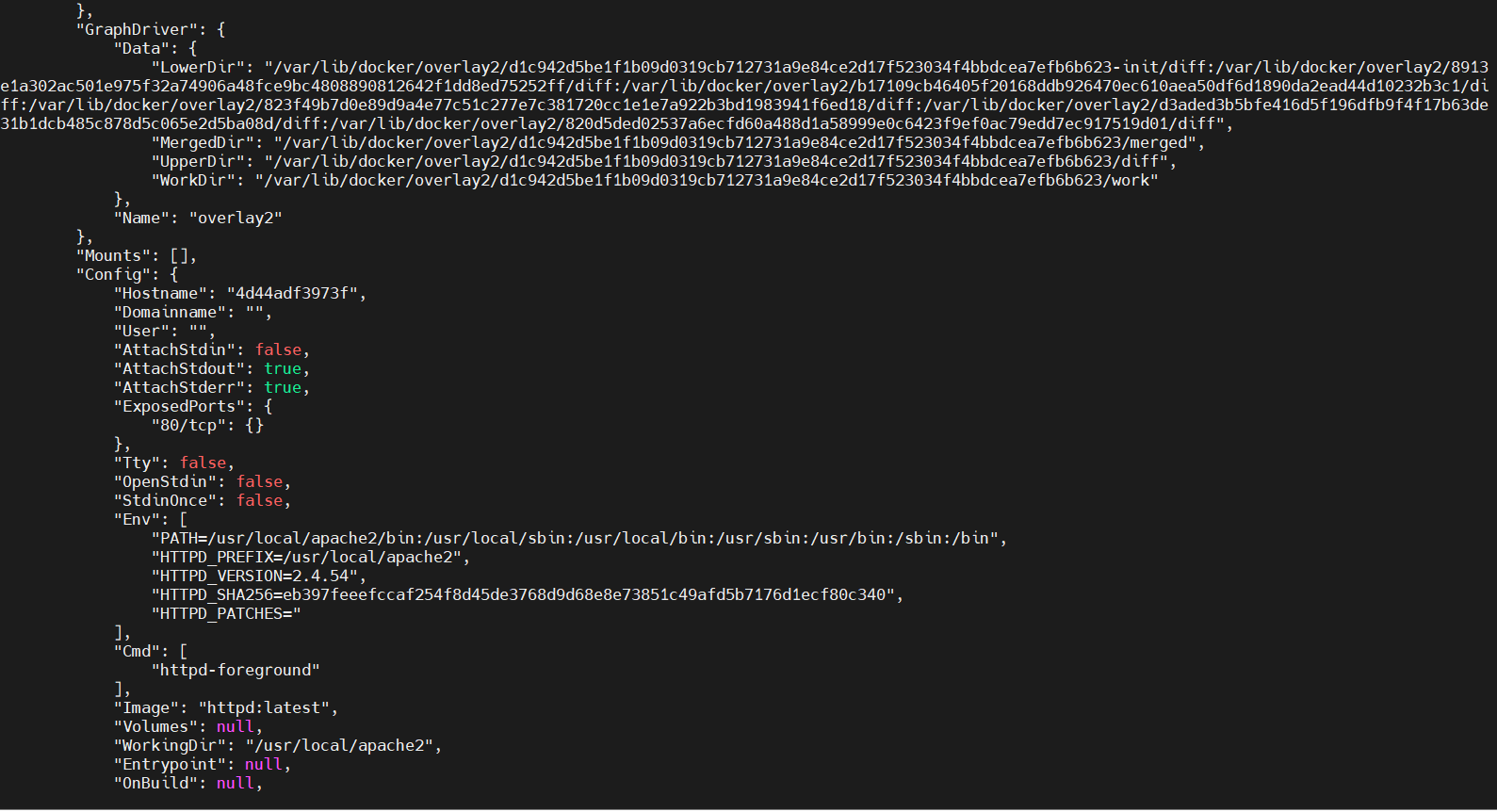


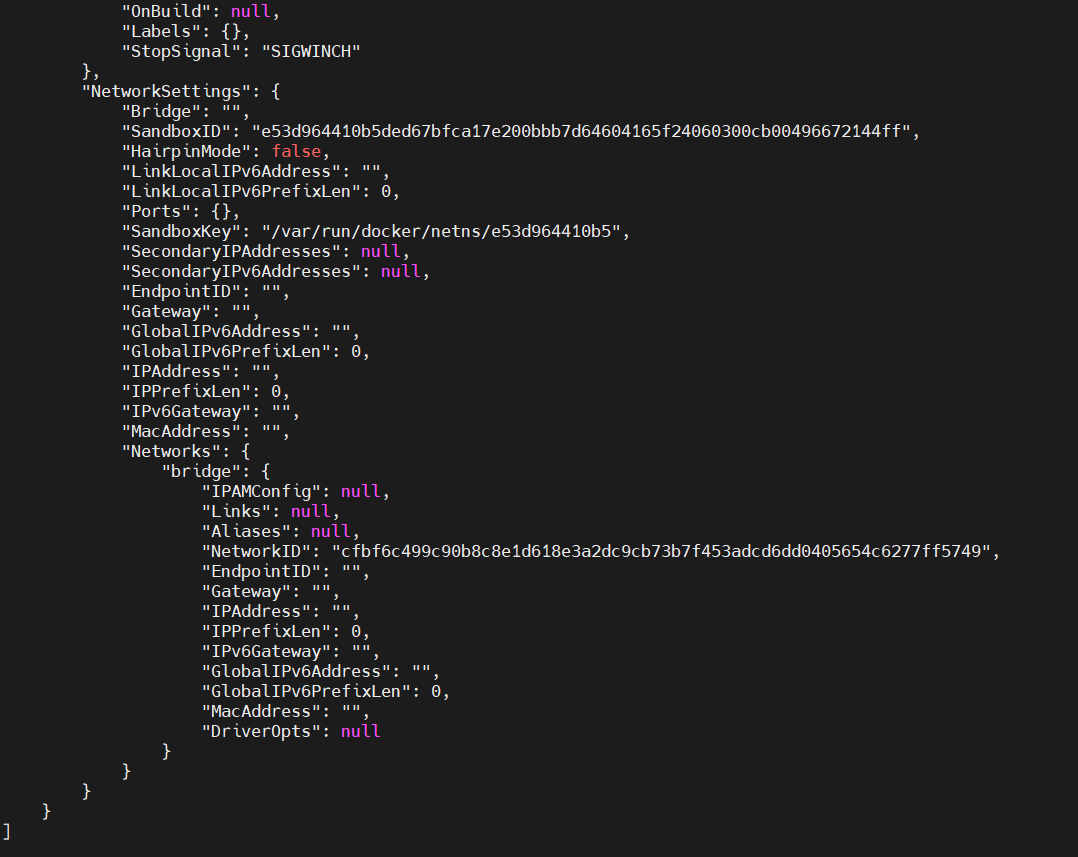
Here is the complete information about the containers





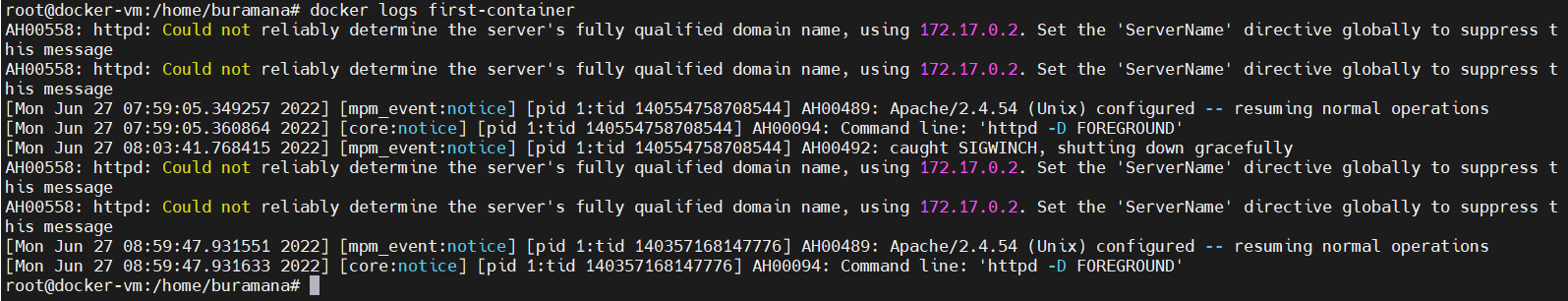






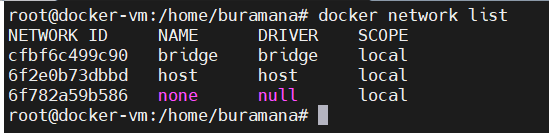
docker logs <container name or id> - to get the logs of container

docker logs first-container

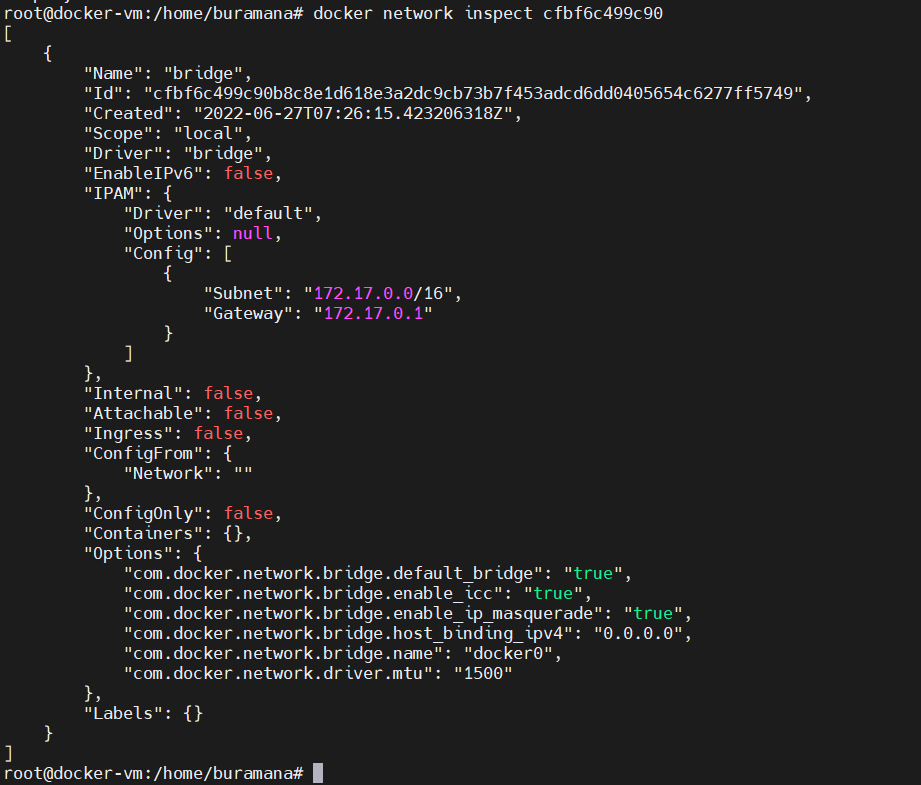


Networking:

docker network list – to list the network layers.



docker network inspect <network id> - To display the detailed information of network



docker network connect <network-id> <container id or name> - Connect the container to network.



docker network disconnect <network-id> <container id or name> - Disconnect the container to network.

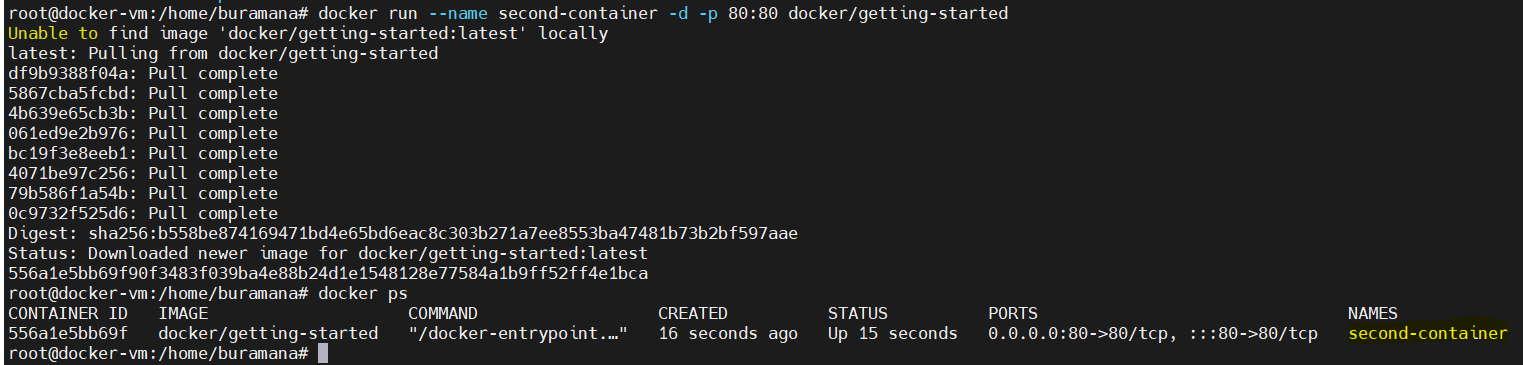


**Information:**

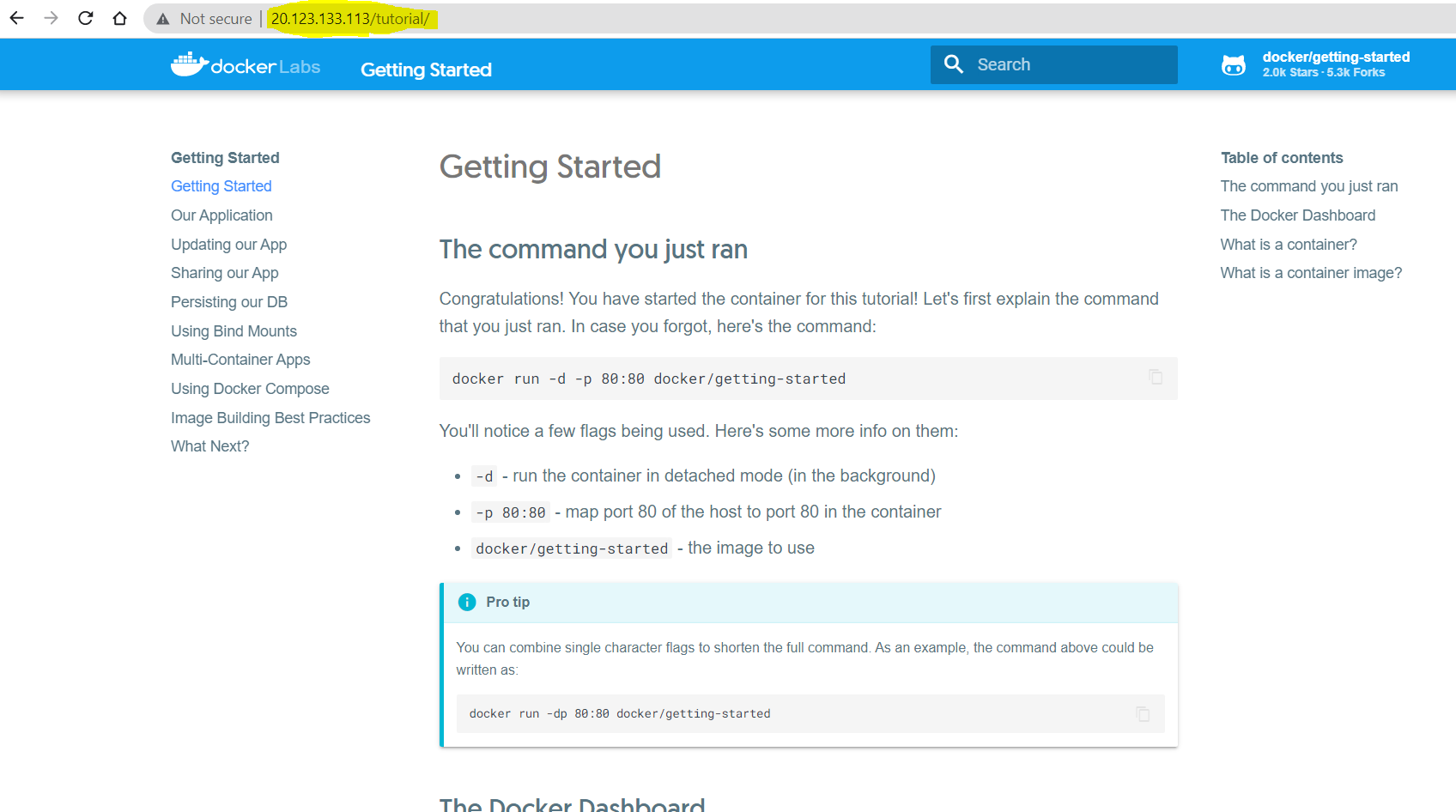
I created new container for this section

docker run –name second-container -d -p 80:80 docker/getting-started

It will create new container with name second-container and expose the port 80 to external through host on 80 and using image name is docker/getting-started.

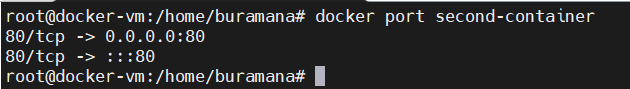


Now I am able to access the application running in container from external network using host ip(where container is running).

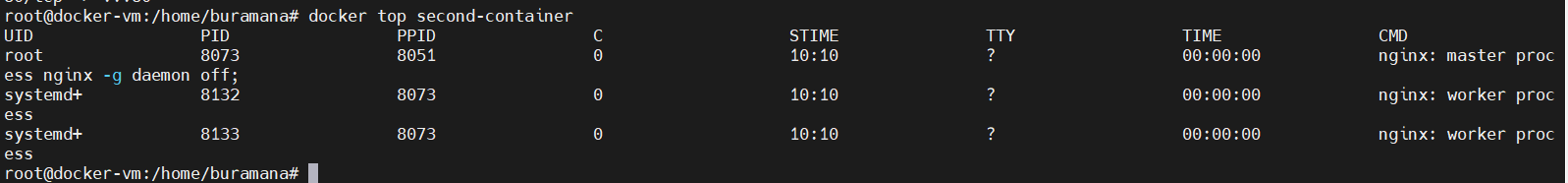


Docker port <container name/id> - to display port number on container and to which port on host itexposed.

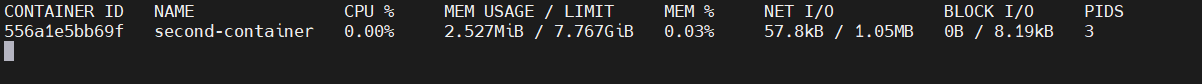
docker port second-container



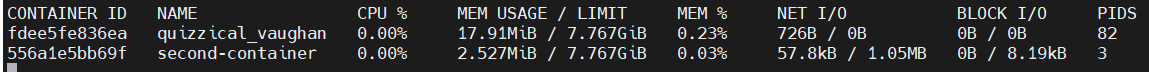
docker top second-container – to display running processes in container



docker stats second-container – to display compute utilisation of resource by particular container

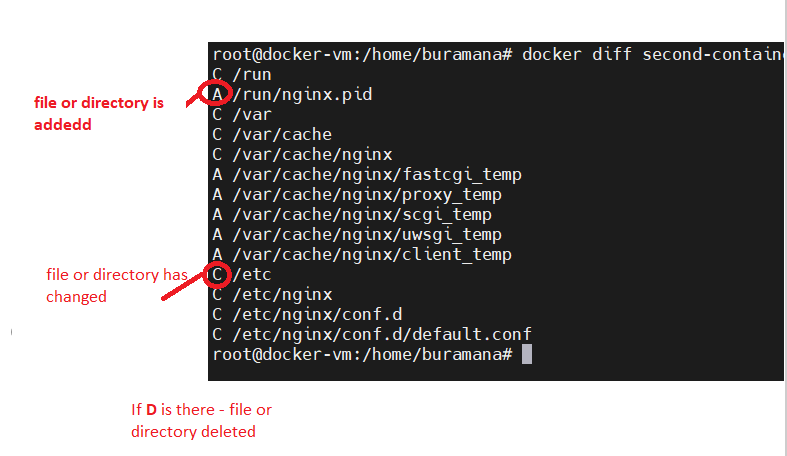


docker stats – to display utilization of resources by each container.



docker diff <container-name/Id> - It will display the changes in container file system.

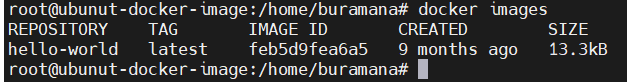
docker diff second-container



# Docker Images

 Docker image is **a file used to execute code in a Docker container**. Docker images act as a set of instructions to build a Docker container, like a template

***docker images -*** To list images in local



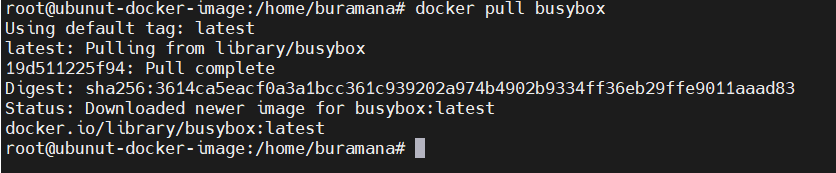
***docker pull <image name>:<tag>*** - to download the image from docker hub public repository.

Note: default tag name will be “latest”

***docker pull nginx*** – pulling nginx image

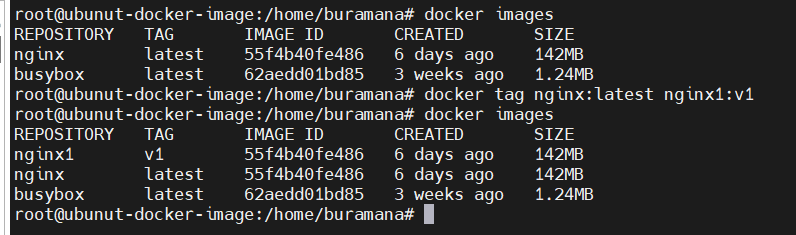


***docker pull busybox*** – pulling busbox image

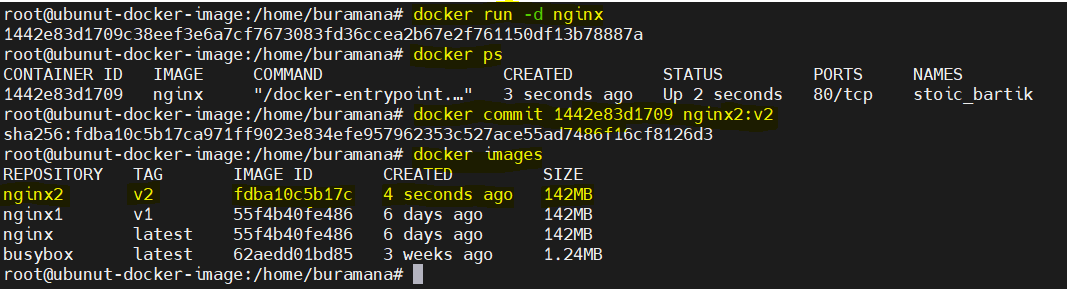


***docker tag <imagename>:<tag> <new-image-name>:<new-tag-name>*** - to retag the existing images

***docker tag nginx:latest nginx1:v1***



***docker commit <container-id> <image-name>:<tagname> -*** Creating image of running container



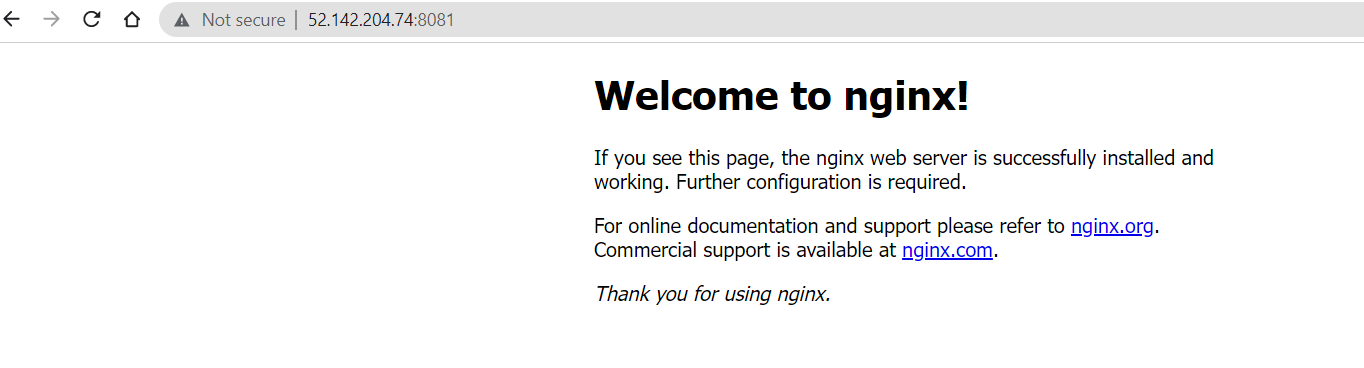
***Creating docker images using Dockerfile***

Dockerfile is simple text file that consists of instructions to create images.

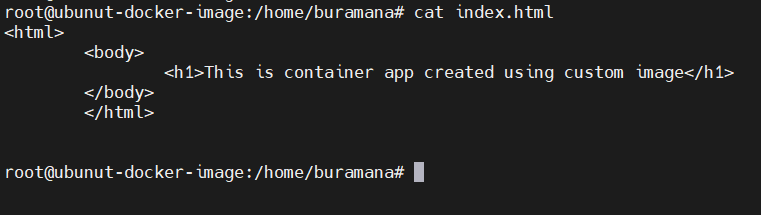
The file name should ***Dockerfile*** without any ***extension***

We will see how to create Dockerfile and creating image from that.

The default web page displayed by nginx container is looks like below.

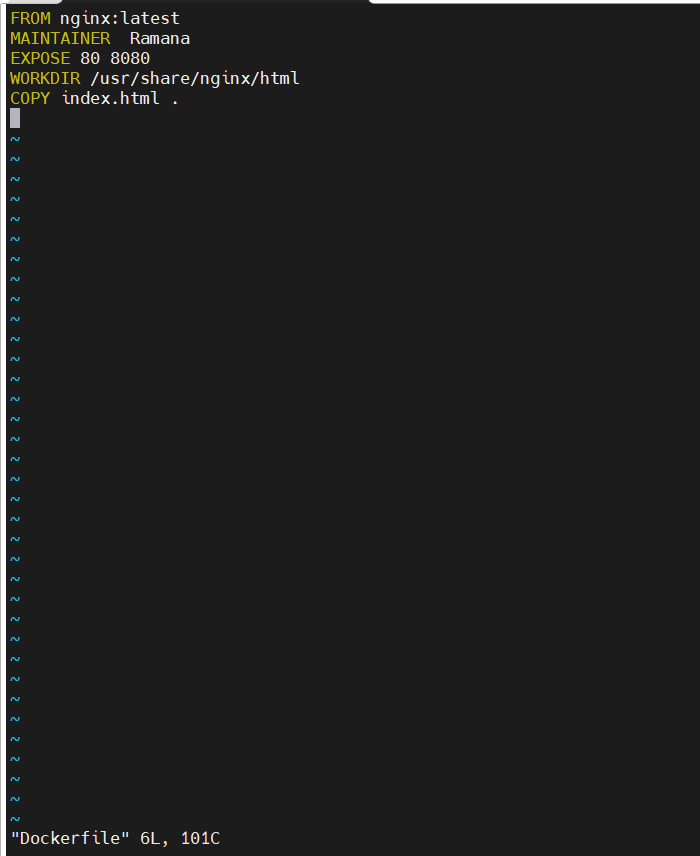


I want to create a new container image that will display the content like below



Dockefile:

Here is the content of dockerfile which I created.



***FROM nginx:latest*** – from instruction to mention the base image to create new image

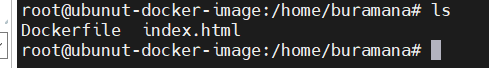
***MAINTAINER Ramana*** – Maintainer instruction is to give the name who creating this dockerfile

***EXPOSE 80 8080*** – Expose instruction to allow the container ports to external network to access the container application

***WORKDIR /usr/share/nginx/html*** – Workdir instruction make the given path as current directory

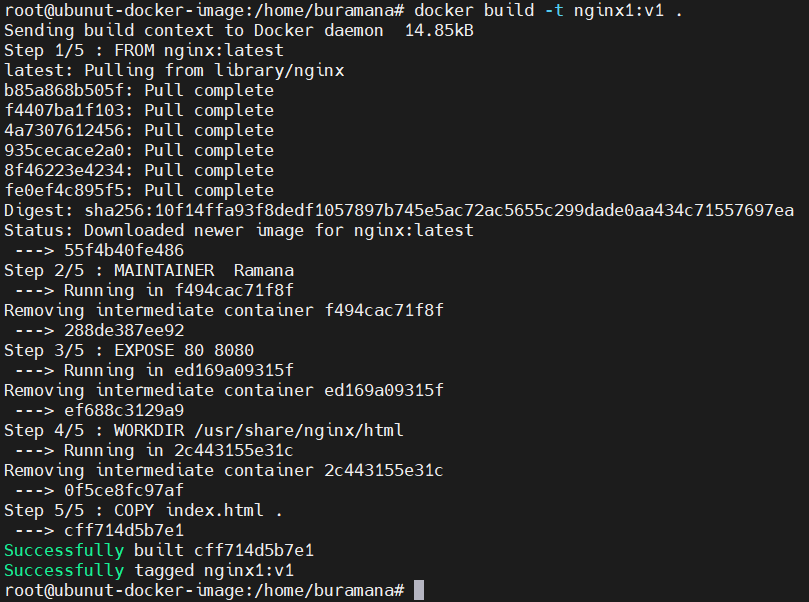
***COPY index.html . –*** Copy instruction to copy the files to docker image

Now I have two files

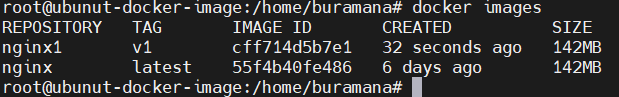


now to build the new image using Dockerfile

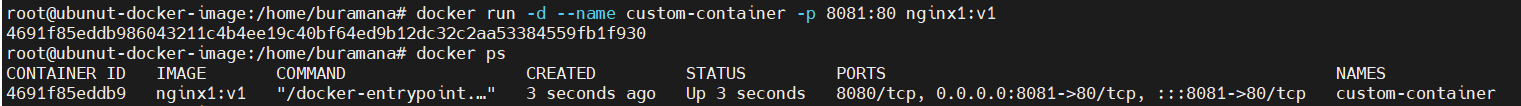
***docker build -t <imagename>:<tag-name> .(dot)***



Here is the new image just now created



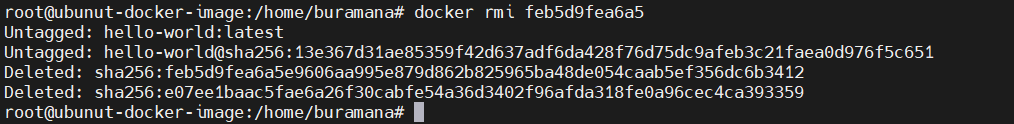
Now I created the container using the newly created image



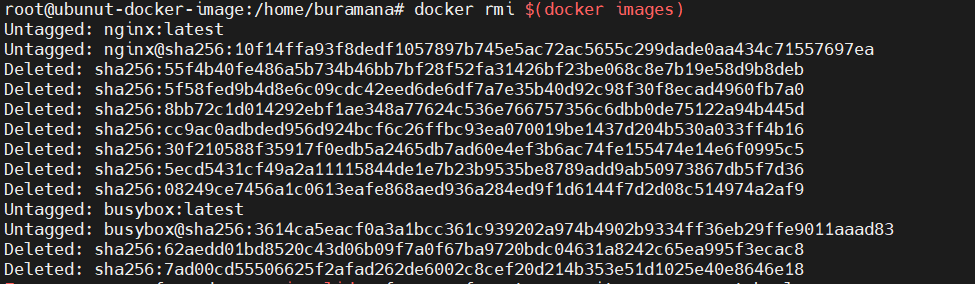
Now try to access the app with host ipaddress



***docker rmi <image name or ID > -*** to remove particular image.it deleted the hello-world image

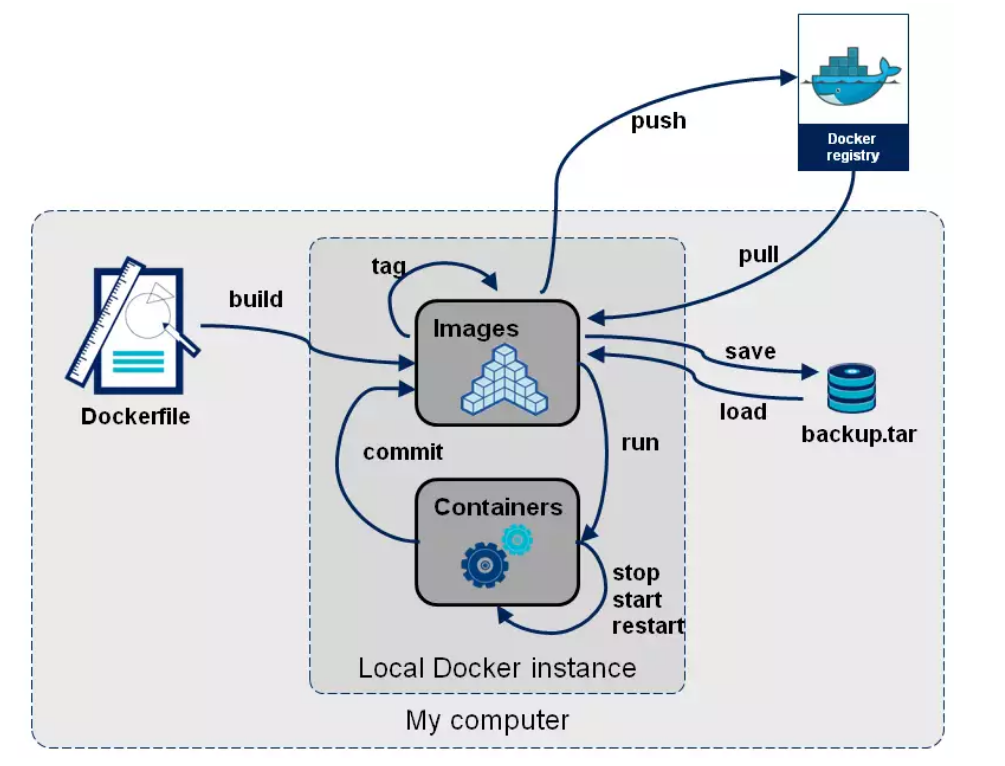


***docker rmi $(docker images) –*** to remove all images in local. It deleted nginx and busybox images



# Docker Registry

The Registry is a stateless, highly scalable server-side application that stores and lets you distribute Docker images



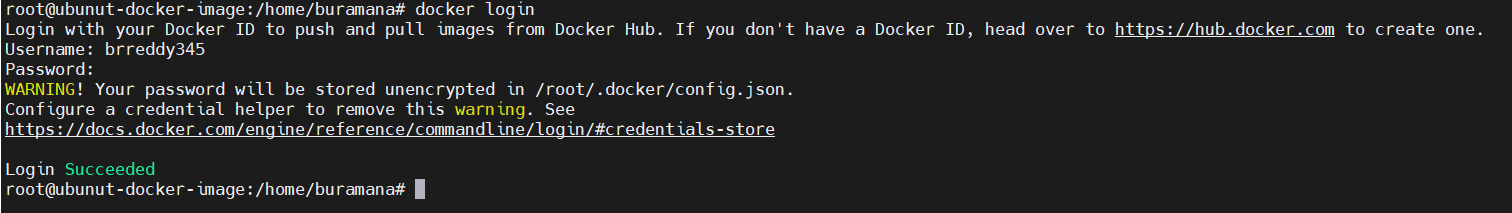
We will interact with registry while pull and push the images form docker client.

To create private registry in docker hub [docker hub signUp](https://hub.docker.com/).

By default when docker installed docker client connect to public registry, from public registry we only pull the images. <http://docker.io/library> is the default registry.

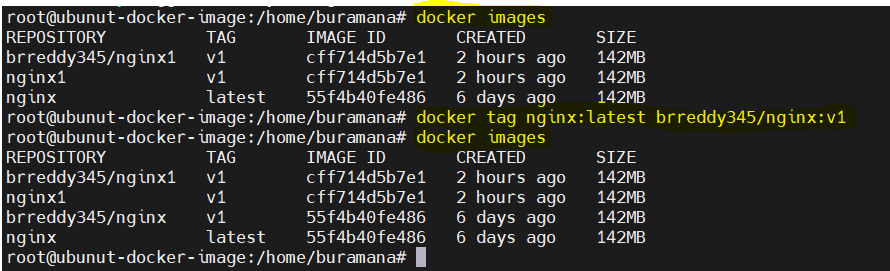
After signed up to docker hub, need to login from cli.

***docker login*** and hit enter, it will ask for ***username*** and ***password***



To push existing image to private registry we need to re-tag the image as below

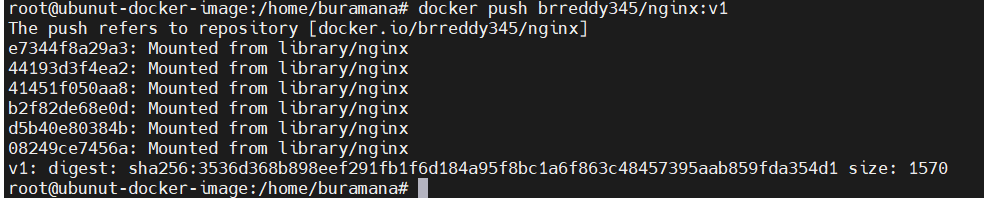
***docker tag <old-image-name>:<tag-name> <registry-name>/<new-image-name>:<tag-name>***



To push image to registry

***docker push <registry-name/<image-name>:<tag-name>***

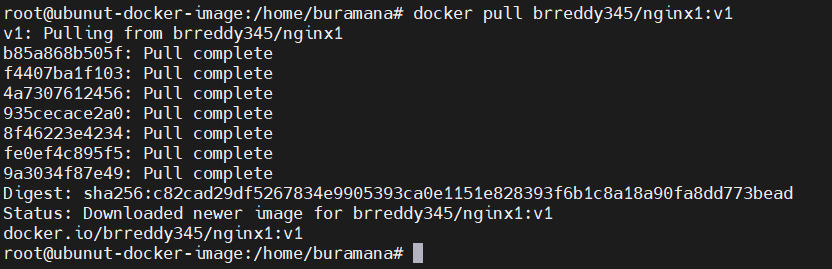
***docker push brreddy345/nginx:v1***



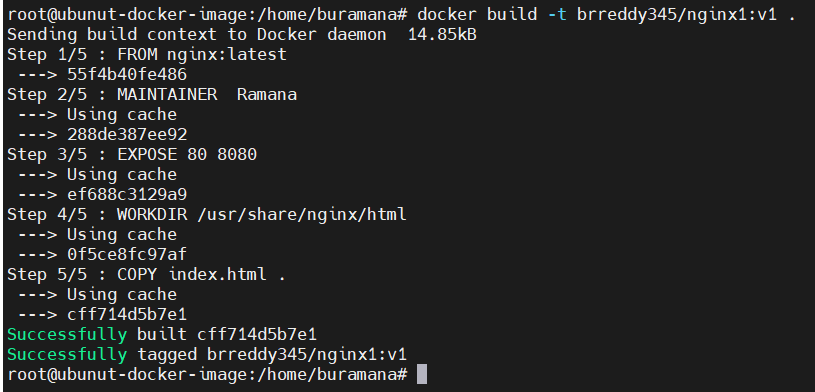
To pull the image from remote registry

***docker pull <registry-name/<image-name>:<tag-name>***

***docker pull brreddy345/nginx:v1***



Create new build image using Dockerfile as below.



# Docker Compose

# Docker Engine, Storage and Networking

# Container Orchestration