1. **Aim: Building APT.NET Core MVC Application.**

1)Install .Net Core Sdk (Link: <https://dotnet.microsoft.com/learn/dotnet/hello-world-tutorial/install>)

2)create folder MyMVC folder in C: drive or any other drive

3)open command prompt and perform following operations

Command: to create mvc project

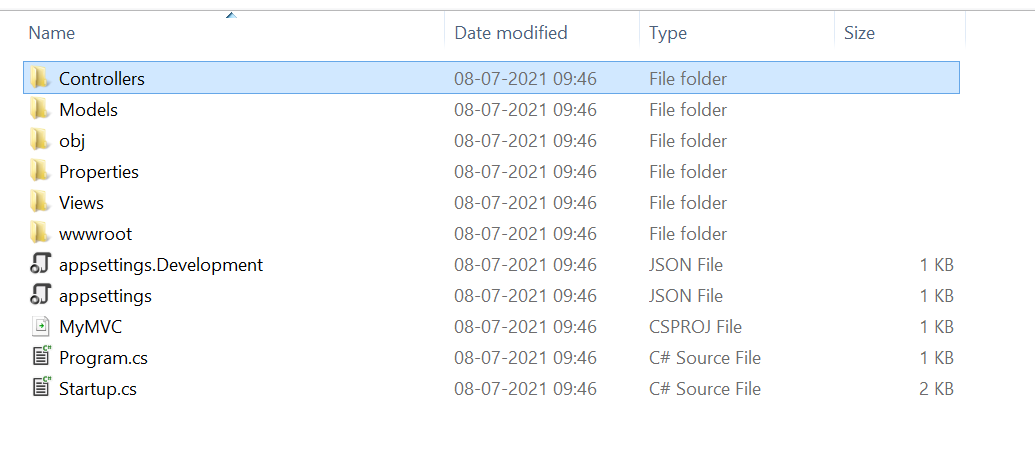
dotnet new mvc --auth none

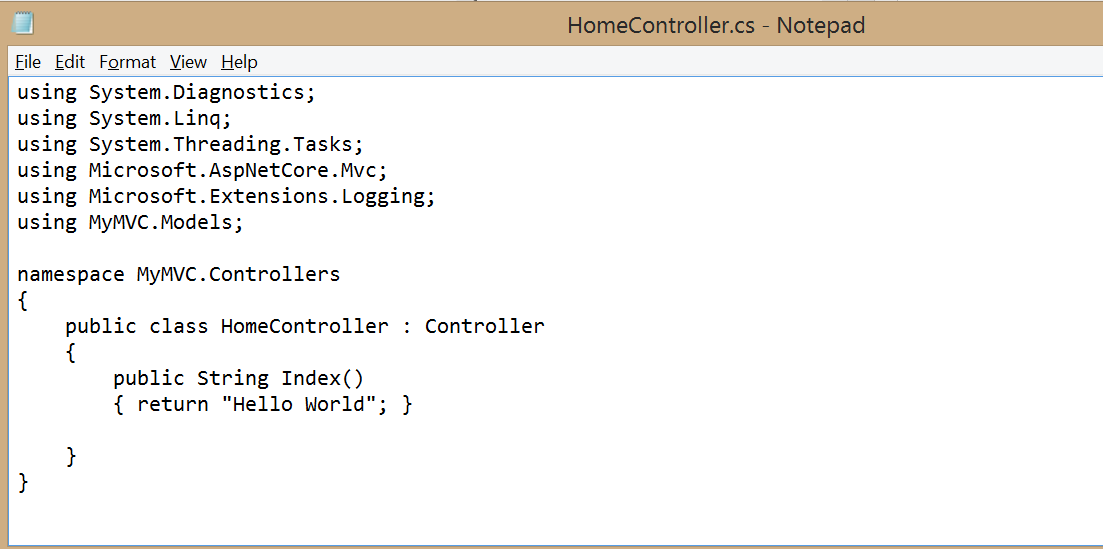
output:

Graphical user interface, application, Word

Description automatically generated

4) Go to controllers folder and modify HomeController.cs file to match following:

****

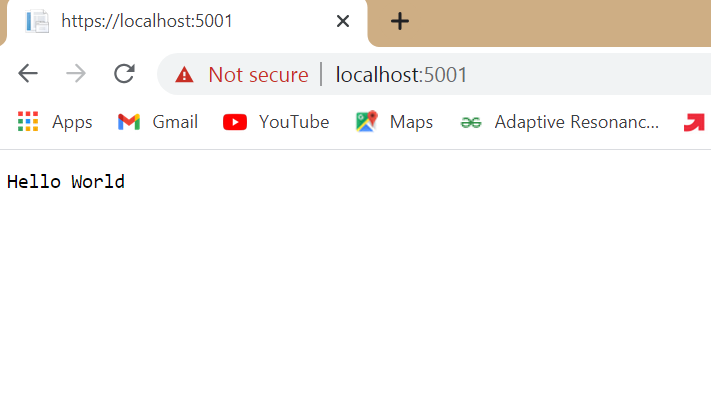
****

Run the Project

Text

Description automatically generated

Now open browser and and type URL: localhost:5000

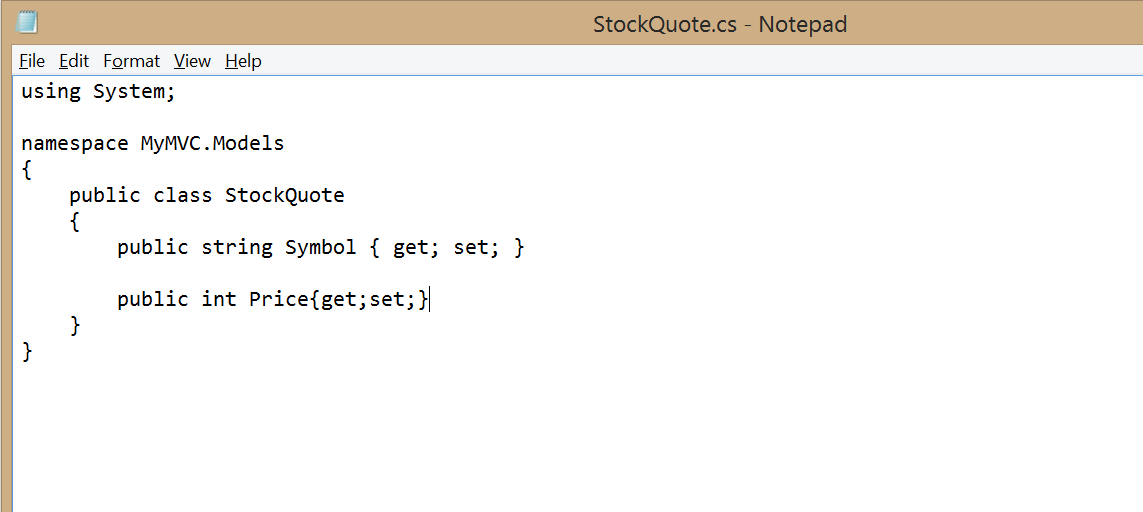


Now go back to command prompt and stop running project using CTRL+C

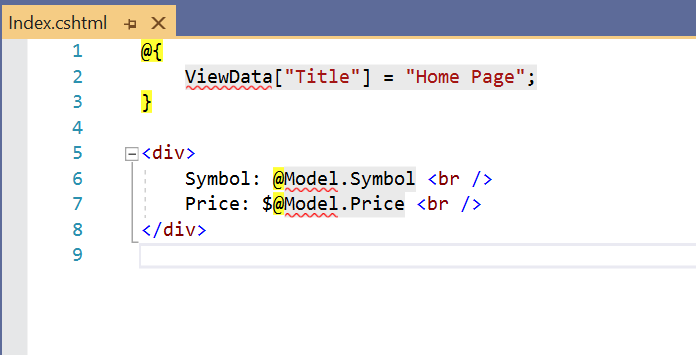
Text

Description automatically generated

Go to models folder and add new file StockQuote.cs to it with following content



Now Add View to folder then home folder in it and modify index.cshtml file to match following



Now modify HomeController.cs file to match following:

Graphical user interface, text, application

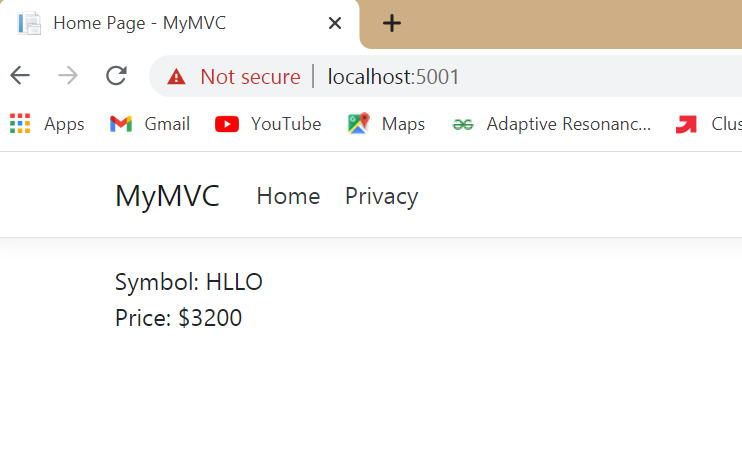
Description automatically generated

Now run the project using

Text

Description automatically generated

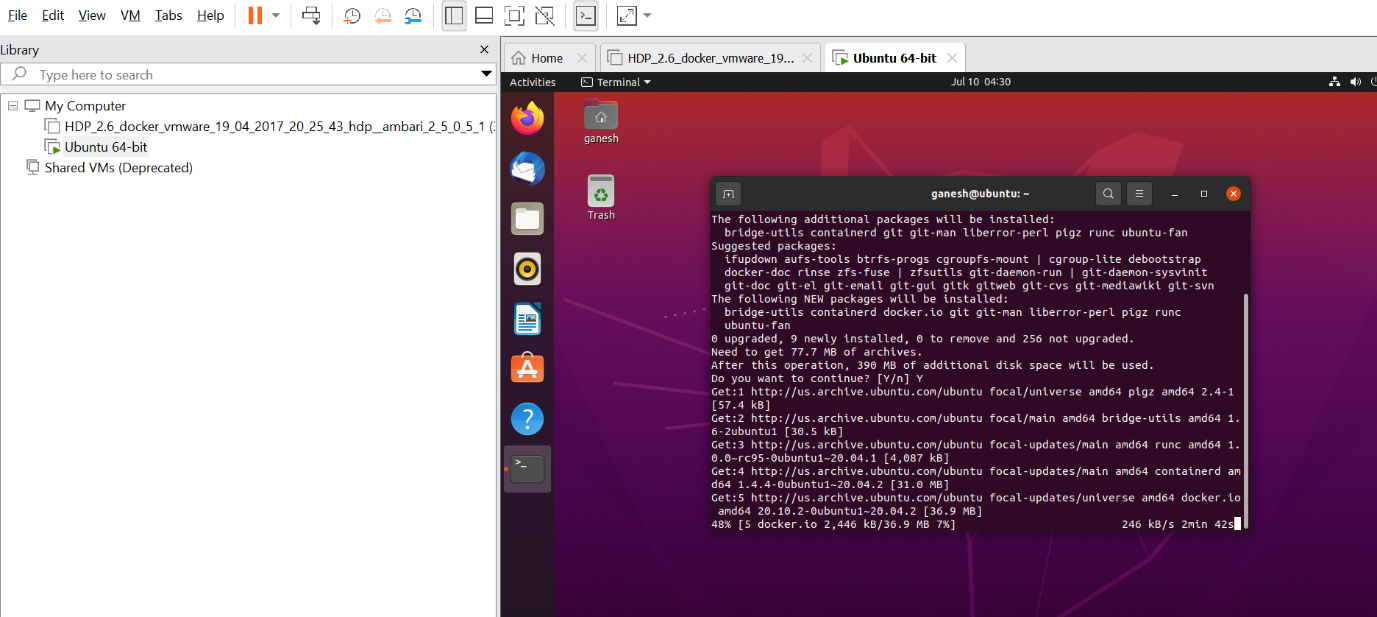
Now go back to browser and refresh to get modified view response



1. **Aim: Working with Docker, Docker Commands, Docker Images and Containers**

**After install ubuntu in vmware. Install docker**

**Command: sudo apt-get install docker.io**

****

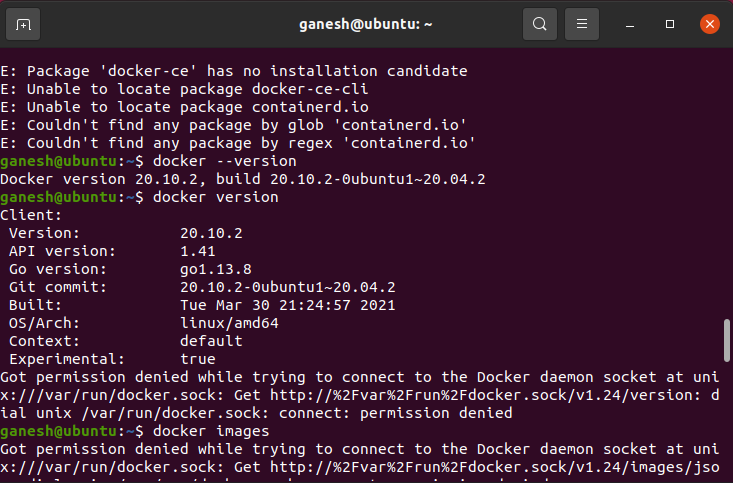
Install using the repository

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.

Docker Commands:

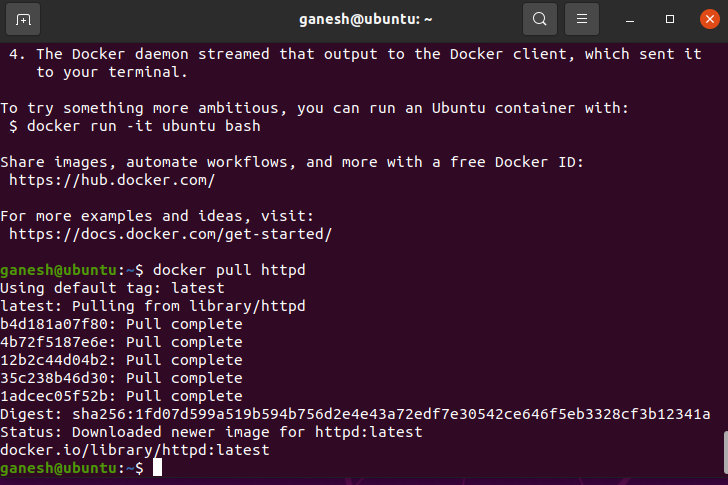
Docker –version

Docker version

****

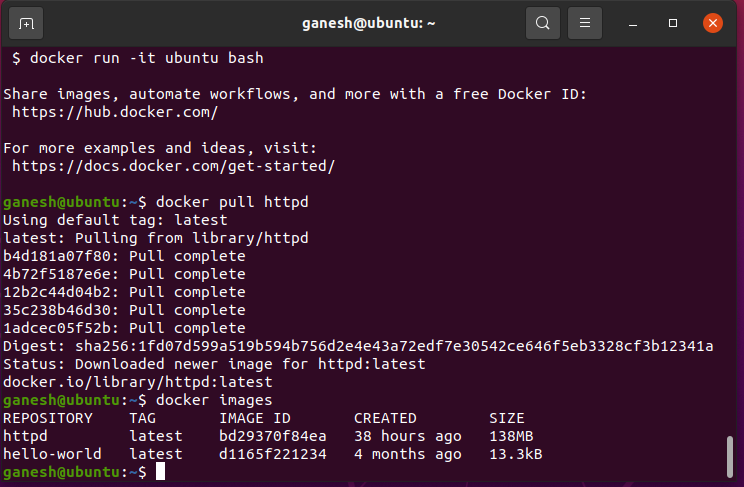
**Docker pull httpd**

Pull an image or a repository from a registry



**Docker images**

It lists all the images



**#nano Dockerfile**

**FROM busybox**

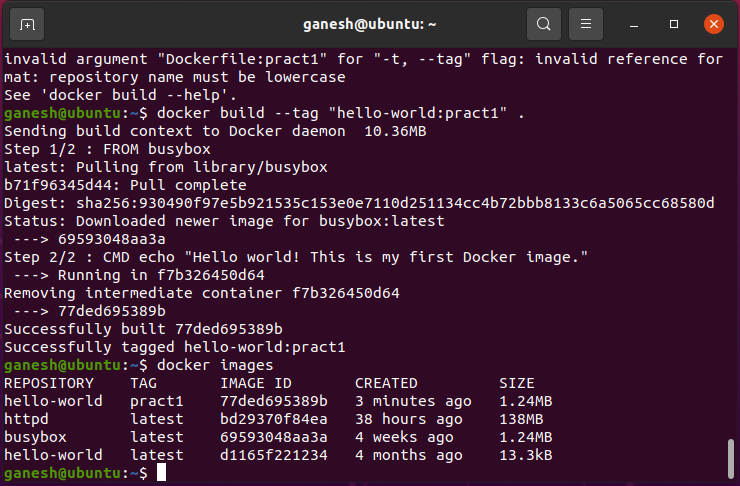
**CMD echo "Hello world! This is my first Docker image."**

**//above two line we have to add into dockerfile**

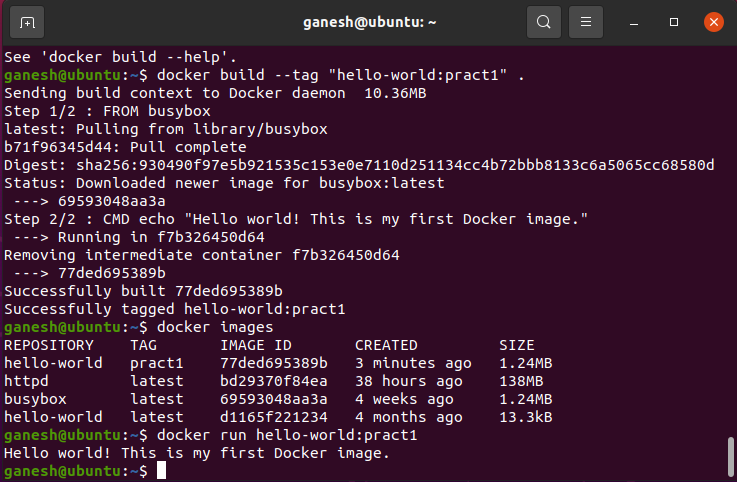
**to save press ctrl+o(to write) then enter then ctrl+x (to exit)**

**docker build --tag "hello-world:pract1" .**

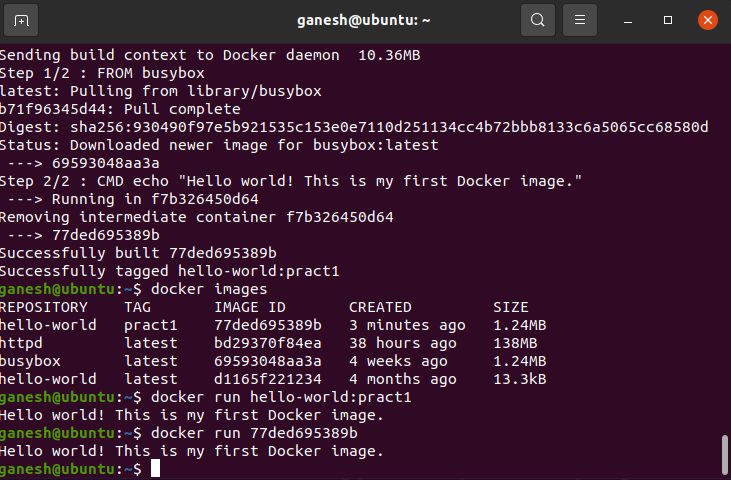
**docker images**

****

**docker run hello-world:pract1**

****

**docker run 77ded695389b**

****

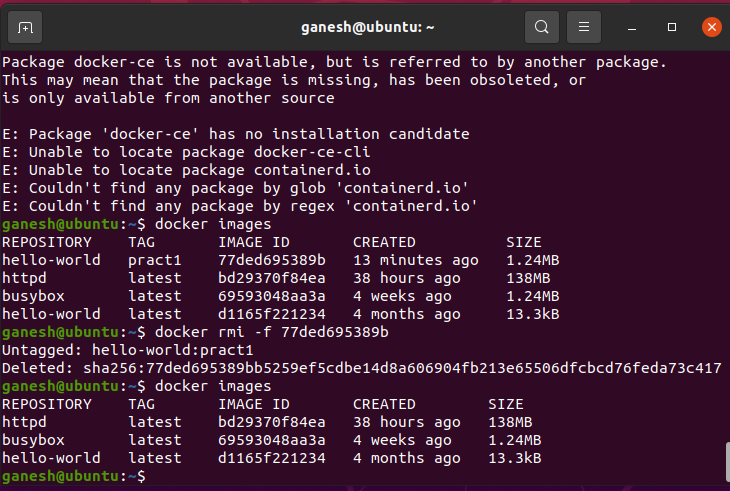
**Docker rmi**

**Remove one or more images**

**docker rmi -f images-id**

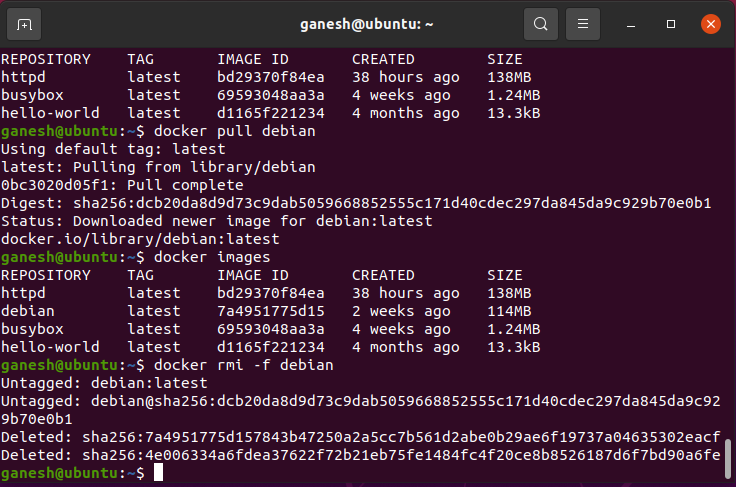
**docker rmi -f 77ded695389b**

**After running docker images we can see that 77ded695389b is deleted.**

****

**docker rmi -f Respository-name**

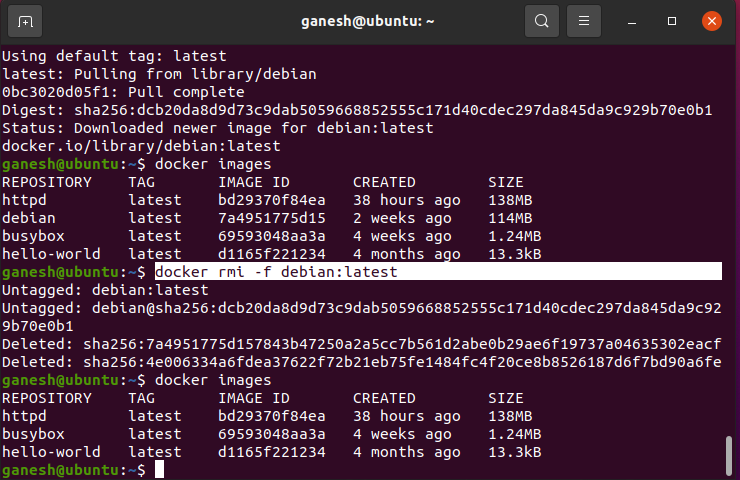
**docker rmi -f Debian**



**docker rmi -f Respository-name:tag**

**docker rmi -f debian:latest**

**After this debain image will be deleted**

****

1. **Aim: Installing software packages on Docker, Working with Docker Volumes and Networks.**

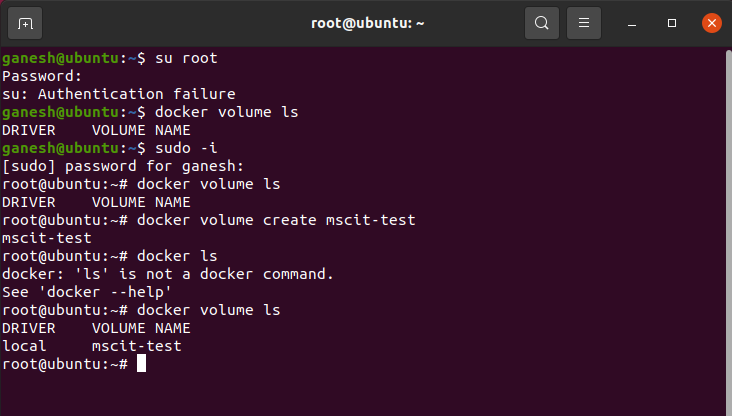
Volumes are the preferred mechanism for persisting data generated by and used by Docker containers. While [bind mounts](https://docs.docker.com/storage/bind-mounts/) are dependent on the directory structure and OS of the host machine, volumes are completely managed by Docker.

List volumes created

Command: docker volume ls

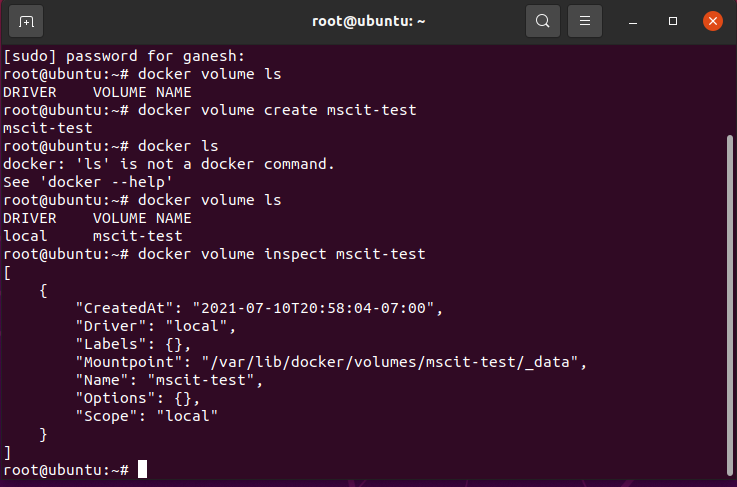
To create volume.

Command: docker volume create mscit-test



Return low-level information on Docker objects

Command: docker volume inspect mscit-test

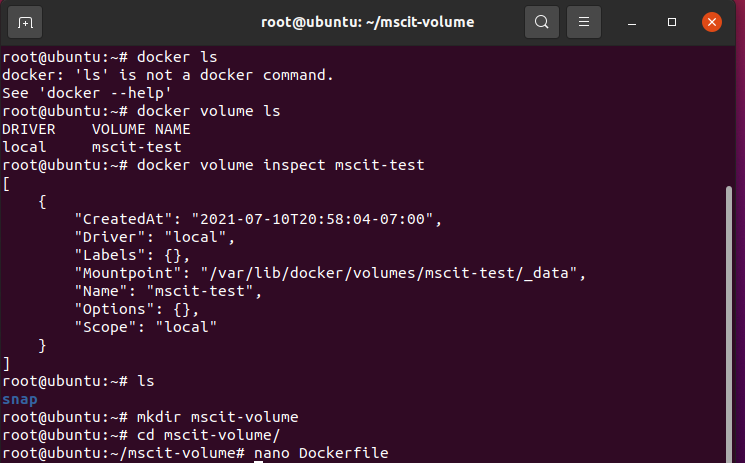


Create a directory

mkdir mscit-volume

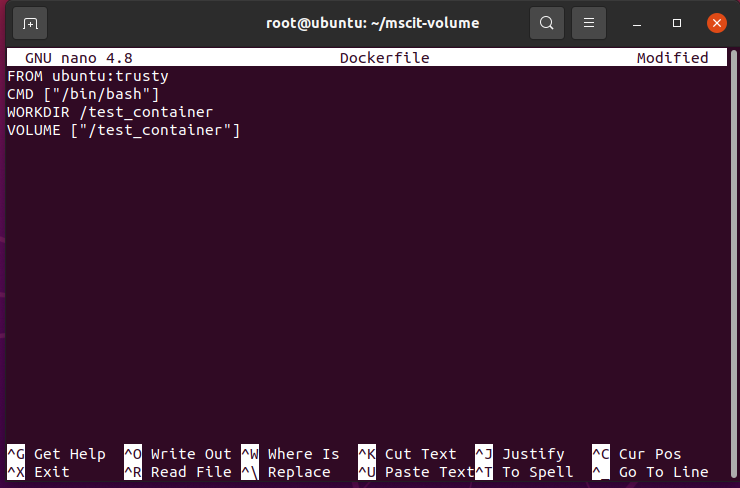
Now, change directory to mscit-volume

cd mscit-volume/



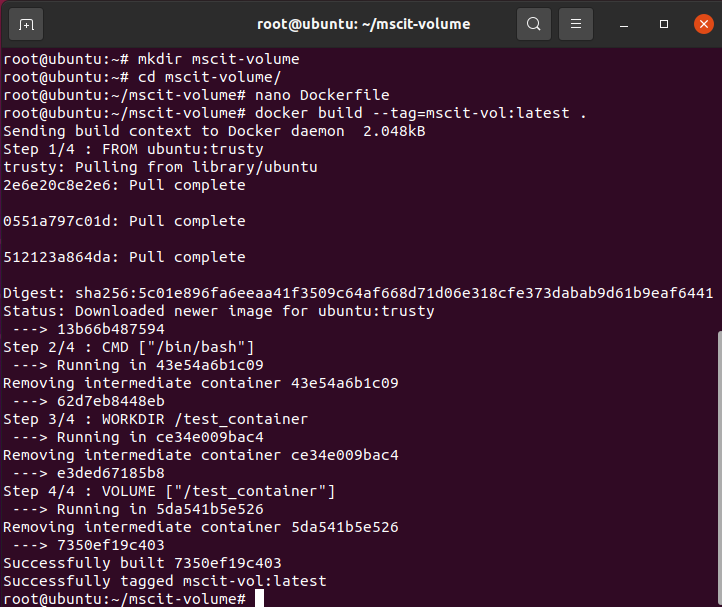
Create a file

Nano Dockerfile



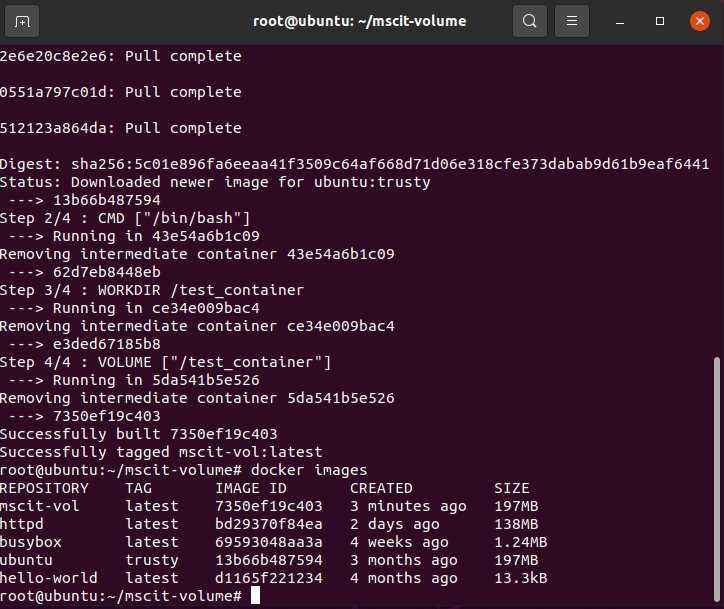
To create an image file

docker build --tag=mscit-vol:latest .



Check the image create

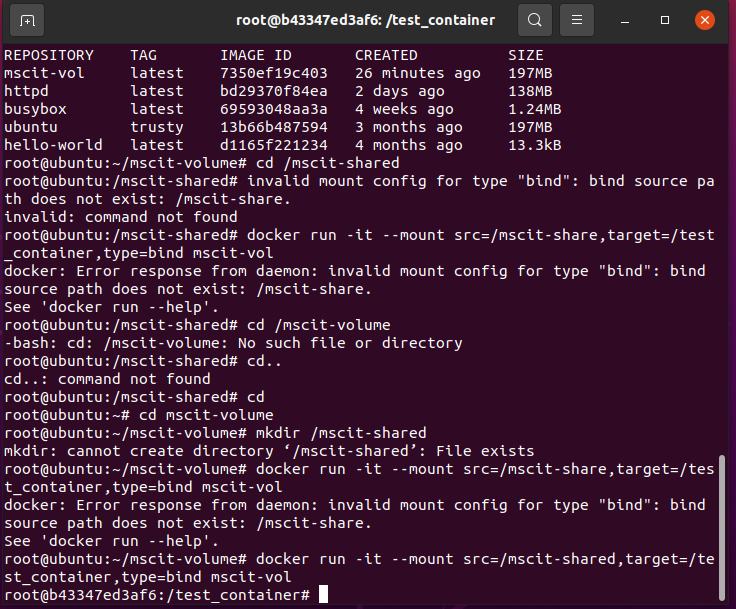
Command: docker images



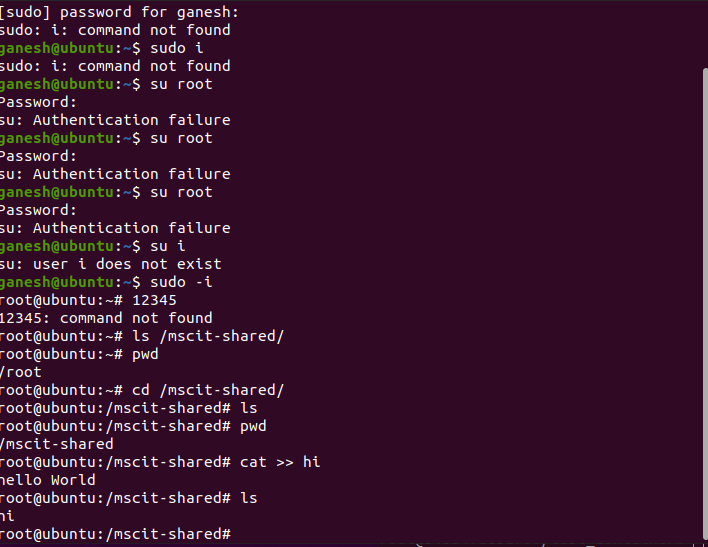
Mounting the container

docker run -it --mount src=/mscit-shared,target=/test\_container,type=bind mscit-vol

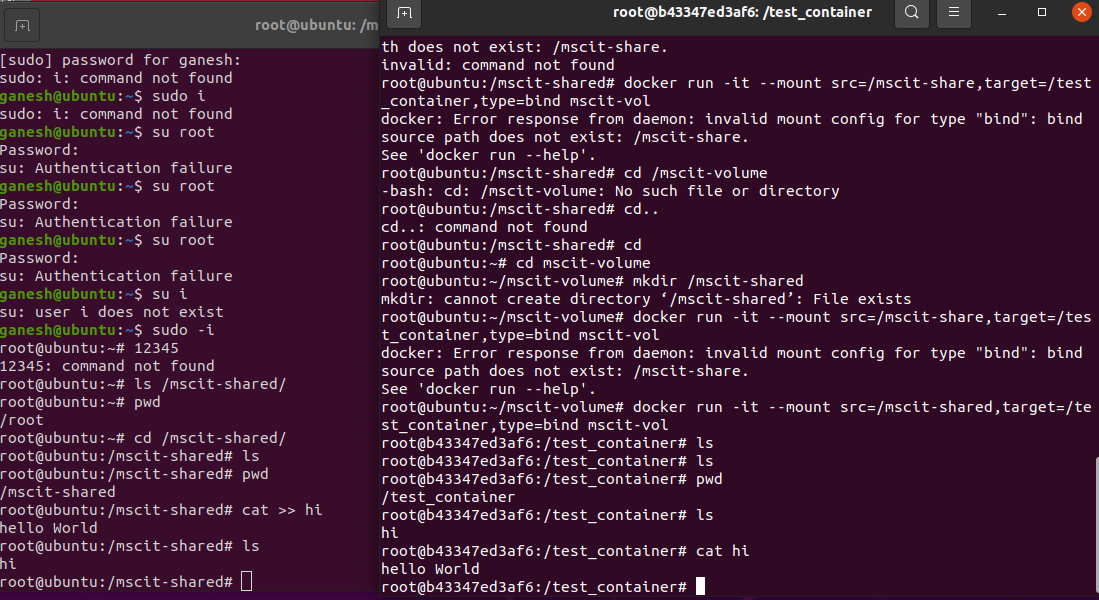
It will change in root and show test\_Container

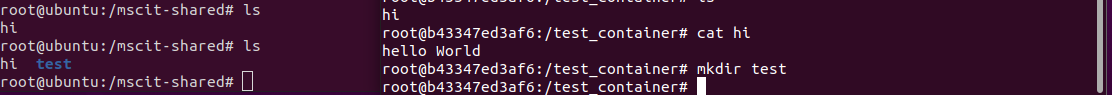


Now open other terminal and get into mscit-shared directory and create a file called hi



Now check the file created in root is listed in test\_Container and vice-versa.





We can see that file location are mapped.

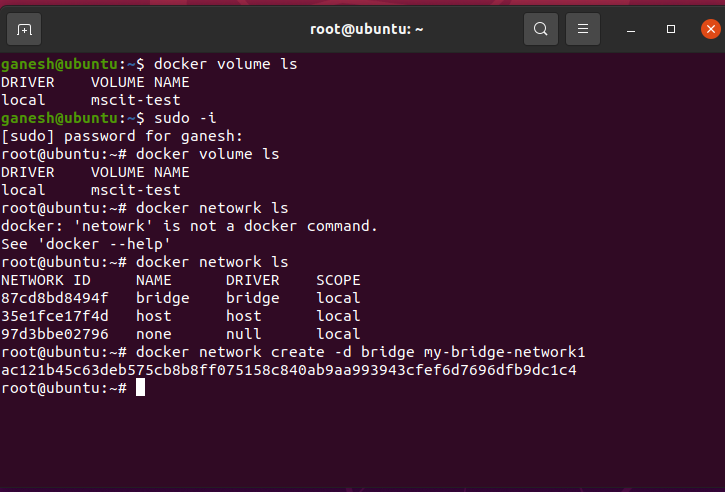
When below command is executed, it will delete the volume.

docker volume rm mscit-test

**Network:**

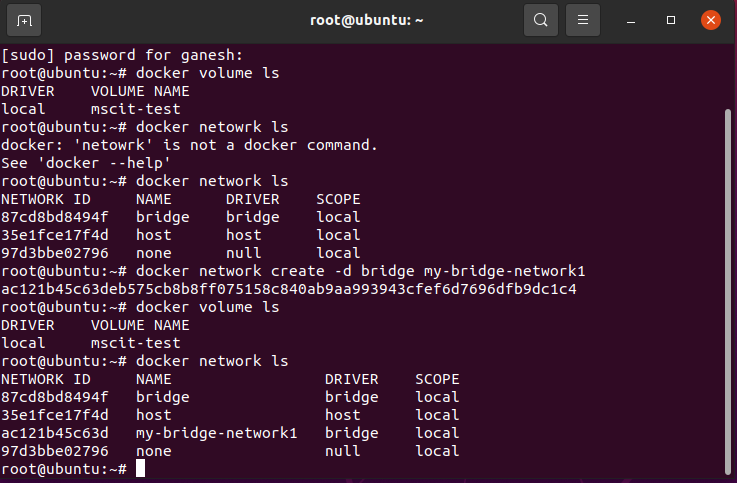
Create network with following command

docker network create -d bridge my-bridge-network1



Check network is created with below command

Command: docker network ls



We can inspect the created network with below command

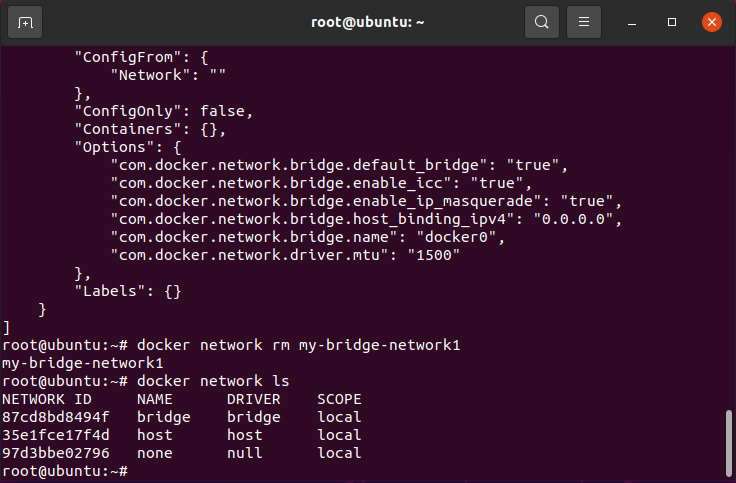
docker network inspect bridge (network name)



Now, lets remove the create network using below command.

docker network rm network-name

With docker network ls we can see the my-bridge-network1 is delected.



With below command we can delete unused networks

docker network prune

