**Part - I**

1. Demonstrate to create a container and run the official (Docker Hub) images using Docker.

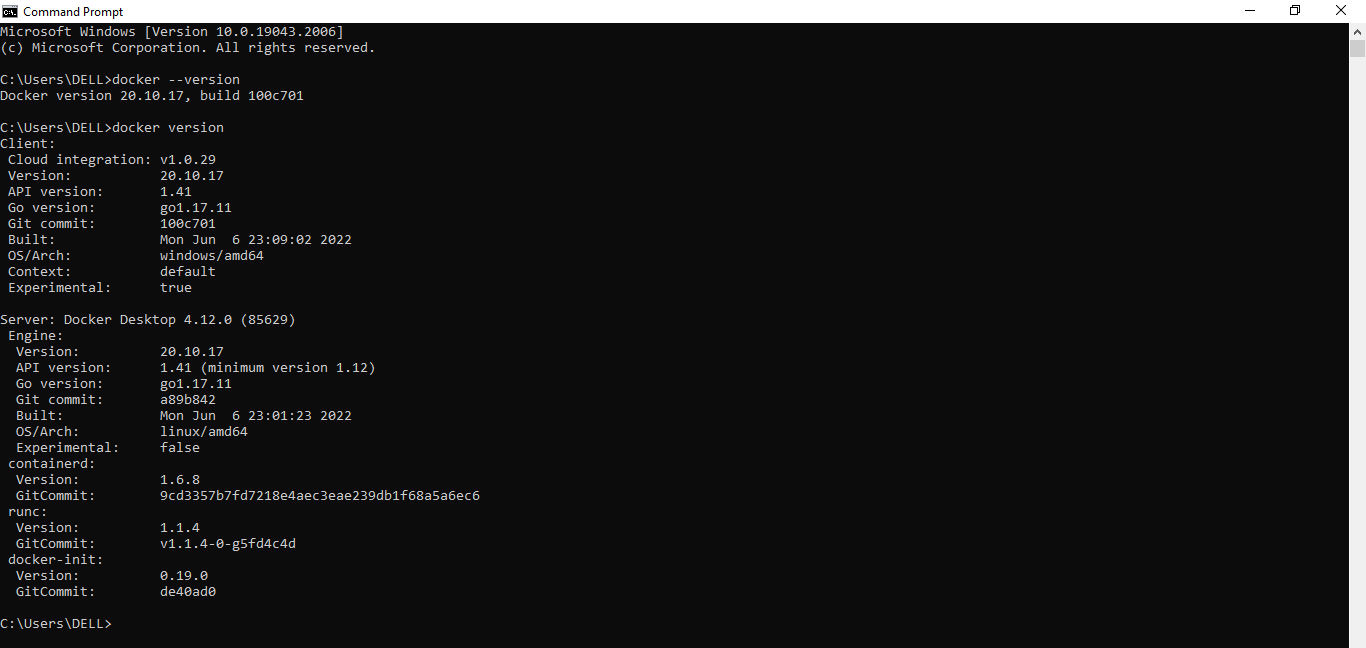
For example, images like hello-world, busybox. Include screenshots of  
pull command to pull the image

List the newly created Image

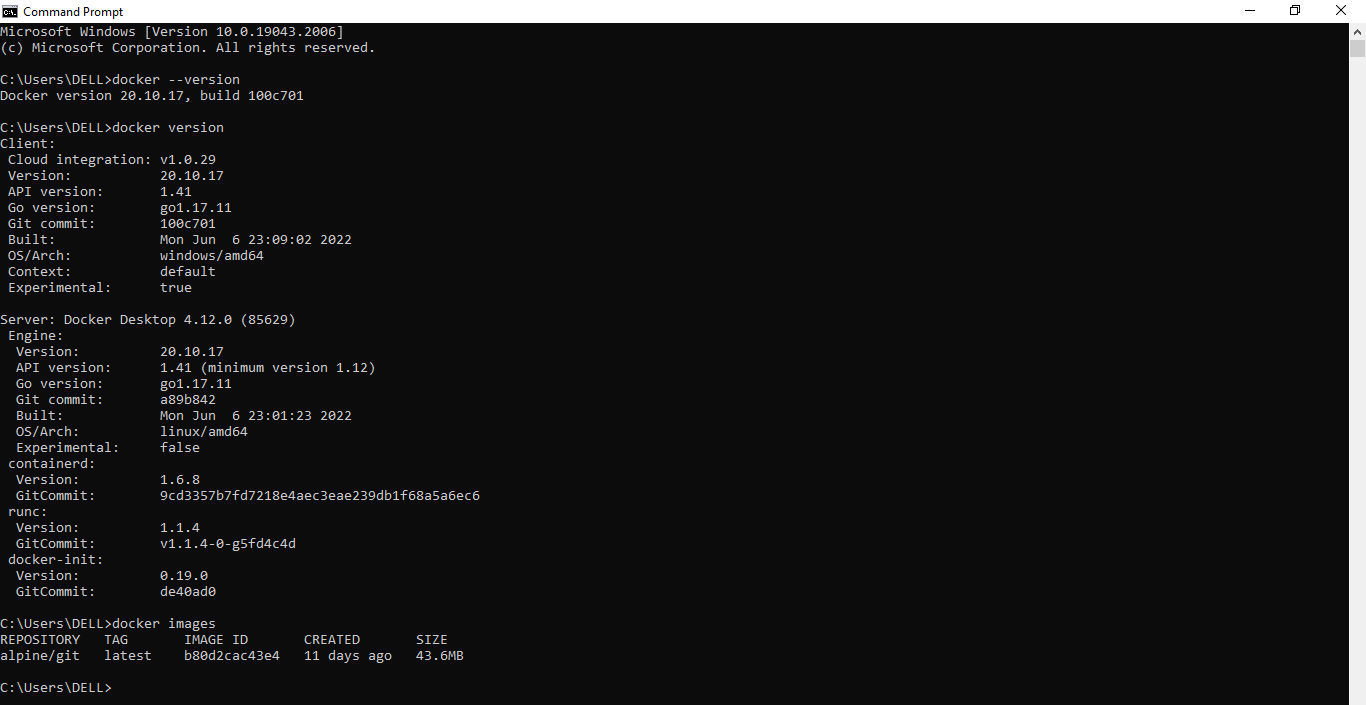
Run the image Locally.

**Output :**

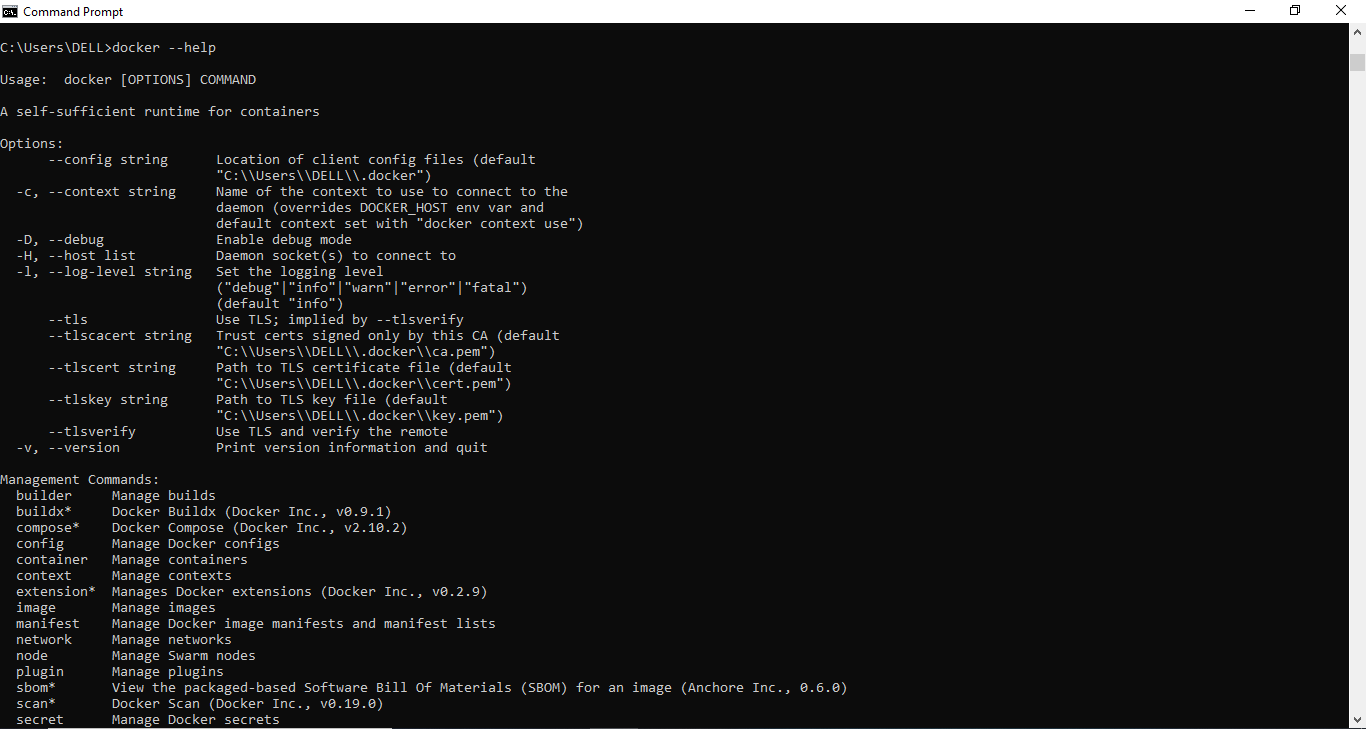
Checking the version of docker –version, docker version to check whether client is connected to the server side or not.

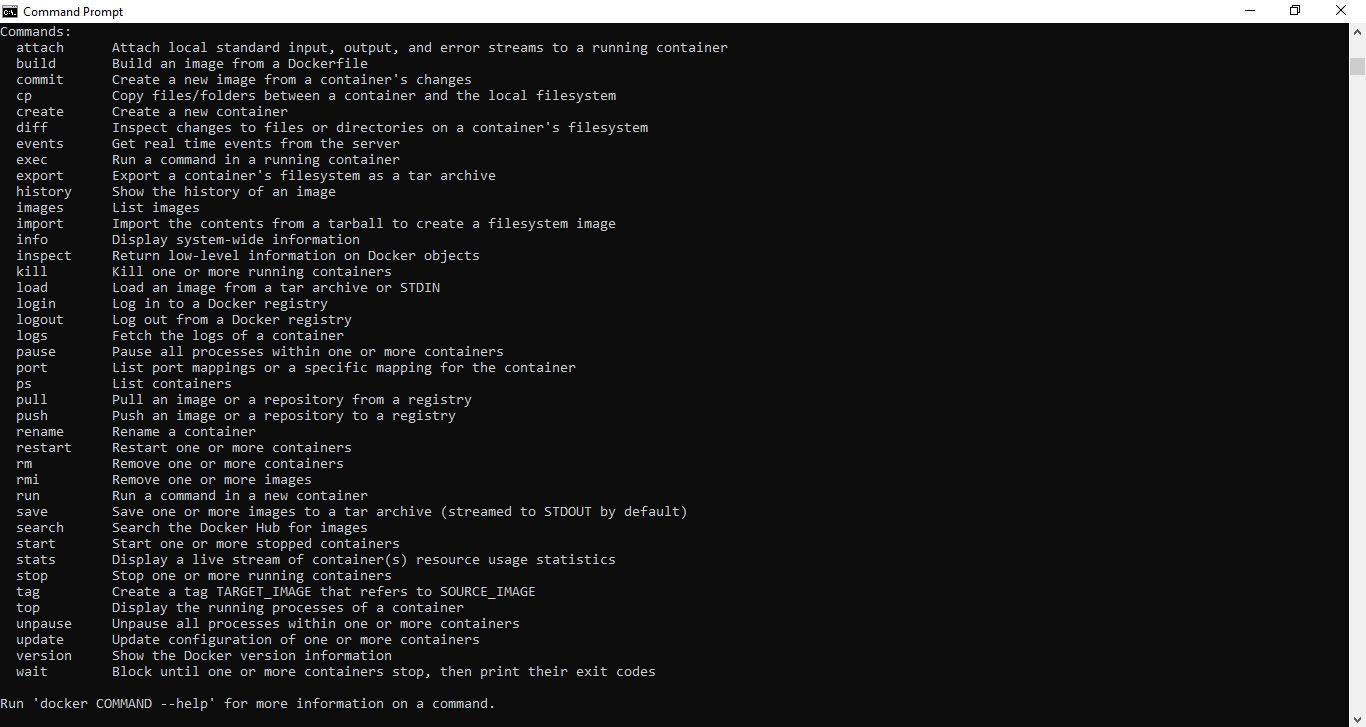
****

Checking docker images.

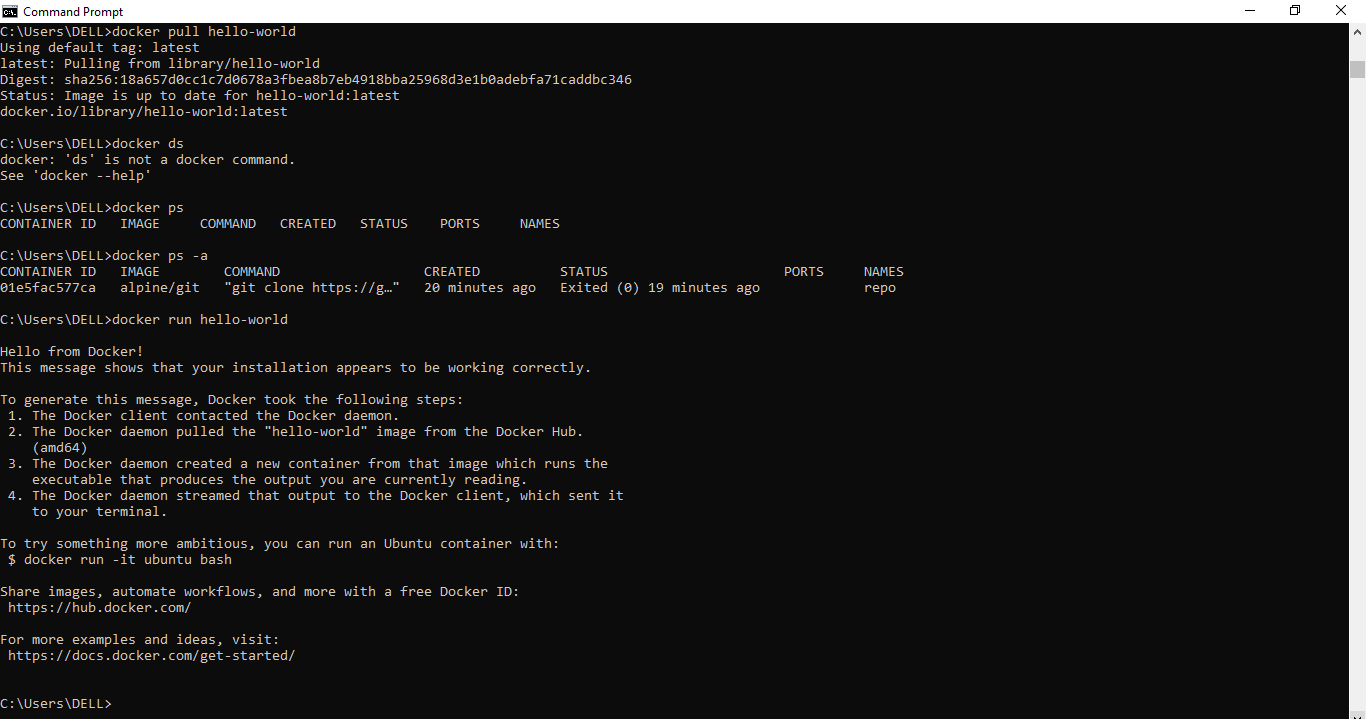
****

Here to check different commands in the docker by entering the docker –help.

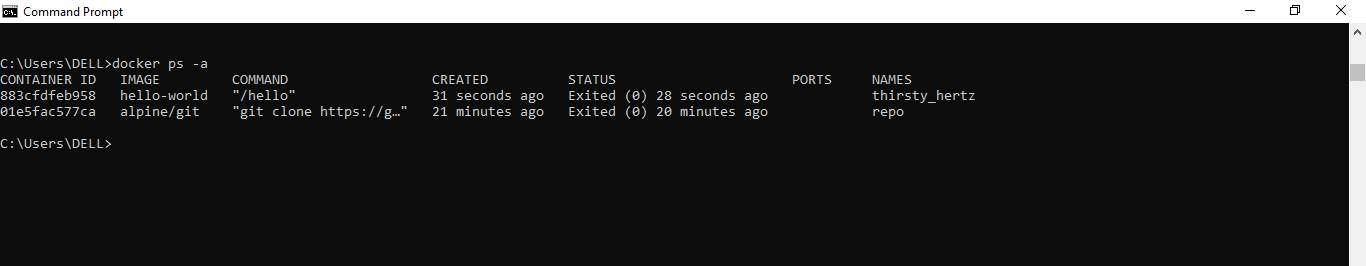
****

****

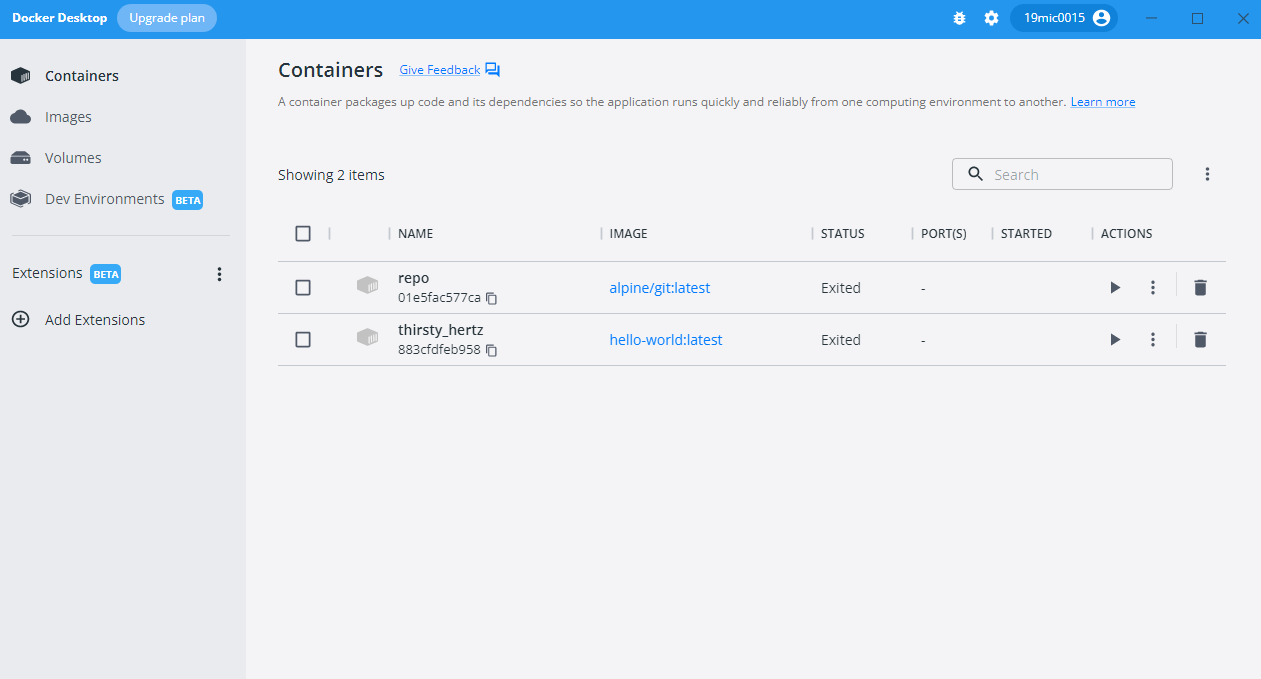
Here we have to pull the hello-world by entering the docker pull hello-world. And to check this whether it is pulled by entering the docker ps –a.

****

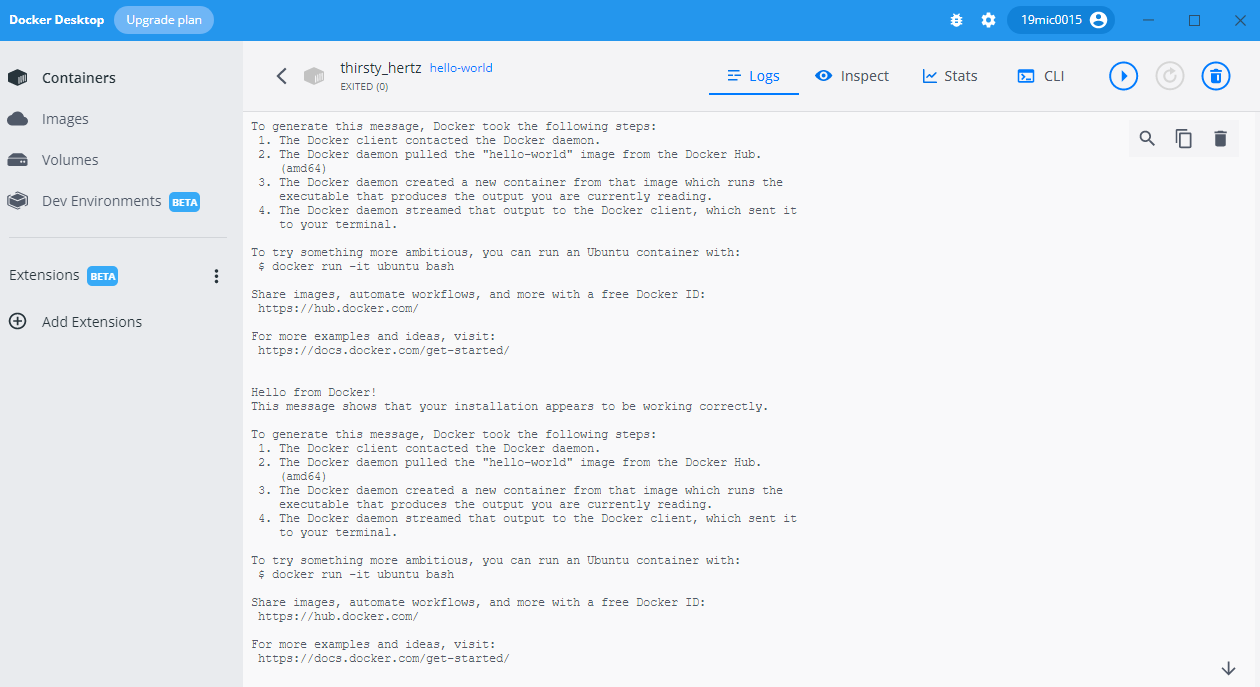
Then type as docker run hello-world and to check whether the dependency are available in the container or not by using cmd as docker ps -a

****

Here after pulling the resources the output of the container .

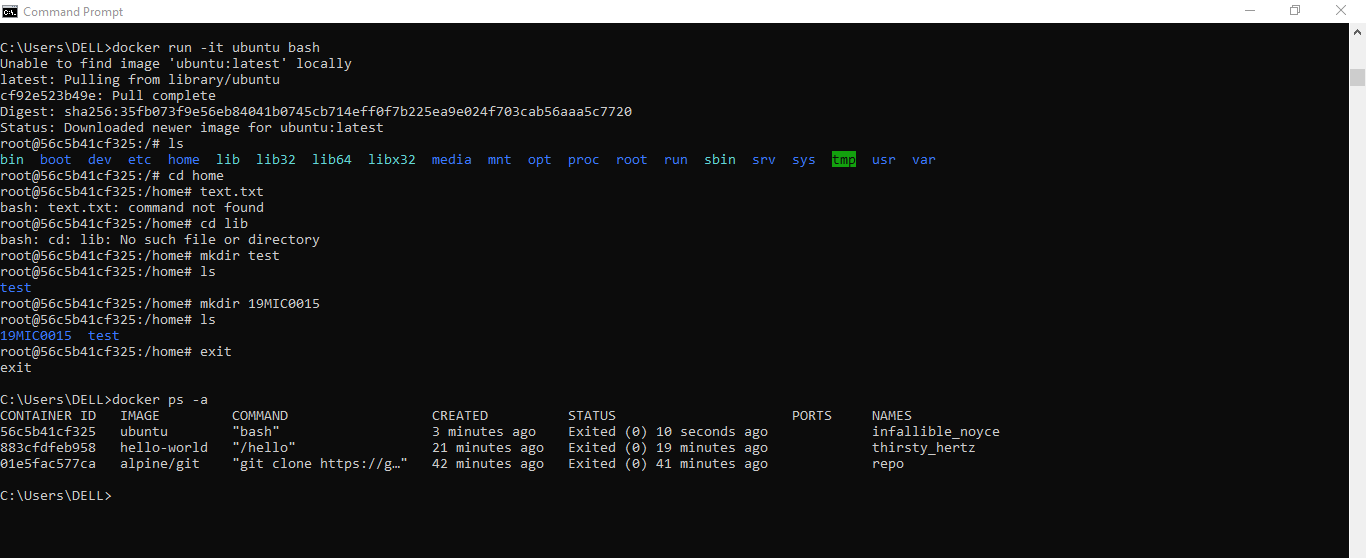
****

Here we have the output of the hello-world image in the docker hub.

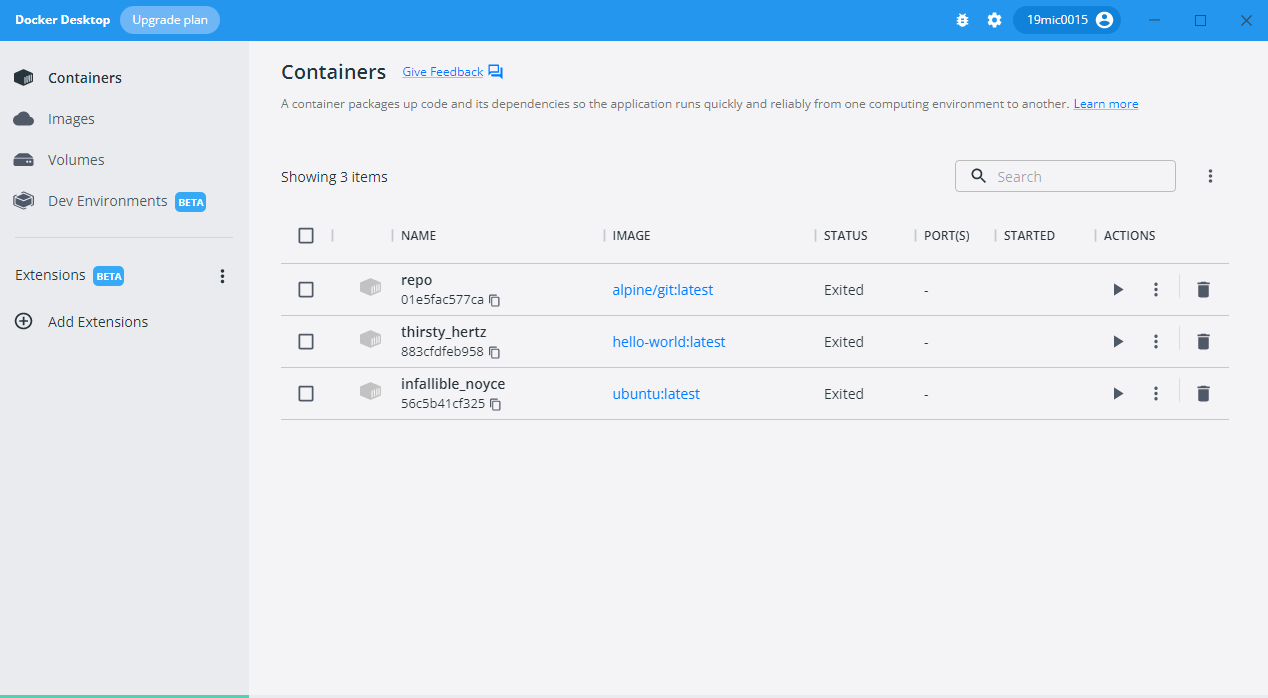
****

Here now we need to pull the Ubuntu resource for displaying the Ubuntu image in the docker hub.

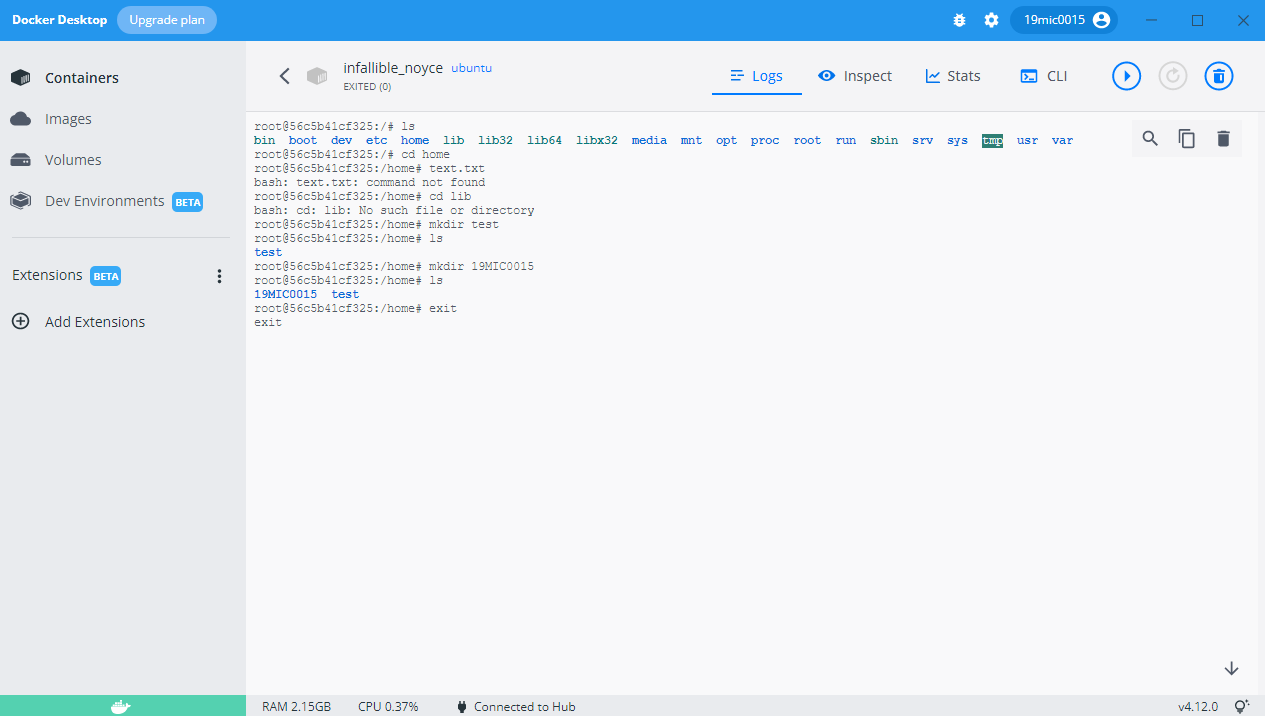
Type as docker run –it Ubuntu bash, for running the Ubuntu image in the docker hub. Then we have Ubuntu shell in that type as ls and by using mkdir create some folder on the Ubuntu shell and check whether is working or not.

****

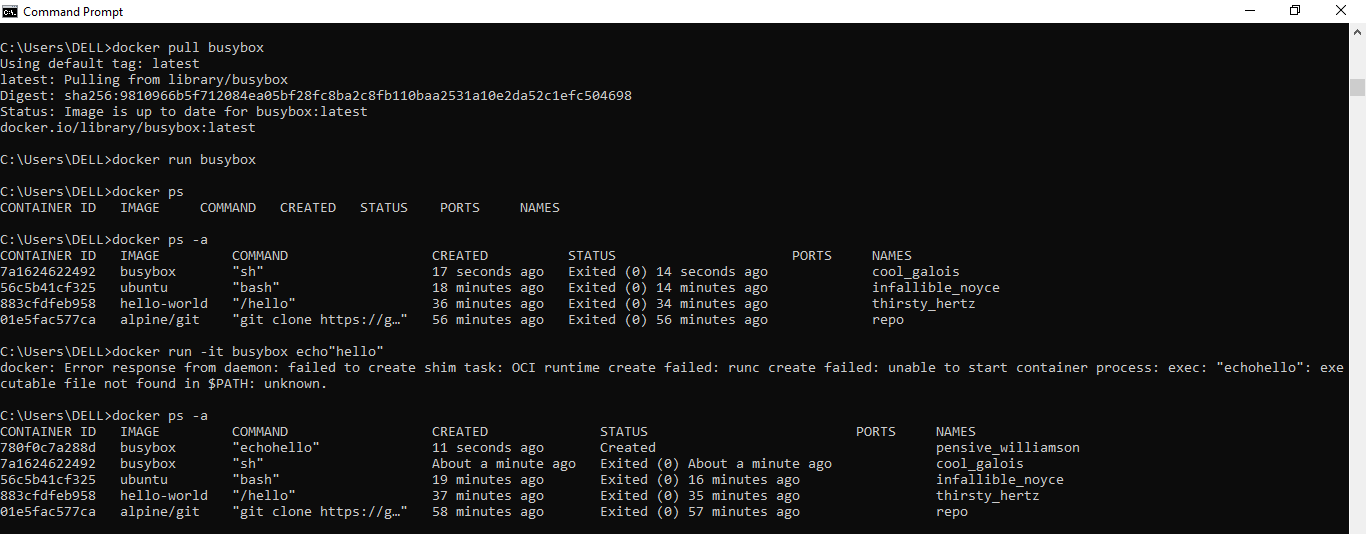
Here we have the output of the container created on the docker hub.

****

Here the output of the shell we have proceed in the Ubuntu image on the docker hub.

****

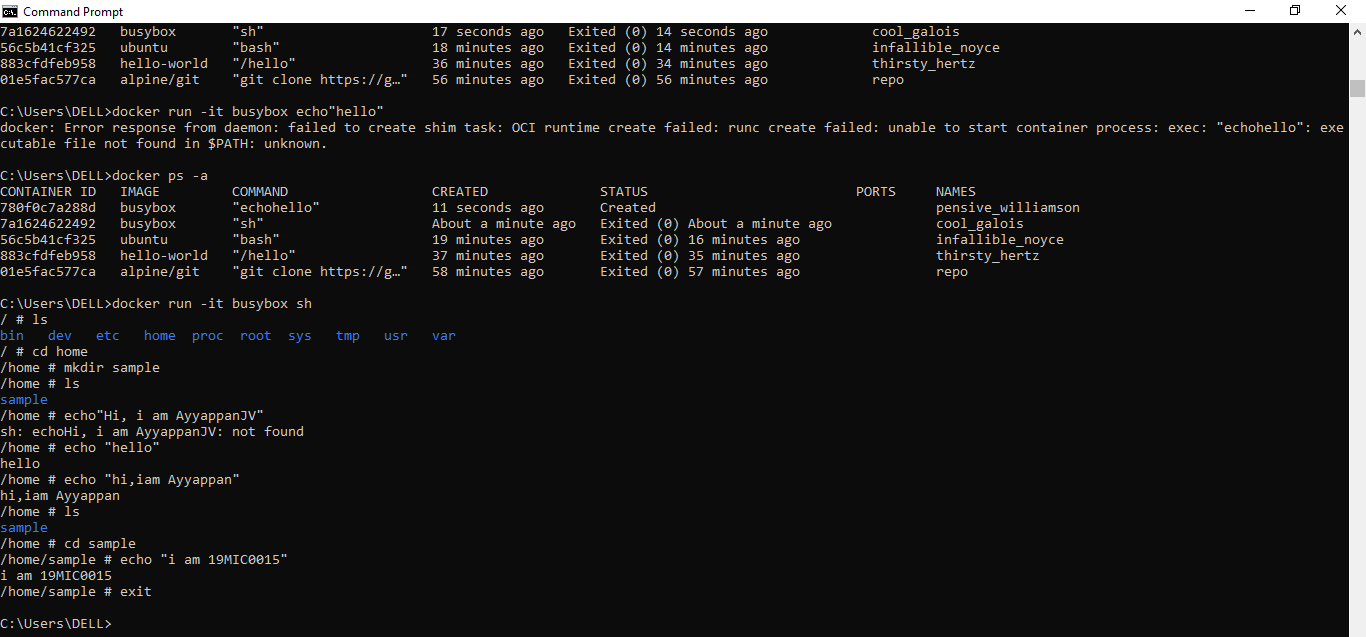
Now we can pull the busy box by using the docker pull busybox , and running the busybox by using the docker run busybox.

****

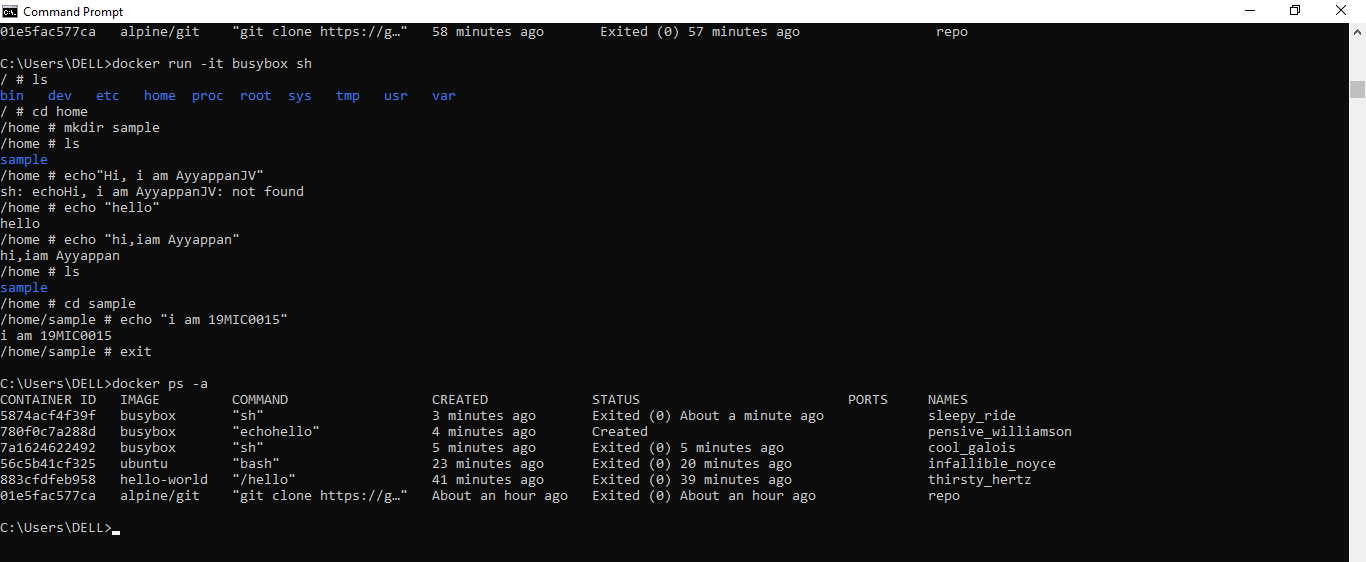
To see the created container we have to type as dokcer ps –a to show the created container on the docker hub.

Here we can display the string using the busy box by typing the docker run –it busybox sh

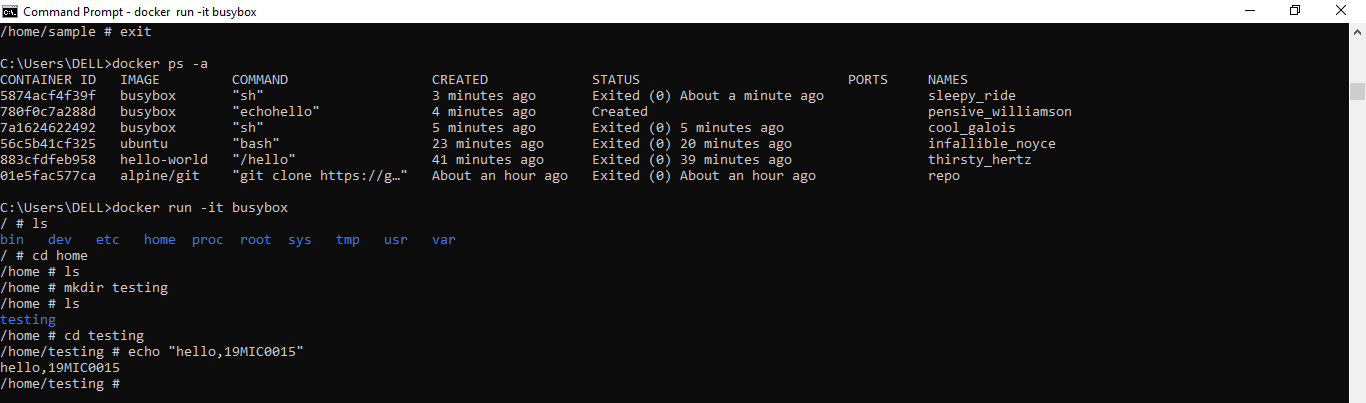
Then it will take you to the busy box shell and type here as echo “some string” then proceed.

****

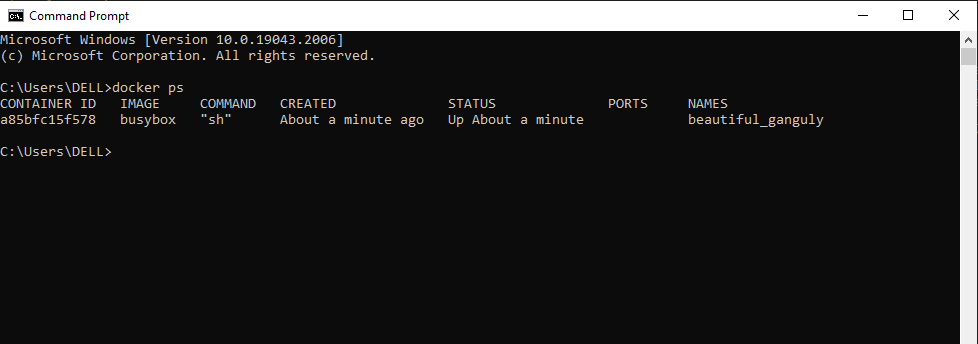
After performing all instructions here we have to see the number of container created on the docker hub by typing the docker ps -a

****

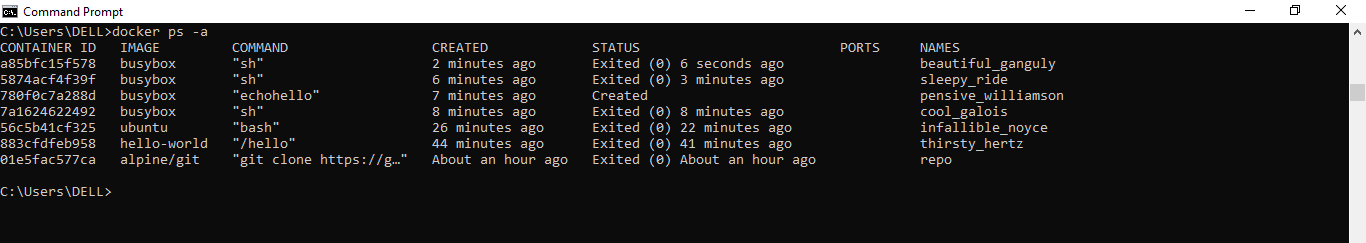
Then we can type docker ps –a tpo display the number of containers created on the docker hub.

****

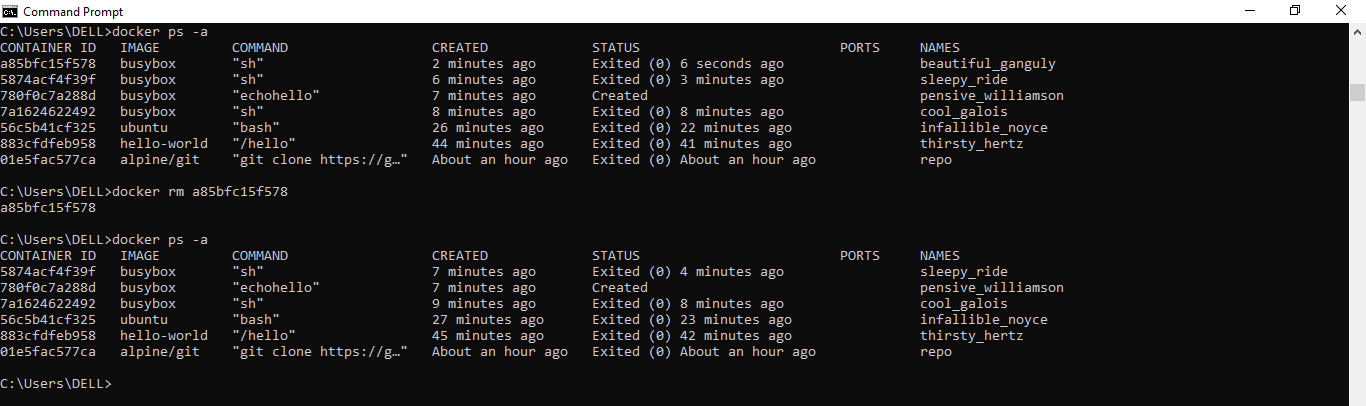
Then go to another cmd and type as docekr ps here we can the created container on the docker hub.

****

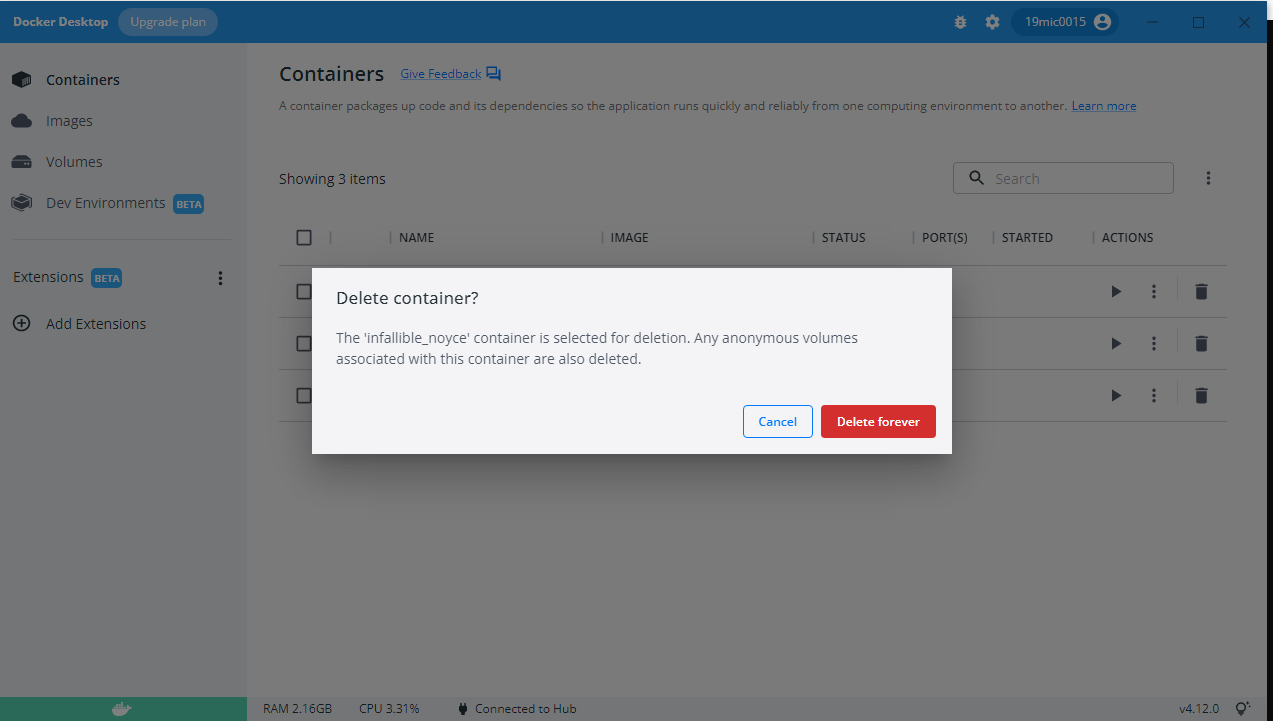
Then type as docker ps –a here we can the number of containers created on the docker.

****

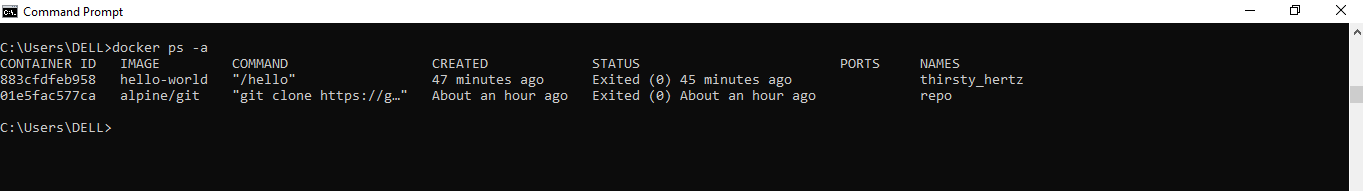
Then to delete the container or any docker resource type as docker rm container\_name.

****

Then we have another option also here we can go to the docker hub and select the container to delete it and click on the delete option.

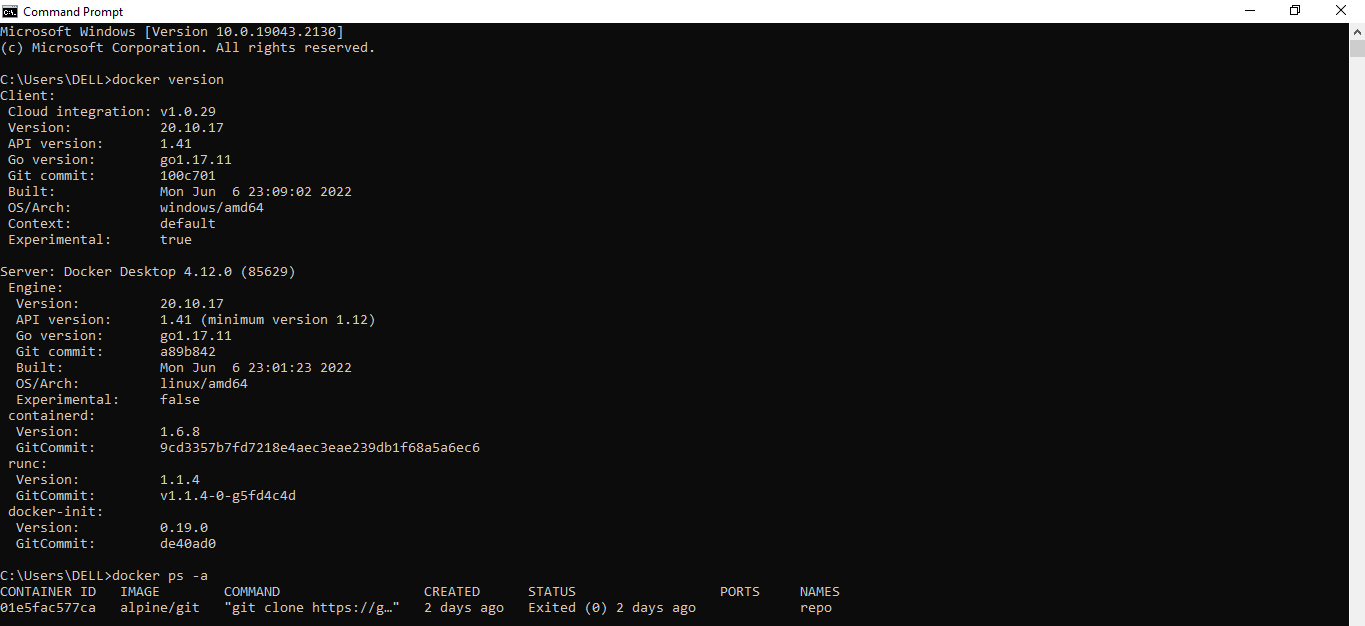
****

Here we can see the seleted container are deleted on the docker hub.

****

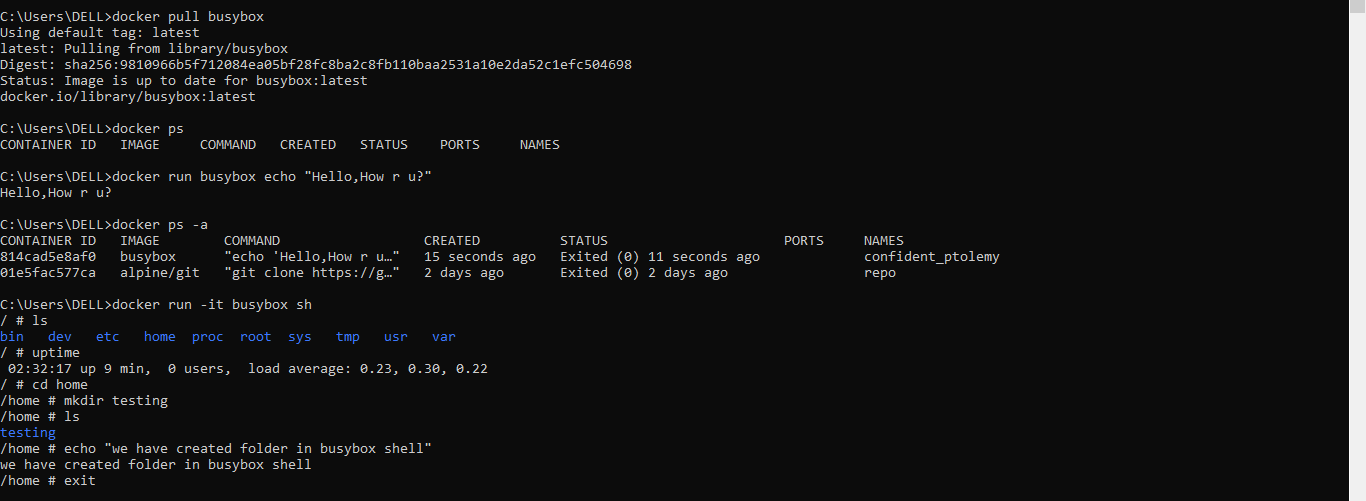
Here we can see the version of the docker.

We need to check whether the client and server side of the dockjer desktop is connected or not we need to verify.

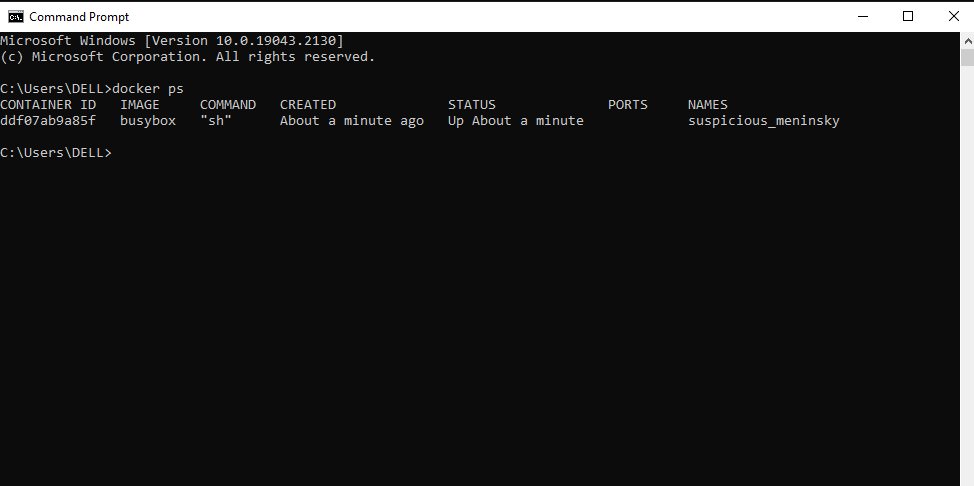
****

Here again we need to pull the busy box using the docker , then type as docker ps –a to check the container is created or not.

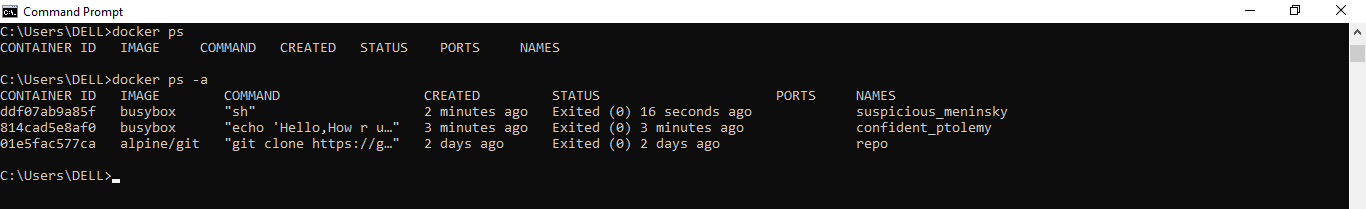
Then run the docker run –it busybox sh it will create the busy box shell here we have created the folder in that shell then printing out the simple string in busybox using the docker.

****

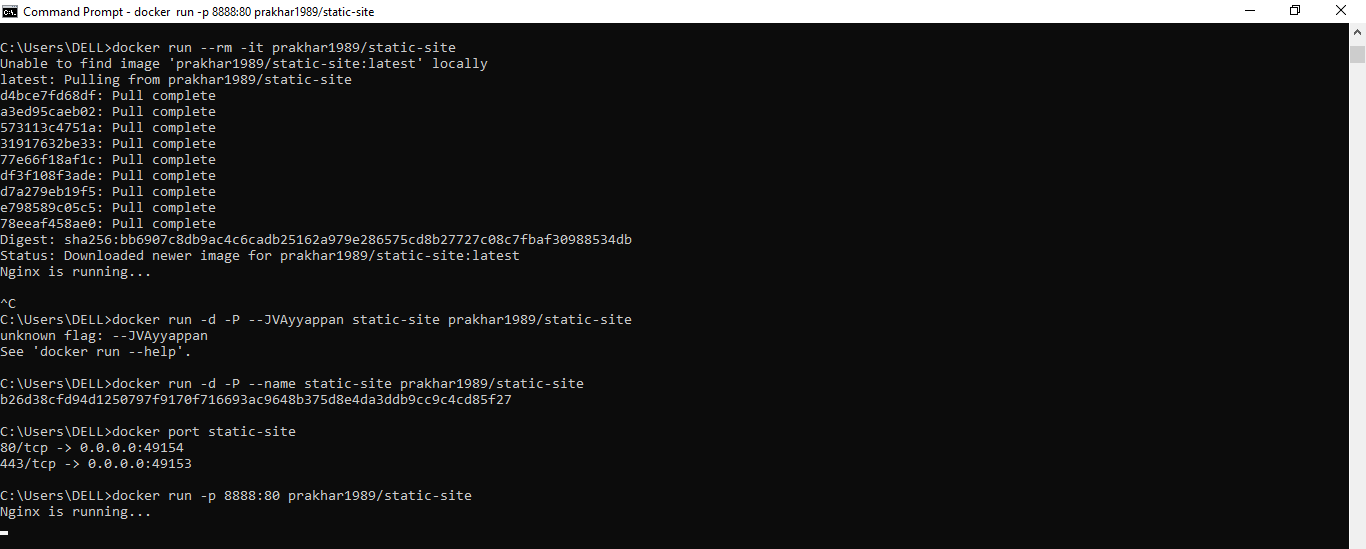
Here open another cmd then typeas docker ps in that we can see the number of container created on the docker hub.

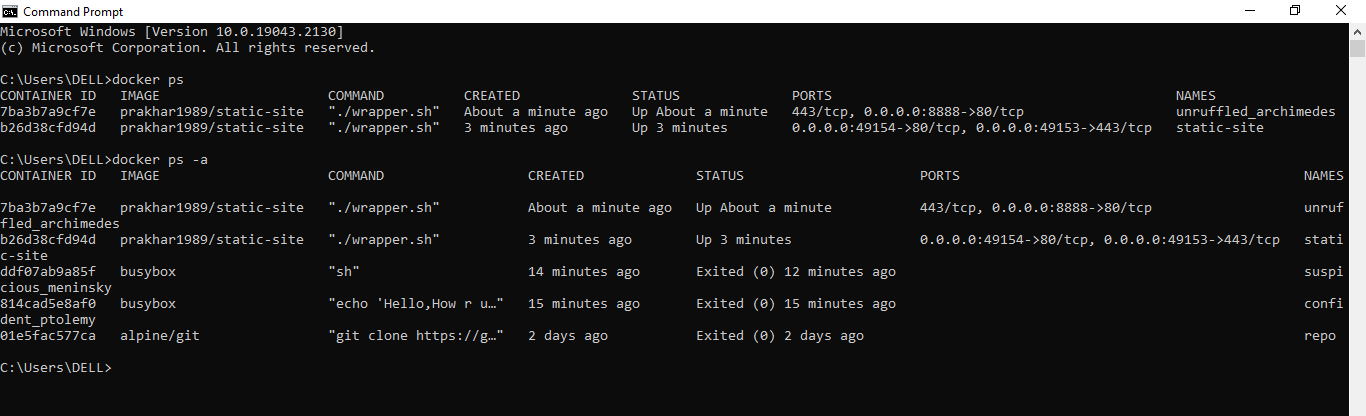
****

Now after executing the busy box then check the number of container available on the docker by using the command as docker ps –a.

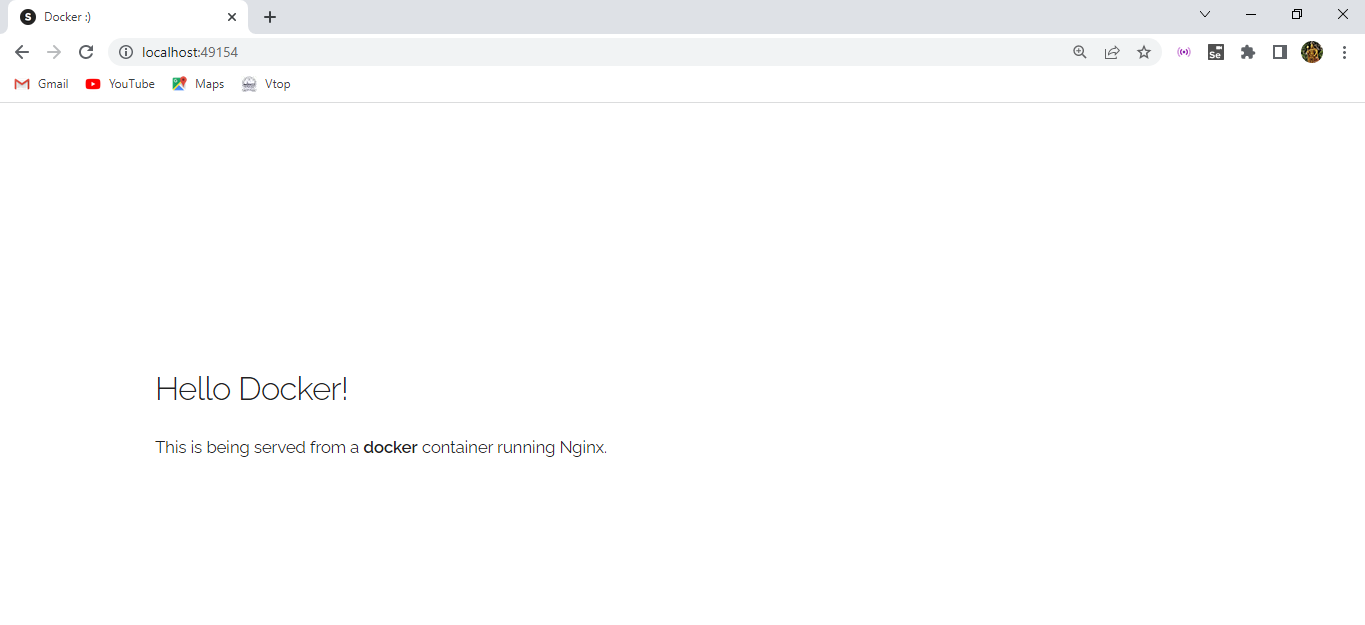
****

Here we can run the static site creation using the docker hub by typing these commands.

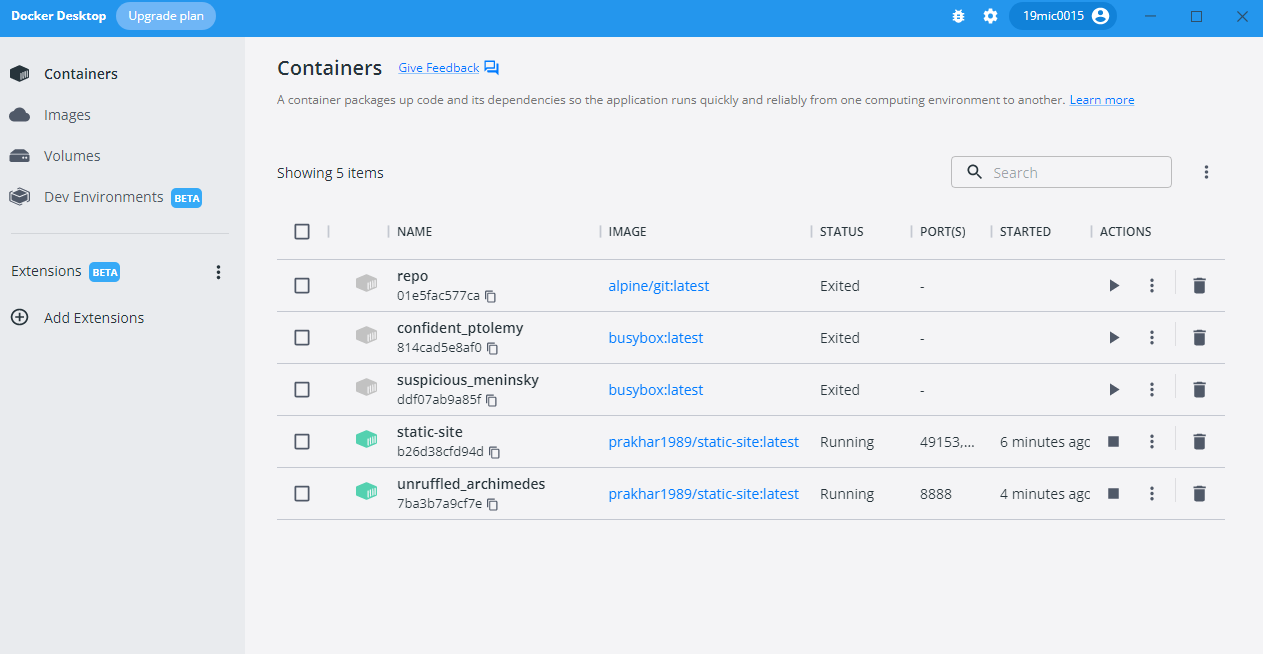
****

****

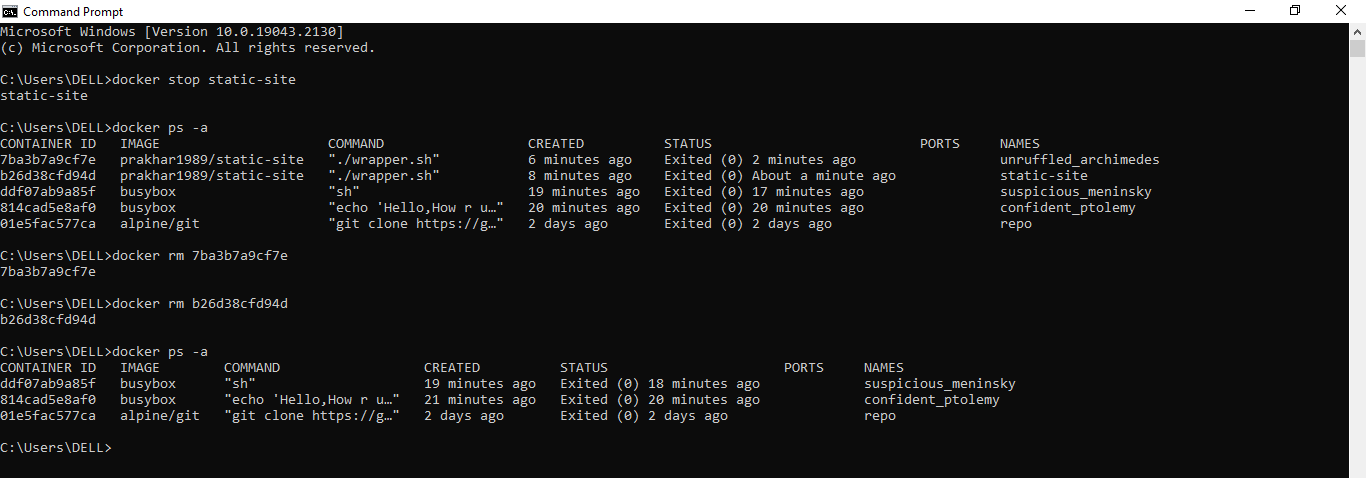
Then follow these all commands to display the static site on our docker hub and we get the output of the site.

****

Here we have number of containers created on the docker hub on the docker hub page.

****

Here, after getting the output of the static site we can stop it, by using docker stop static site cmd to it.

****

**Part -II**

Create a customized docker image using the docker file( Dockerfile).

Demonstrate step by step process for creating and running the image.

Create Docker image for the given Python application (sample application is attached).

Image name should be your register number.

Include screenshots of

Building Image [ After executing build command]

List the newly created Image

Run the image Locally

List Image in the Docker Hub [ After Push]

Pull Image from Docker hub.

**Output :**

**Create a customized docker image using the docker file(Dockerfile).**

**Demonstrateing the step by step process for creating and running the image:-**

**1. Create your own repository in the Docker\_Hub.**

**2. Create the Docker file.**

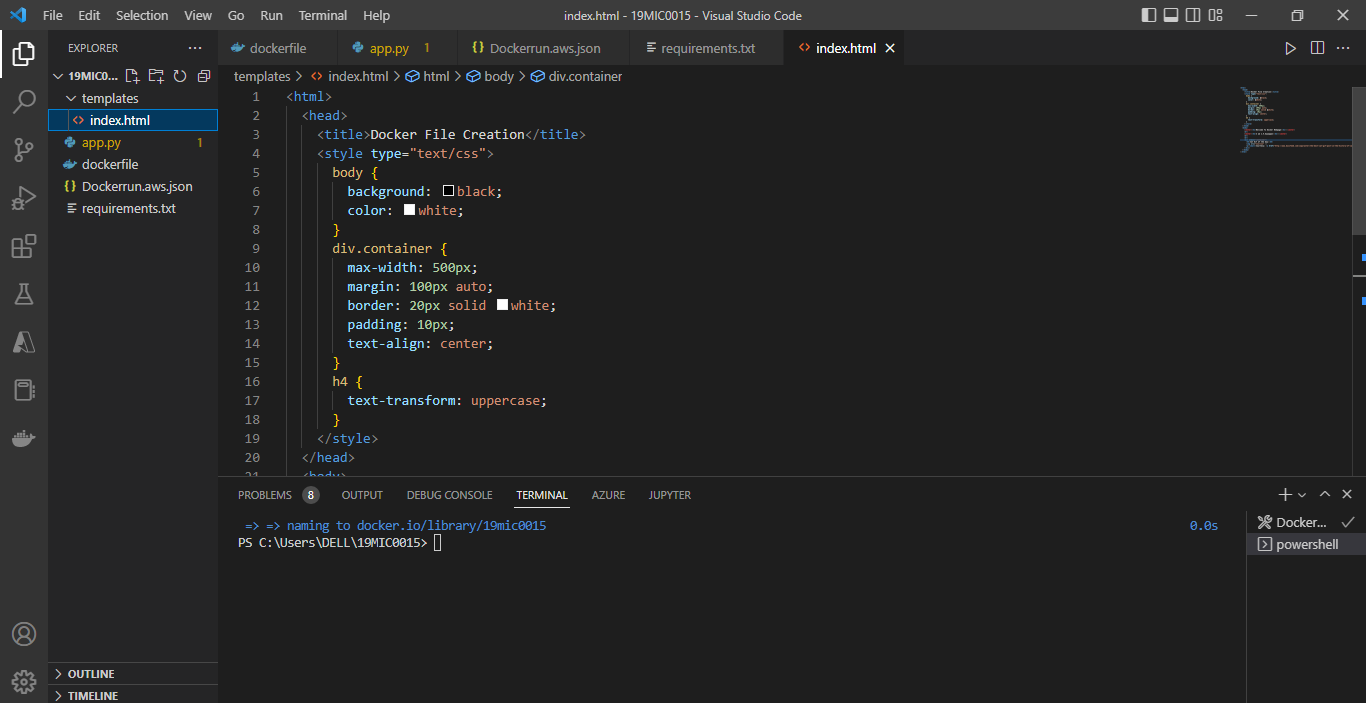
**3. Build the image.**

**4. Run the image locally.**

**5. Push the image to your repository.**

**6. Then push the image globally (By using browser).**

**Index.html :-**

****

<html>

  <head>

    <title>Docker File Creation</title>

    <style type="text/css">

      body {

        background: black;

        color: white;

      }

      div.container {

        max-width: 500px;

        margin: 100px auto;

        border: 20px solid white;

        padding: 10px;

        text-align: center;

      }

      h4 {

        text-transform: uppercase;

      }

    </style>

  </head>

  <body>

    <center><h1>Welcome To Docker Webpage</h1></center>

    <br>

    <center><h2>I am J.V.Ayyappan</h2></center>

    <br>

    <br>

    <div class="container">

      <h4>Cat Gif of the day</h4>

      <img src="{{url}}" />

      <p><small>Courtesy: <a href="http://www.buzzfeed.com/copyranter/the-best-cat-gif-post-in-the-history-of-cat-gifs">Buzzfeed</a></small></p>

    </div>

  </body>

</html>

**DockerFile :-**

FROM python:3.10.8

#Directory for the app

WORKDIR /usr/src/app

#copy the files to container

COPY . .

#installing the required dependencies

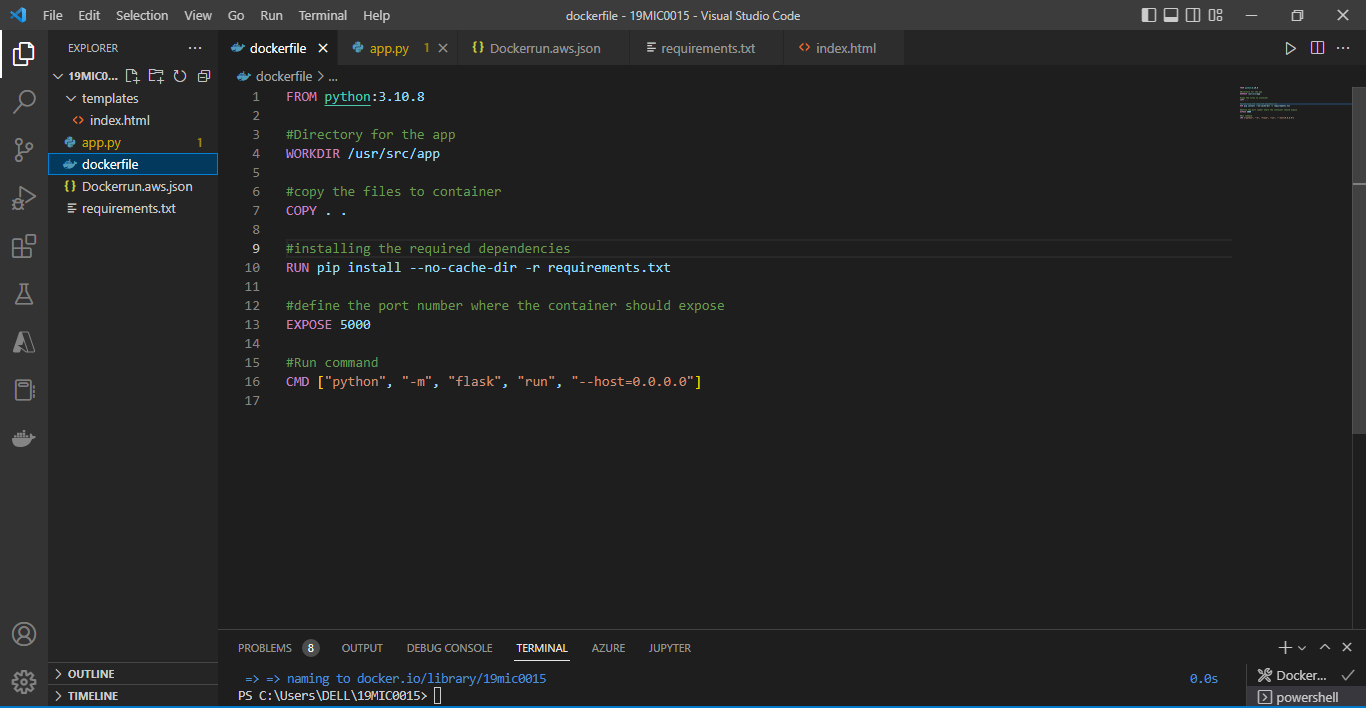
RUN pip install --no-cache-dir -r requirements.txt

#define the port number where the container should expose

EXPOSE 5000

#Run command

CMD ["python", "-m", "flask", "run", "--host=0.0.0.0"]



**App.py :-**

from flask import Flask, render\_template

import os

import random

app = Flask(\_\_name\_\_)

# list of cat images

images = [

    "https://firebasestorage.googleapis.com/v0/b/docker-curriculum.appspot.com/o/catnip%2F0.gif?alt=media&token=0fff4b31-b3d8-44fb-be39-723f040e57fb",

    "https://firebasestorage.googleapis.com/v0/b/docker-curriculum.appspot.com/o/catnip%2F1.gif?alt=media&token=2328c855-572f-4a10-af8c-23a6e1db574c",

    "https://firebasestorage.googleapis.com/v0/b/docker-curriculum.appspot.com/o/catnip%2F10.gif?alt=media&token=647fd422-c8d1-4879-af3e-fea695da79b2",

    "https://firebasestorage.googleapis.com/v0/b/docker-curriculum.appspot.com/o/catnip%2F11.gif?alt=media&token=900cce1f-55c0-4e02-80c6-ee587d1e9b6e",

    "https://firebasestorage.googleapis.com/v0/b/docker-curriculum.appspot.com/o/catnip%2F2.gif?alt=media&token=8a108bd4-8dfc-4dbc-9b8c-0db0e626f65b",

    "https://firebasestorage.googleapis.com/v0/b/docker-curriculum.appspot.com/o/catnip%2F3.gif?alt=media&token=4e270d85-0be3-4048-99bd-696ece8070ea",

    "https://firebasestorage.googleapis.com/v0/b/docker-curriculum.appspot.com/o/catnip%2F4.gif?alt=media&token=e7daf297-e615-4dfc-aa19-bee959204774",

    "https://firebasestorage.googleapis.com/v0/b/docker-curriculum.appspot.com/o/catnip%2F5.gif?alt=media&token=a8e472e6-94da-45f9-aab8-d51ec499e5ed",

    "https://firebasestorage.googleapis.com/v0/b/docker-curriculum.appspot.com/o/catnip%2F7.gif?alt=media&token=9e449089-9f94-4002-a92a-3e44c6bd18a9",

    "https://firebasestorage.googleapis.com/v0/b/docker-curriculum.appspot.com/o/catnip%2F8.gif?alt=media&token=80a48714-7aaa-45fa-a36b-a7653dc3292b",

    "https://firebasestorage.googleapis.com/v0/b/docker-curriculum.appspot.com/o/catnip%2F9.gif?alt=media&token=a57a1c71-a8af-4170-8fee-bfe11809f0b3",

]

@app.route("/")

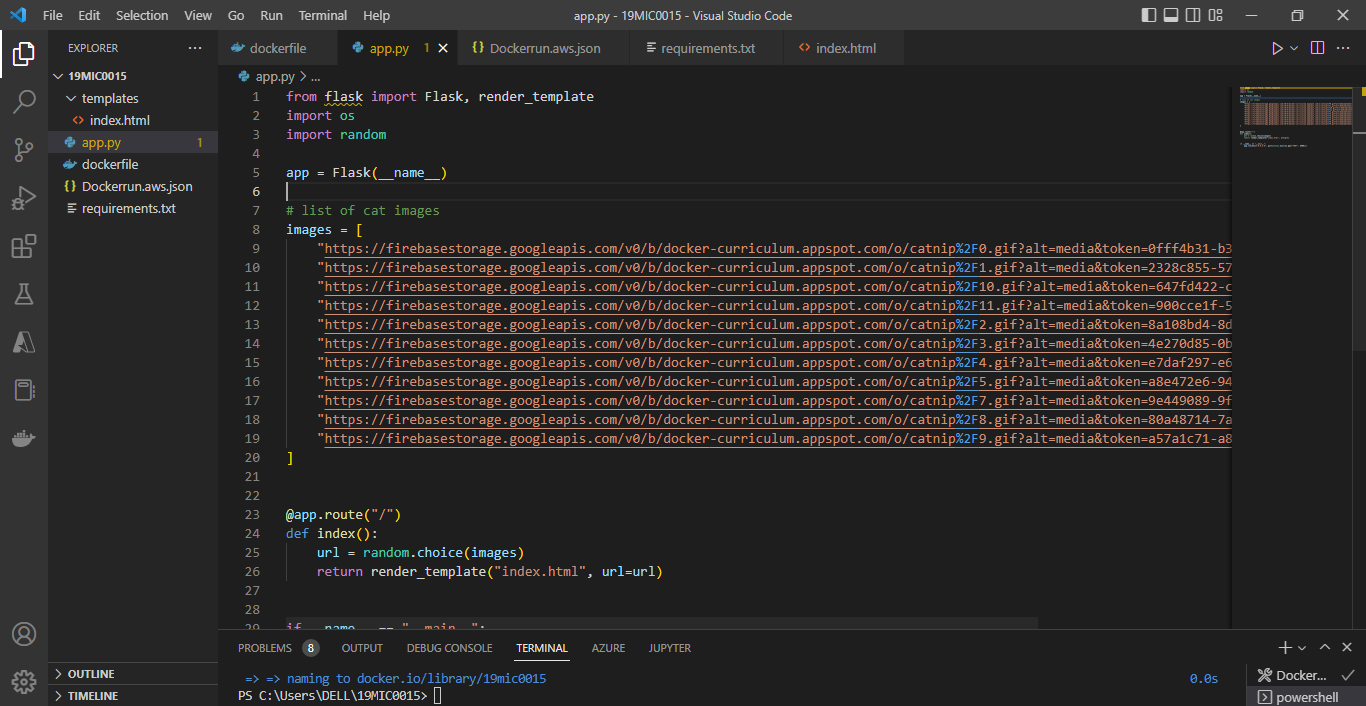
def index():

    url = random.choice(images)

    return render\_template("index.html", url=url)

if \_\_name\_\_ == "\_\_main\_\_":

    app.run(host="0.0.0.0", port=int(os.environ.get("PORT", 5000)))



**Requirement.txt :-**

attrs>=19.1.0

boto3==1.17.23

click>=7.0

itsdangerous==2.0.1

flasgger==0.9.5

Flask==2.0.3

Flask-RESTful>=0.3.6

flask-cors==3.0.8

gunicorn==20.1.0

Jinja2>=2.10.1

jsonschema>=3.0.1,<4.0

marshmallow>=3.0,<=3.6

marshmallow3-annotations>=1.0.0

pytz==2021.1

requests>=2.25.0

requests-aws4auth==1.1.0

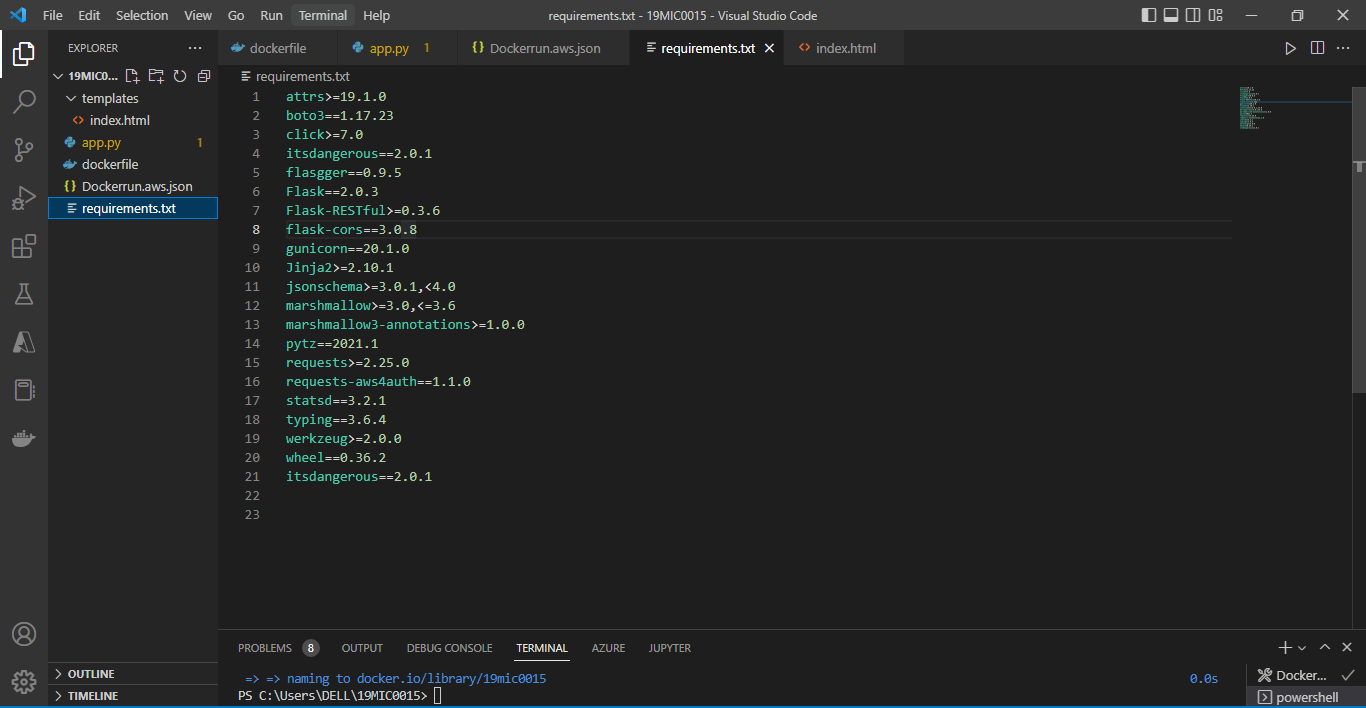
statsd==3.2.1

typing==3.6.4

werkzeug>=2.0.0

wheel==0.36.2

itsdangerous==2.0.1



**Dockerrun.aws.json :-**

{

  "AWSEBDockerrunVersion": "1",

  "Image": {

    "Name": "prakhar1989/catnip",

    "Update": "true"

  },

  "Ports": [

    {

      "ContainerPort": 5000,

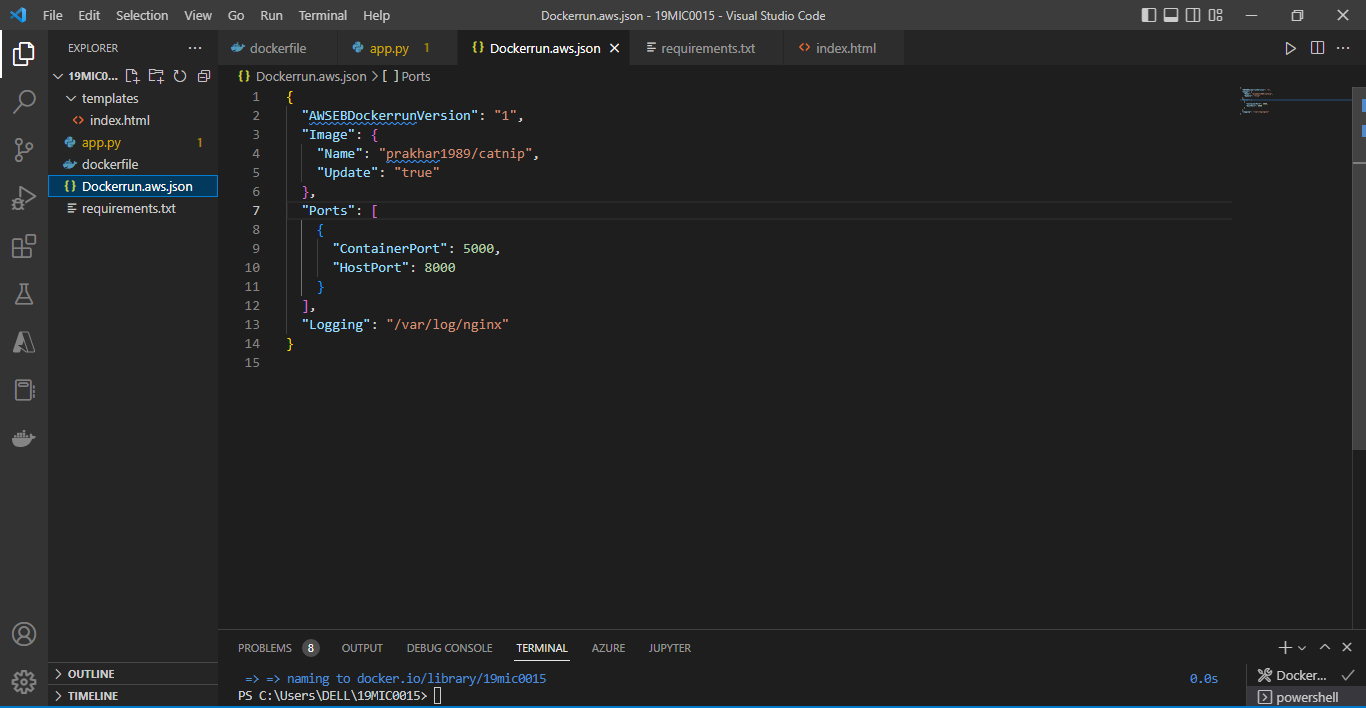
      "HostPort": 8000

    }

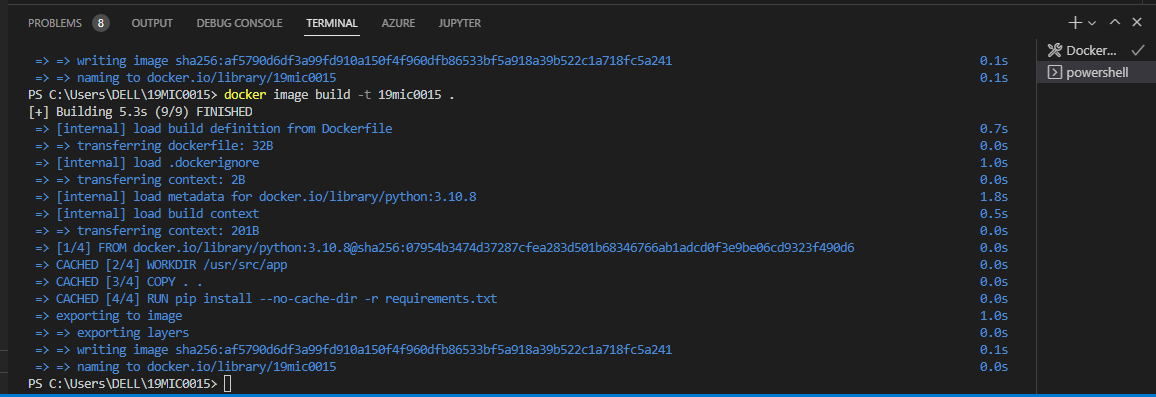
  ],

  "Logging": "/var/log/nginx"

}

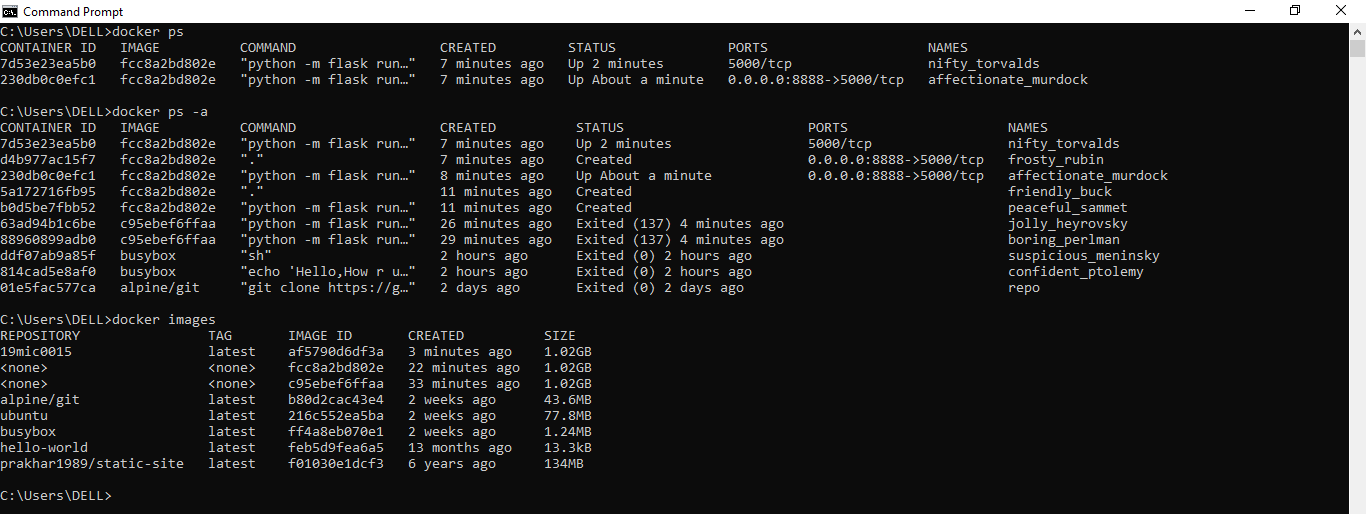
****

After creating the dockerfile, index.html, app.py, requirements.txt and dockerrun.aws.join file go to terminal for executing the docker image globally.

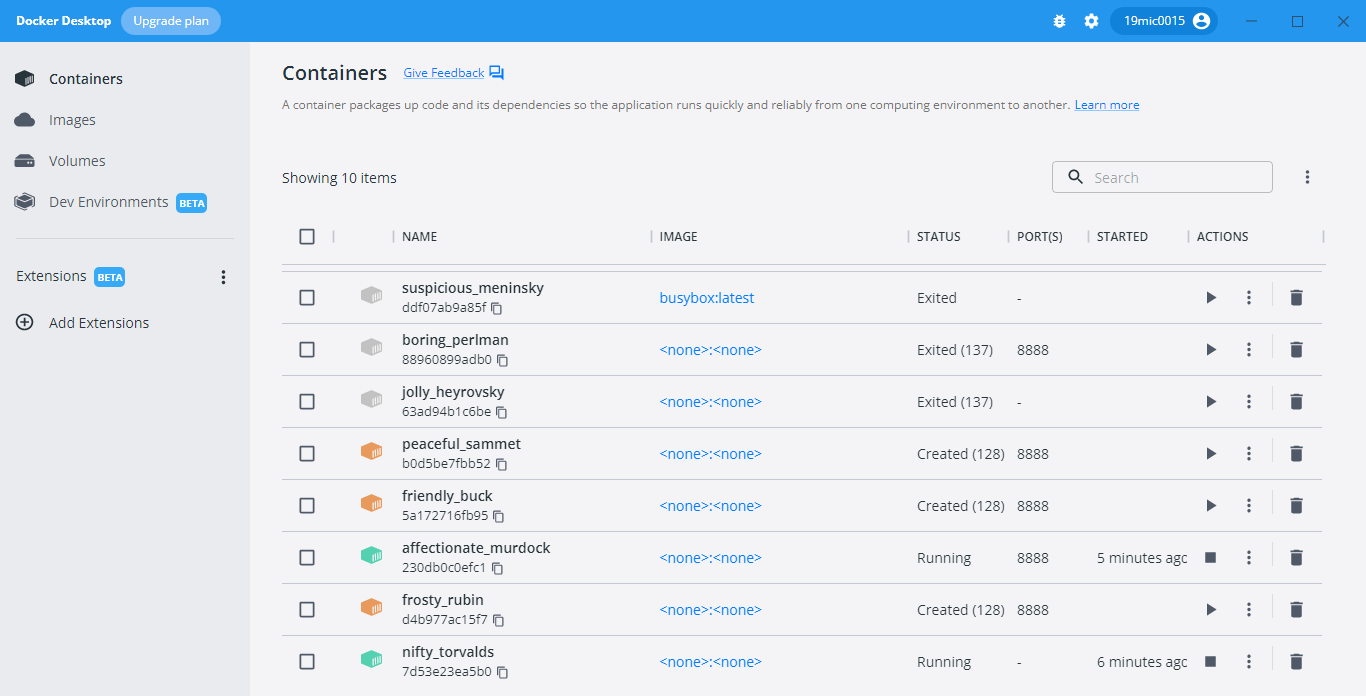
****

Then go to cmd type as docker ps –a to check the number of containers and images created on the docker hub.

Then type as docker image here we can see the docker images in which the python flask is created on locally into our system.

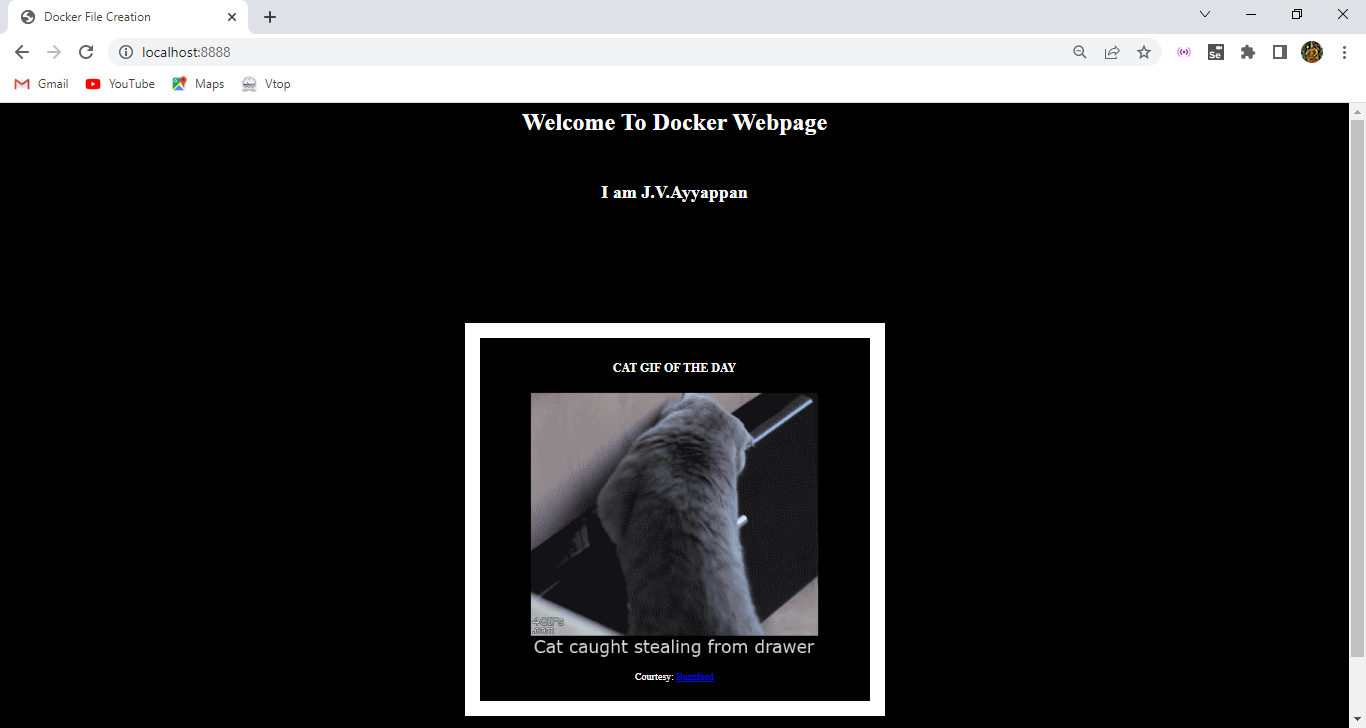
****

Here we can see number of images created using the python flask which on the locally docker\_hub.

****

Here we have the final output of the docker image using the python flask on the docker\_hub

Then the port is to run the docker page is **localhost:8888** we get the output of an docker image.

****