4.2 Push an Image to Docker Hub



This section will guide you to**:**

* Build a custom Docker image using the basic code of Python and push it to Docker Hub.

This guide has three subsections, namely**:**

4.2.1 Preparing a custom Docker image

4.2.2 Pushing the Docker image to Docker Hub

4.2.3 Pushing the code to the GitHub repositories

* *Docker is already installed in your lab. (Refer MEAN: Lab Guide - Phase 4)*

**Step 4.2.1:** Preparing a custom Docker image

* Open the terminal
* Create a directory and write basic Python source code using the procedure given below:

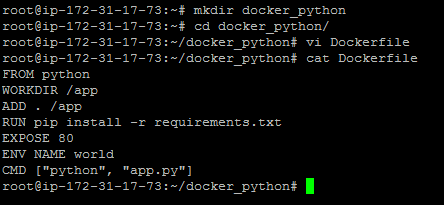
**mkdir docker\_python**

**cd docker\_python/**

**vi Dockerfile**

* Add the code given below to this Dockerfile:

FROM python  
WORKDIR /app  
ADD . /app  
RUN pip install -r requirements.txt  
EXPOSE 80  
ENV NAME world  
CMD [“python”, “app.py”]

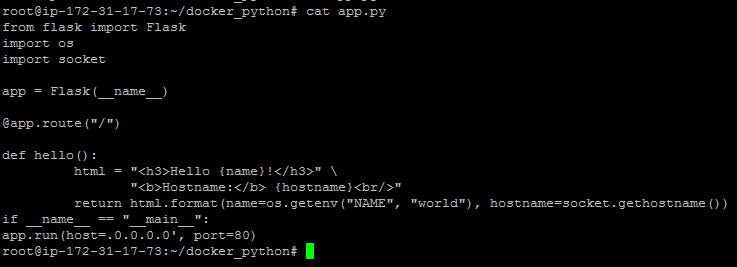


* Create a Python app. Follow the steps below to create an app.py python file:

**vi app.py**

* Add the content below in app.py python source file:

from flask import Flask  
import os  
import socket  
app = Flask(\_\_name\_\_)[@app](http://twitter.com/app).route(“/”)def hello():  
 html = “<h3>Hello {name}!</h3>” \  
 “<b>Hostname:</b> {hostname}<br/>”  
 return html.format(name=os.getenv(“NAME”, “world”), hostname=socket.gethostname())  
if \_\_name\_\_ == “\_\_main\_\_”:  
app.run(host=’0.0.0.0', port=80)

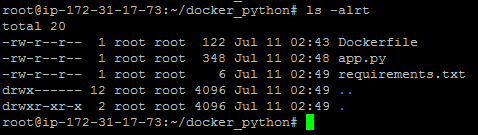


* Create a **requirements.txt** file with the content below:

**vi requirements.txt**

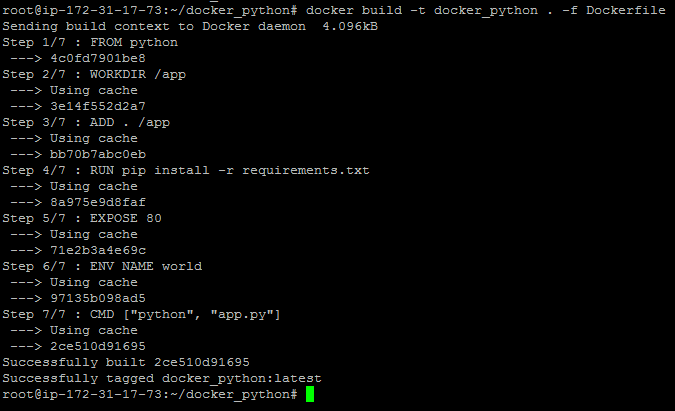
Flask

* Type **ls -alrt**
* You will get the file mentioned below for building a custom Docker image



* Proceed with docker build command to build a custom Docker image

**docker build -t docker\_python . -f Dockerfile**

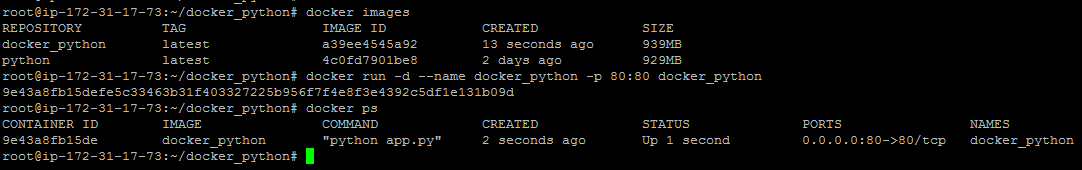


* Once the image is built, check the image using **docker run** command and run it to initialize the custom container on Docker host.

**docker images**

**docker run -d --name docker\_python -p 80:80 docker\_python**

**docker ps**



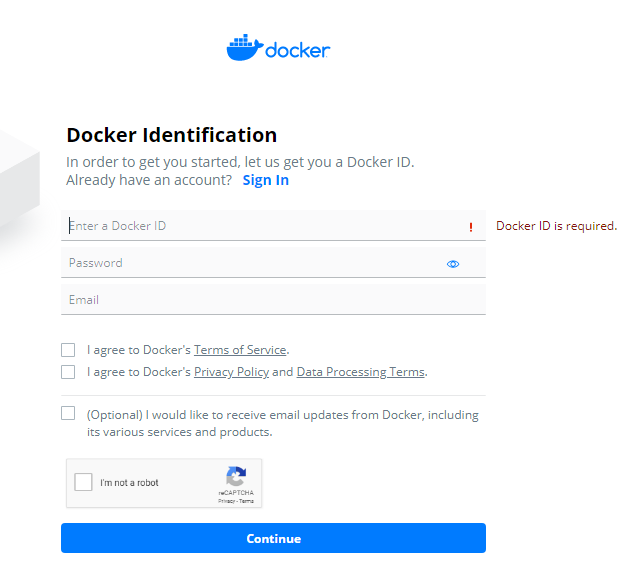
* Once the container is up and running, validate the connectivity using the **curl** command to see if Python code is running on port 80 or not.

**curl localhost**



**Step 4.2.2:** Pushing Docker image to Docker Hub

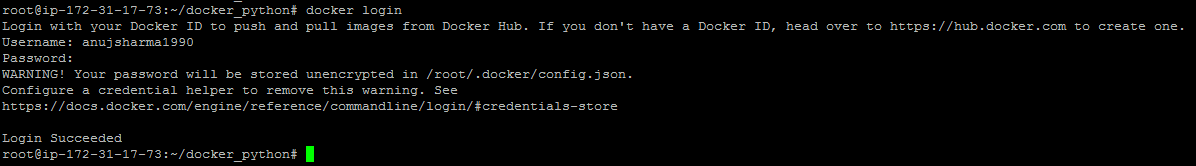
* Once the Docker image is prepared, we need to push this custom Docker image to Docker Hub.
* For pushing the image to Docker Hub, create an account on Docker Hub. Follow a simple sign up process to create a new account.
* Go to <https://hub.docker.com/>
* Enter a username that is also your Docker ID. Your Docker ID must be between 4 and 30 characters long, and can only contain numbers and lowercase letters.
* Enter a unique, valid email address.
* Enter a password between 6 and 128 characters long.
* Click **Sign up**. Docker sends a verification email to the address you provided.
* Click the link in the email to verify your address.



* Once the account is created, we need to login to Docker Hub to push the Docker image to Docker Hub. Type the following command :

**docker login**

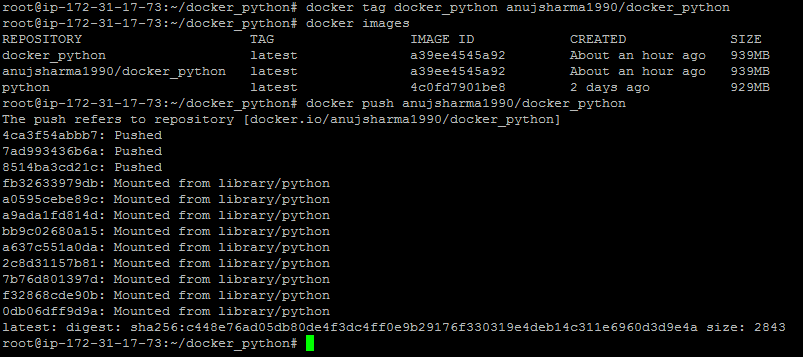
* Type the username and password.



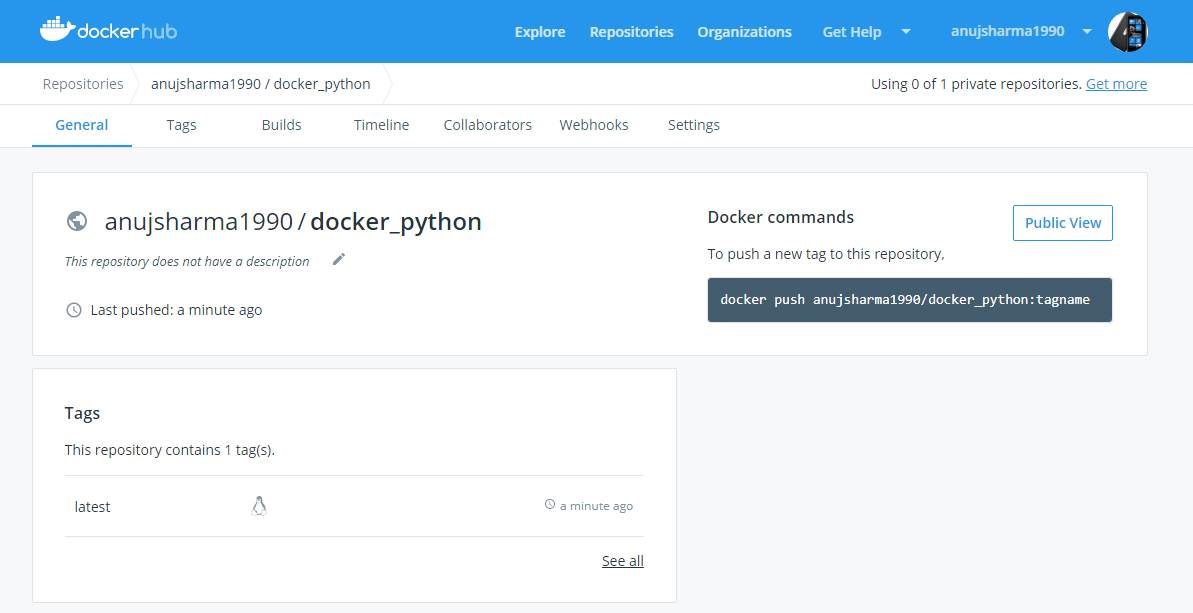
* Create a tag that can be used to push the custom image to Docker Hub.

**docker tag docker\_python anujsharma1990/docker\_python**

**docker push anujsharma1990/docker\_python**



* Login to your Docker Hub account.
* Shown below is the uploaded custom Docker image to Docker Hub:



**Step 4.2.3:** Pushing the code to GitHub repositories

Open your command prompt and navigate to the folder where you have created your files.

cd <folder path>

Initialize your repository using the following command:

git init

Add all the files to your git repository using the following command:

git add . 

Commit the changes using the following command:

git commit . -m “Changes have been committed.”

Push the files to the folder you created initially using the following command:

git push -u origin master