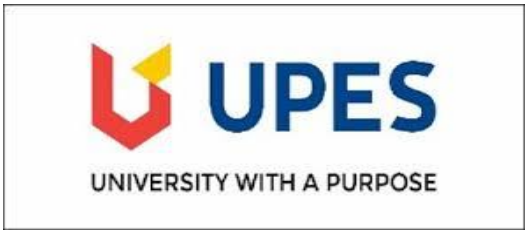


Lab Experiment – 14

Creating a custom image using Dockerfile and uploading to Docker Hub

NAME -Divyaansh Jain

ROLL NO – R171218040

SAP ID – 500067134

COURSE – B. Tech CSE- DevOps Batch 1

SUBJECT – Continuous Integration and Deployment

SEMESTER – 5th semester

Submitted To:

Dr. Hitesh Kumar Sir

Experiment for demonstrating Creating a custom image using Dockerfile and uploading to Docker hub, then using this image creating a container

- Creating a Dockerfile which consists of centos as the main image and I have installed httpd software which is used to display a web page which is copied to the Apache web server root document /var/www/html and exposing port number 80 which is the default port and after all configuration is done, starting the httpd service.

```
File  Edit  View  Search  Terminal  Help
FROM centos
RUN yum install httpd -y
COPY htmlpage /var/www/html
EXPOSE 80
CMD /usr/sbin/httpd -DFOREGROUND
~
```

- The “htmlpage” which I copied to the container is shown below.

```
File  Edit  View  Search  Terminal  Help
<html>
<head>
  <title>TEST WEB SERVER PAGE</title>
</head>

<body style="background-color:grey;">
  <h1>My name is Divyaansh Jain</h1>
</body>

</html
~
```

- Now we will build an image using this Dockerfile via “docker build” command giving an image name and tag.

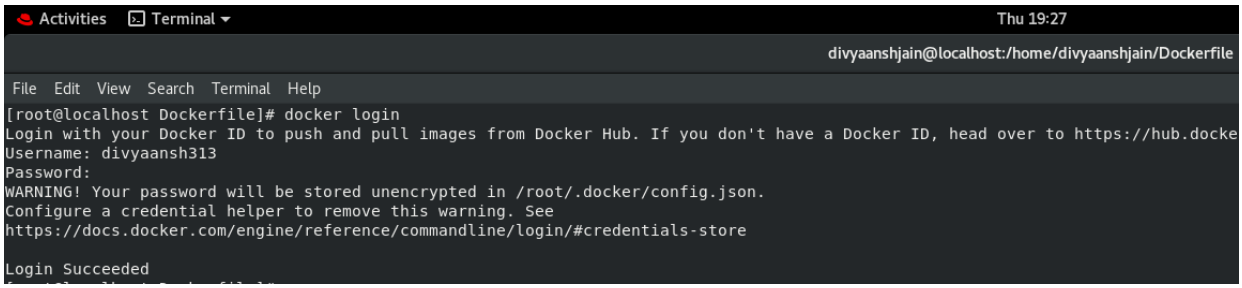
```
[root@localhost Dockerfile]# docker build -t mywebsite:v1 .
Sending build context to Docker daemon  3.072kB
Step 1/5 : FROM centos
 ---> 470671670cac
Step 2/5 : RUN yum install httpd -y
 ---> Running in 593263bc0395
CentOS-8 - AppStream                            117 kB/s | 5.8 MB     00:50
CentOS-8 - Base                                 150 kB/s | 2.2 MB     00:15
CentOS-8 - Extras                               7.7 kB/s | 8.6 kB     00:01
Dependencies resolved.
================================================================================
 Package          Arch   Version                                  Repo       Size
================================================================================
Installing:
 httpd            x86_64 2.4.37-21.module_el8.2.0+494+1df74eae  AppStream 1.7 M
Installing dependencies:
 apr              x86_64 1.6.3-9.el8                           AppStream 125 k
 apr-util         x86_64 1.6.1-6.el8                           AppStream 105 k
 httpd-filesystem noarch 2.4.37-21.module_el8.2.0+494+1df74eae  AppStream  36 k
 httpd-tools      x86_64 2.4.37-21.module_el8.2.0+494+1df74eae  AppStream 103 k
 mod_http2        x86_64 1.11.3-3.module_el8.2.0+486+c01050f0.1 AppStream 156 k
 brotli           x86_64 1.0.6-1.el8                           BaseOS    323 k
 centos-logos-httpd
```

```
Complete!
Removing intermediate container 593263bc0395
 ---> 121a64e8a517
Step 3/5 : COPY htmlpage /var/www/html
 ---> 7c51202729ee
Step 4/5 : EXPOSE 80
 ---> Running in 2ae06d508233
Removing intermediate container 2ae06d508233
 ---> 604a9a18317e
Step 5/5 : CMD /usr/sbin/httpd -DFOREGROUND
 ---> Running in dc262f8724b3
Removing intermediate container dc262f8724b3
 ---> 707e33c1df20
Successfully built 707e33c1df20
Successfully tagged mywebsite:v1
[root@localhost Dockerfile]#
```

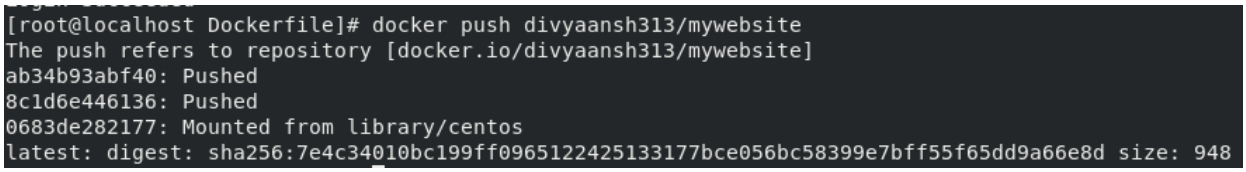
- For uploading this image to Docker Hub, it should be properly formatted and this can be done with “docker tag” command.

```
File  Edit  View  Search  Terminal  Help
[root@localhost divyaanshjain]# docker tag mywebsite:v1 divyaansh/mywebsite
```

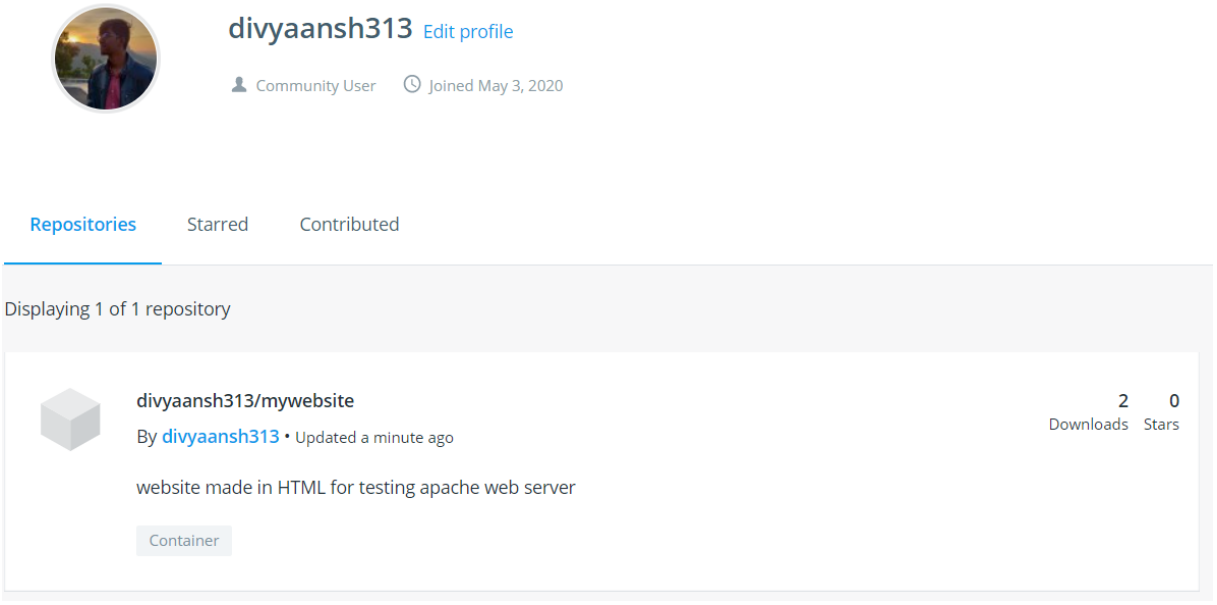
- For pushing your images to Docker Hub, you should be logged in to your account and this can be done using “docker login” command.



- Now you can push your images in your account by using “docker push” command.



- Go to Docker Hub and login to see if your image has been uploaded.



- Fetch that image into another system using command “docker pull”.

```
File  Edit  View  Search  Terminal  Help
[root@localhost Dockerfile]# docker pull divyaansh313/mywebsite
Using default tag: latest
latest: Pulling from divyaansh313/mywebsite
Digest: sha256:7e4c34010bc199ff0965122425133177bce056bc58399e7bff55f65dd9a66e8d
Status: Image is up to date for divyaansh313/mywebsite:latest
docker.io/divyaansh313/mywebsite:latest
```

- Run a container using this image and attach a port, in this case 8050 to the webserver’s port using “docker run -p” command.

```
File  Edit  View  Search  Terminal  Help
[root@localhost Dockerfile]# docker run -d -p 8050:80 divyaansh313/mywebsite
5928124b75973710141fe766b8f282b5060769f052d47480b40454216067a83d
```

- After the container runs successfully, use your browser to view the web page.