Table of Contents:

Step -1 : Update the ubuntu index by using update command

$ sudo apt-get update

```
dyadav@ubuntu:~$ sudo apt-get update
[sudo] password for dyadav:
Hit:1 http://ports.ubuntu.com/ubuntu-ports jammy InRelease
Get:2 http://ports.ubuntu.com/ubuntu-ports jammy-updates InRelease [128 kB]
Get:3 http://ports.ubuntu.com/ubuntu-ports jammy-backports InRelease [127 kB]
Get:4 http://ports.ubuntu.com/ubuntu-ports jammy-security InRelease [129 kB]
Get:5 http://ports.ubuntu.com/ubuntu-ports jammy-updates/main arm64 Packages [1,600 kB]
Get:6 http://ports.ubuntu.com/ubuntu-ports jammy-updates/main Translation-en [333 kB]
Get:7 http://ports.ubuntu.com/ubuntu-ports jammy-updates/restricted arm64 Packages [1,688 kB]
Get:8 http://ports.ubuntu.com/ubuntu-ports jammy-updates/restricted Translation-en [373 kB]
Ign:9 https://pkg.jenkins.io/debian binary/ InRelease
Hit:10 https://pkg.jenkins.io/debian binary/ Release
Get:12 http://ports.ubuntu.com/ubuntu-ports jammy-updates/universe arm64 Packages [1,052 kB]
Get:13 http://ports.ubuntu.com/ubuntu-ports jammy-updates/universe Translation-en [258 kB]
Get:14 http://ports.ubuntu.com/ubuntu-ports jammy-backports/universe arm64 Packages [27.1 kB]
Get:15 http://ports.ubuntu.com/ubuntu-ports jammy-backports/universe Translation-en [16.5 kB]
Get:16 http://ports.ubuntu.com/ubuntu-ports jammy-security/main arm64 Packages [1,401 kB]
Get:17 http://ports.ubuntu.com/ubuntu-ports jammy-security/main Translation-en [276 kB]
Get:18 http://ports.ubuntu.com/ubuntu-ports jammy-security/restricted arm64 Packages [1,639 kB]
Get:19 http://ports.ubuntu.com/ubuntu-ports jammy-security/restricted Translation-en [363 kB]
Get:20 http://ports.ubuntu.com/ubuntu-ports jammy-security/universe arm64 Packages [835 kB]
Fetched 10.2 MB in 6s (1,792 kB/s)
Reading package lists... Done
dyadav@ubuntu:~$
```

Step -2 : Install the Docker in Ubuntu Linux Machine.

$ sudo apt-get install docker.io

```
dyadav@ubuntu:~$
dyadav@ubuntu:~$ sudo apt-get install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd docker.io pigz runc ubuntu-fan
0 upgraded, 6 newly installed, 0 to remove and 50 not upgraded.
Need to get 56.2 MB of archives.
After this operation, 231 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ports.ubuntu.com/ubuntu-ports jammy/universe arm64 pigz arm64 2.6-1 [55.7 kB]
Get:2 http://ports.ubuntu.com/ubuntu-ports jammy/main arm64 bridge-utils arm64 1.7-1ubuntu3 [34.4 kB]
Get:3 http://ports.ubuntu.com/ubuntu-ports jammy-updates/main arm64 runc arm64 1.1.12-0ubuntu2~22.04.1 [7,816 kB]
Get:4 http://ports.ubuntu.com/ubuntu-ports jammy-updates/main arm64 containerd arm64 1.7.12-0ubuntu2~22.04.1 [27.1 MB]
Get:5 http://ports.ubuntu.com/ubuntu-ports jammy-updates/universe arm64 docker.io arm64 24.0.7-0ubuntu2~22.04.1 [21.2 MB]
Get:6 http://ports.ubuntu.com/ubuntu-ports jammy/universe arm64 ubuntu-fan all 0.12.16 [35.2 kB]
Fetched 56.2 MB in 15s (3,851 kB/s)
Preconfiguring packages ...
Selecting previously unselected package pigz.
(Reading database ... 211453 files and directories currently installed.)
Preparing to unpack .../0-pigz_2.6-1_arm64.deb ...
Unpacking pigz (2.6-1) ...
Selecting previously unselected package bridge-utils.
Preparing to unpack .../1-bridge-utils_1.7-1ubuntu3_arm64.deb ...
Unpacking bridge-utils (1.7-1ubuntu3) ...
Selecting previously unselected package runc.
Preparing to unpack .../2-runc_1.1.12-0ubuntu2~22.04.1_arm64.deb ...
Unpacking runc (1.1.12-0ubuntu2~22.04.1) ...
Selecting previously unselected package containerd.
Preparing to unpack .../3-containerd_1.7.12-0ubuntu2~22.04.1_arm64.deb ...
```

Step-3 : Verify the docker Version by running

$ docker –version : Check the version, Also add your current user to docker Group.

$ systemctl status docker : to check the Docker Service Status.

```
dyadav@ubuntu:~$ systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2024-07-21 14:17:49 UTC; 7min ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 23383 (dockerd)
      Tasks: 9
     Memory: 945.9M
        CPU: 24.114s
     CGroup: /system.slice/docker.service
             └─23383 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

$ usermod -aG docker $USER : this command will add your user into Docker Group

$ cat /etc/group : now your username is a part of the docker group.

You will see this error message before adding your user to docker group.

```
dyadav@ubuntu:~$ docker ps
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/containers/json": dial unix /var/run/docker.sock: connect: permission denied
dyadav@ubuntu:~$ cat /etc/group
```

```
docker:x:138:dyadav
dyadav@ubuntu:~$
```

$ newgrp docker : to activate the new changes hence no need to restart the Ubuntu Linux Machine.

```
dyadav@ubuntu:~$ docker --version
Docker version 24.0.7, build 24.0.7-0ubuntu2~22.04.1
dyadav@ubuntu:~$
dyadav@ubuntu:~$
dyadav@ubuntu:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS              PORTS                  NAMES
4d55983a2d20   mysql:latest   "docker-entrypoint.s…"   About a minute ago   Up About a minute   3306/tcp, 33060/tcp   mysql
dyadav@ubuntu:~$
dyadav@ubuntu:~$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS              PORTS                  NAMES
4d55983a2d20   mysql:latest   "docker-entrypoint.s…"   About a minute ago   Up About a minute   3306/tcp, 33060/tcp   mysql
dyadav@ubuntu:~$
```

Step-4 : Pull the Public docker image from dockerHub

$ docker pull mysql:latest :--> To pull the mysql docker image from dockerhub

```
dyadav@ubuntu:~$ docker pull mysql:latest
latest: Pulling from library/mysql
c72f53f7235b: Pull complete
c7e4ed755af2: Pull complete
6c8c802f90bc: Pull complete
eecc55f854cd: Pull complete
cc8dabc09813: Pull complete
3a27c3f0dbd7: Pull complete
a55978eb4258: Pull complete
767d62f87325: Pull complete
afe5d39ea75c: Pull complete
72a1a98f7aad: Pull complete
Digest: sha256:72a37ddc9f839cfd84f1f6815fb31ba26f37f4c200b90e49607797480e3be446
Status: Downloaded newer image for mysql:latest
docker.io/library/mysql:latest
```

$ docker pull nginx:latest :----> To pull the nginx docker image from dockerhub

```
dyadav@ubuntu:~$ docker pull nginx:latest
latest: Pulling from library/nginx
ea235d1ccf77: Pull complete
e29cef106877: Pull complete
e9bf20d5335e: Pull complete
1394e86b8f58: Pull complete
7b2b3e0f512f: Pull complete
6a11b5a77155: Pull complete
fb6d6e4aad9c: Pull complete
Digest: sha256:67682bda769fae1ccf5183192b8daf37b64cae99c6c3302650f6f8bf5f0f95df
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
```

Once it's pulled successfully, Use the below command to check docker images.

$ docker images

$ docker rmi <ImageID> : To remove the Image

```
dyadav@ubuntu:~$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
mysql        latest    c3239d2b6d88   2 weeks ago    608MB
nginx        latest    443d199e8bfc   4 weeks ago    193MB
dyadav@ubuntu:~$
```

Note : we create an image from file & container from build Image.

File —-> Image —-> Container

$ docker rmi -f $(docker images -aq) : To Delete all the docker images at Once.

Step-5 : Run the Mysql Container from the pulled image.

$ docker run -d --name mysql -e MYSQL_ROOT_PASSWORD=root mysql:latest

-d : run the container in detached mode

-e : give the environment variable

```
dyadav@ubuntu:~$ docker run -d --name mysql -e MYSQL_ROOT_PASSWORD=root mysql:latest
31b0a5b9b50cb51181f43aa40d044b541b7ebbfb83dd0c2012ba366a530e28c4
dyadav@ubuntu:~$
dyadav@ubuntu:~$
dyadav@ubuntu:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS                  NAMES
31b0a5b9b50c   mysql:latest   "docker-entrypoint.s…"   2 seconds ago   Up 2 seconds   3306/tcp, 33060/tcp   mysql
```

Use the below command to get into mysql Container
docker exec -it <<Container ID >>bash

```
dyadav@ubuntu:~$ docker exec -it 31b0a5b9b50c bash
bash-5.1#
bash-5.1#
bash-5.1#
bash-5.1# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 9.0.0 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

use below command to check the databases & create it.
show databases;
create database devops;

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.01 sec)

mysql> create database devops;
Query OK, 1 row affected (0.00 sec)

mysql> create database Test;
Query OK, 1 row affected (0.01 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| Test               |
| devops             |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
6 rows in set (0.00 sec)
```

Challenge: Once you delete the Container, you lose the Database as well.

Solution: Add the Volume & bind the volume with Container.

docker volume for persistent :

$ docker volume ls : List down all the Volume

$ docker volume create mqsql-volume : it will create a volume

```
dyadav@ubuntu:~$ docker volume ls
DRIVER    VOLUME NAME
local     295a1d86d354cc23879b8ed4e7c61843da680b123d8c89840b40a2fd525c209c
local     f69e8d4fe1fd8d39f3c5cac8cccc9cce0d3d760008b41542eaea4ff85ca756c3
dyadav@ubuntu:~$ docker volume create mqsql-volume
mqsql-volume
dyadav@ubuntu:~$ docker volume ls
DRIVER    VOLUME NAME
local     295a1d86d354cc23879b8ed4e7c61843da680b123d8c89840b40a2fd525c209c
local     f69e8d4fe1fd8d39f3c5cac8cccc9cce0d3d760008b41542eaea4ff85ca756c3
local     mqsql-volume
dyadav@ubuntu:~$
```

$ docker volume remove <Volume-Name> : To remove the created docker Volume.

$ docker volume inspect mqsql-volume : to check the docker volume path

$ docker volume prune : to Delete the non-Use Volume.

Mysql default path : var/lib/docker/volumes/mqsql-volume

```
dyadav@ubuntu:~$ docker volume inspect mqsql-volume
[
    {
        "CreatedAt": "2024-07-21T15:14:02Z",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/mqsql-volume/_data",
        "Name": "mqsql-volume",
        "Options": null,
        "Scope": "local"
    }
]
```

Now create a docker image with Volume.

$ docker run -d --name mysql -v mqsql-volume:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=root mysql:latest

- v : for Volume & give volume Name:**/<<mysql database path>>**

/var/lib/mysql : is the default mysql database path

```
dyadav@ubuntu:~$ docker run -d --name mysql -v mqsql-volume:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=root mysql:latest
6aafb392e1b2484d9bd0fb3f21a5a0d6a9cc73b8d56ce6a0a717041cc7c441ad
dyadav@ubuntu:~$
dyadav@ubuntu:~$
dyadav@ubuntu:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS                                   NAMES
6aafb392e1b2   mysql:latest   "docker-entrypoint.s…"   7 seconds ago   Up 6 seconds   3306/tcp, 33060/tcp                     mysql
6319692e076d   nginx:latest   "/docker-entrypoint.…"   2 hours ago     Up 2 hours     0.0.0.0:8000->80/tcp, :::8000->80/tcp   nginx
dyadav@ubuntu:~$
```

===> Created 2 Database

```
mysql> create database DevOps;
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> create database KLM-Airline;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near
 '-Airline' at line 1
mysql> create database KLM_Airline;
Query OK, 1 row affected (0.00 sec)

mysql>
mysql>
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| DevOps             |
| KLM_Airline        |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
6 rows in set (0.00 sec)
```

===> Deleted the Container & built the container again.

```
dyadav@ubuntu:~$ docker stop 6aafb392e1b2 && docker rm 6aafb392e1b2
6aafb392e1b2
6aafb392e1b2
dyadav@ubuntu:~$
```

===> Created the Container again & we can see our old created database.

```
dyadav@ubuntu:~$ docker run -d --name mysql -v mqsql-volume:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=root mysql:latest
3a4486b4759a101909808e39e81d9cb95622ad2748675c08f080325ae54fa719
dyadav@ubuntu:~$
dyadav@ubuntu:~$
dyadav@ubuntu:~$ docker exec -it 3a4486b4759 bash
bash-5.1# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 9.0.0 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
mysql>
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| DevOps             |
| KLM_Airline        |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
6 rows in set (0.01 sec)

mysql>
```

Step-6 : Run the nginx Container from the pulled image.

Verify the container using below commands:

$ docker ps : shows all the running containers

$ docker ps -a : shows running & stopped both.

```
dyadav@ubuntu:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS                 NAMES
31b0a5b9b50c   mysql:latest   "docker-entrypoint.s…"   7 minutes ago   Up 7 minutes   3306/tcp, 33060/tcp   mysql
```

==> Install the nginx container:

$ docker run -d --name nginx -p 8000:80 nginx:latest

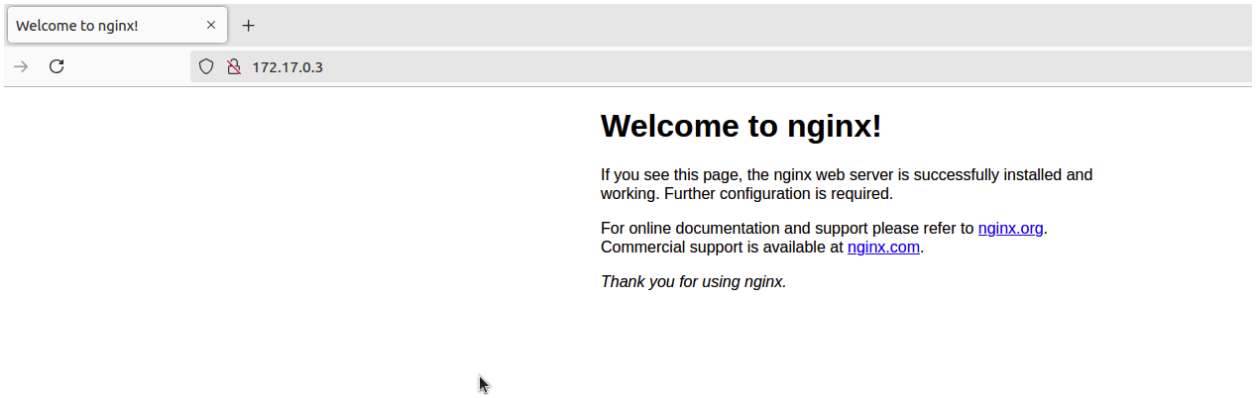
- p : publish port 8000 “Local Host Port” & 80 “nginx Container port.

```
dyadav@ubuntu:~$ docker run -d --name nginx -p 8000:80 nginx:latest
8cca3af1c5fed601f72ac07d0d3cdcf0a7f1a23f202ee780e8870cb35fd5f0f8
dyadav@ubuntu:~$
```

```
dyadav@ubuntu:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                                   NAMES
6319692e076d   nginx:latest   "/docker-entrypoint.…"   22 minutes ago   Up 22 minutes   0.0.0.0:8000->80/tcp, :::8000->80/tcp   nginx
4d55983a2d20   mysql:latest   "docker-entrypoint.s…"   34 minutes ago   Up 34 minutes   3306/tcp, 33060/tcp                     mysql
```

Host Port: Container Port

use the Private Ip Address:80 in ubuntu Linux Desktop



Projects :

JAVA Project

Create a Dockerfile to build a image

a) Clone github project on ubuntu machine.

git clone https://github.com/dheeruyadav54/simple-java-docker.git

vim Dockerfile —> Create a Dockerfile with Capital D.

 # Write the Dockerfile per below format.

 # Base Image of Java

FROM openjdk:17-jdk-slim

#Create a Working Directory

WORKDIR /java

 # Copy the Code

COPY /src/Main.java /java/Main.java

#Compile the Code

RUN javac Main.java

#Run the Code
CMD ["java","Main"]

```
# Base Image of Java
FROM openjdk:17-jdk-slim

#Create a Working Directory
WORKDIR /java

# Copy the Code
COPY /src/Main.java /java/Main.java

#Compile the Code
RUN javac Main.java

#Run the Code
CMD ["java","Main"]
~
~
```

Here is the Code & directory for Main.Java

```
dyadav@ubuntu:~/simple-java-docker$ ls
Dockerfile  README.md  src
dyadav@ubuntu:~/simple-java-docker$
dyadav@ubuntu:~/simple-java-docker$
dyadav@ubuntu:~/simple-java-docker$ ls
Dockerfile  README.md  src
dyadav@ubuntu:~/simple-java-docker$ cd src/
dyadav@ubuntu:~/simple-java-docker/src$
dyadav@ubuntu:~/simple-java-docker/src$ ls
Main.java
```

b) Build the image from above Dockerfile

$ docker build -t java:latest .

-t : for Tag name

. : Dot means Current Directory Dockerfile

```
dyadav@ubuntu:~/simple-java-docker$ docker build -t java:latest .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon     64kB
Step 1/5 : FROM openjdk:17-jdk-slim
17-jdk-slim: Pulling from library/openjdk
6d4a449ac69c: Pull complete
a59f13dc084e: Pull complete
1d5035d2d5c6: Pull complete
Digest: sha256:aaa3b3cb27e3e520b8f116863d0580c438ed55ecfa0bc126b41f68c3f62f9774
Status: Downloaded newer image for openjdk:17-jdk-slim
 ---> 8a3a2ffec52a
Step 2/5 : WORKDIR /java
 ---> Running in 5ea96c590072
Removing intermediate container 5ea96c590072
 ---> 118d50bc017b
Step 3/5 : COPY /src/Main.java /java/Main.java
 ---> 8cf7853eaa6a
Step 4/5 : RUN javac Main.java
 ---> Running in 130993577cd3
Removing intermediate container 130993577cd3
 ---> 61b08d35e093
Step 5/5 : CMD ["java","Main"]
 ---> Running in 9238b21787f0
Removing intermediate container 9238b21787f0
 ---> d8bd05d0c0ac
Successfully built d8bd05d0c0ac
Successfully tagged java:latest
dyadav@ubuntu:~/simple-java-docker$
```

===> Java Docker Images has been created,

```
dyadav@ubuntu:~/simple-java-docker$ docker images
REPOSITORY   TAG           IMAGE ID       CREATED          SIZE
java         latest        d8bd05d0c0ac   3 minutes ago    402MB
mysql        latest        c3239d2b6d88   2 weeks ago      608MB
nginx        latest        443d199e8bfc   4 weeks ago      193MB
openjdk      17-jdk-slim   8a3a2ffec52a   2 years ago      402MB
dyadav@ubuntu:~/simple-java-docker$
```

c) now run the Container per build image.

===> JAVA Code running successfully

```
dyadav@ubuntu:~/simple-java-docker$ docker run java:latest
Hello, Docker! Current date: Mon Jul 22 06:47:56 UTC 2024
dyadav@ubuntu:~/simple-java-docker$
```

d) Let's modify the JAVA Code & rebuild the image

```
import java.util.Date;

public class Main {
    public static void main(String[] args) {
        Date currentDate = new Date();
        System.out.println("Hello, Batch-7 Dosto! Current date: " + currentDate);
    }
}
~
```

===> Again Rebuild the Image with tag Name Java:V1

```
dyadav@ubuntu:~/simple-java-docker$ docker build -t java:V1 .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon     64kB
Step 1/5 : FROM openjdk:17-jdk-slim
 ---> 8a3a2ffec52a
Step 2/5 : WORKDIR /java
 ---> Using cache
 ---> 118d50bc017b
Step 3/5 : COPY /src/Main.java /java/Main.java
 ---> 338be42a9ee5
Step 4/5 : RUN javac Main.java
 ---> Running in 89818b9ffc5c
Removing intermediate container 89818b9ffc5c
 ---> 1cf9a595723a
Step 5/5 : CMD ["java","Main"]
 ---> Running in b9669a0ab5a1
Removing intermediate container b9669a0ab5a1
 ---> 87df04c98e32
Successfully built 87df04c98e32
Successfully tagged java:V1
```

====> Run the Java Image with New Tag Name

```
dyadav@ubuntu:~/simple-java-docker$ docker run java:V1
Hello, Batch-7 Dosto! Current date: Mon Jul 22 06:54:34 UTC 2024
dyadav@ubuntu:~/simple-java-docker$
```

====> With OLD Tag Name

```
dyadav@ubuntu:~/simple-java-docker$ docker run java:latest
Hello, Docker! Current date: Mon Jul 22 06:55:19 UTC 2024
dyadav@ubuntu:~/simple-java-docker$
```

NodeJs Project

a) Clone github project on ubuntu machine.

git clone https://github.com/dheeruyadav54/node-todo-cicd.git

vim Dockerfile —> Create a Dockerfile with Capital D.

```
# Node Base Image
FROM node:12.2.0-alpine

#Working Directry
WORKDIR /node

#Copy the Code
COPY . .

#Install the dependecies
RUN npm install
RUN npm run test
EXPOSE 8000

#Run the code
CMD ["node","app.js"]
~
~
~
```

b) Create a .dockerignore file

$ vim .dockerignore : Add all the files & folder which you don't want to copy from Local to Container.

$ docker build -t nodejs:latest . : Build a nodejs image from Dockerfile

```
dyadav@ubuntu:~/DevOps/node-todo-cicd$ docker build -t nodejs:latest .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  479.2kB
Step 1/7 : FROM node:12.2.0-alpine
12.2.0-alpine: Pulling from library/node
0362ad1dd800: Pull complete
a190fd4bc030: Pull complete
b226c22d5327: Pull complete
Digest: sha256:2ab3d9a1bac67c9b4202[illegible]774664adaa94d2f1e426d8d28e07bf8979df61c8694
Status: Downloaded newer image for node:12.2.0-alpine
 ---> 82b4eda28a0b
Step 2/7 : WORKDIR /node
 ---> Running in d0371a2d4dd9
Removing intermediate container d0371a2d4dd9
 ---> 6b8999dd3ca6
Step 3/7 : COPY . .
 ---> 0e19f94e9cc6
Step 4/7 : RUN npm install
 ---> Running in f75fa01dbe8d
```

```
Step 6/7 : EXPOSE 8000
 ---> Running in 30f1440b15a3
Removing intermediate container 30f1440b15a3
 ---> 87d2e4b98af5
Step 7/7 : CMD ["node","app.js"]
 ---> Running in 0d045e828d5e
Removing intermediate container 0d045e828d5e
 ---> 0995abac7974
Successfully built 0995abac7974
Successfully tagged nodejs:latest
dyadav@ubuntu:~/DevOps/node-todo-cicd$
```

===> Nodejs images have been built.

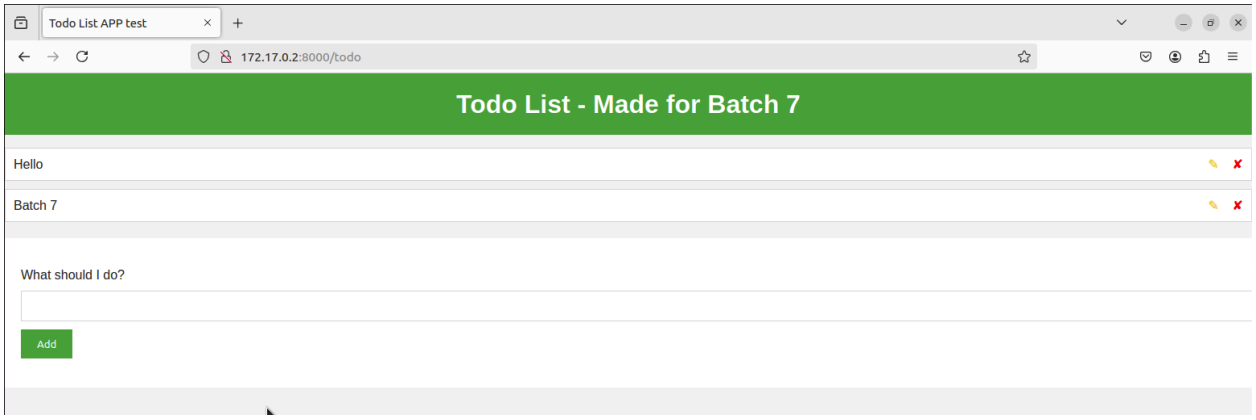
```
dyadav@ubuntu:~/DevOps/node-todo-cicd$ docker images
REPOSITORY   TAG              IMAGE ID       CREATED          SIZE
nodejs       latest           0995abac7974   56 seconds ago   104MB
mysql        latest           c3239d2b6d88   2 weeks ago      608MB
nginx        latest           443d199e8bfc   4 weeks ago      193MB
openjdk      17-jdk-slim      8a3a2ffec52a   2 years ago      402MB
node         12.2.0-alpine    82b4eda28a0b   5 years ago      77.6MB
dyadav@ubuntu:~/DevOps/node-todo-cicd$
dyadav@ubuntu:~/DevOps/node-todo-cicd$
dyadav@ubuntu:~/DevOps/node-todo-cicd$
```

c) Let's run the container.

$ docker run -d -p 8000:8000 nodejs:latest

```
dyadav@ubuntu:~/DevOps/node-todo-cicd$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS                                       NAMES
9fb274fff766   nodejs:latest  "node app.js"            5 seconds ago   Up 4 seconds   0.0.0.0:8100->8000/tcp, :::8100->8000/tcp   dazzl
ing_curie
3a4486b4759a   mysql:latest   "docker-entrypoint.s…"   15 hours ago    Up 15 hours    3306/tcp, 33060/tcp                         mysql
6319692e076d   nginx:latest   "/docker-entrypoint.…"   17 hours ago    Up 17 hours    0.0.0.0:8000->80/tcp, :::8000->80/tcp       nginx
dyadav@ubuntu:~/DevOps/node-todo-cicd$
```

nodejs container is running & serving the web page.



Python-Flask Project

a) Clone github project on ubuntu machine.

git clone https://github.com/dheeruyadav54/flask-app.git

vim Dockerfile —> Create a Dockerfile with Capital D

Created a Docker file per requirement

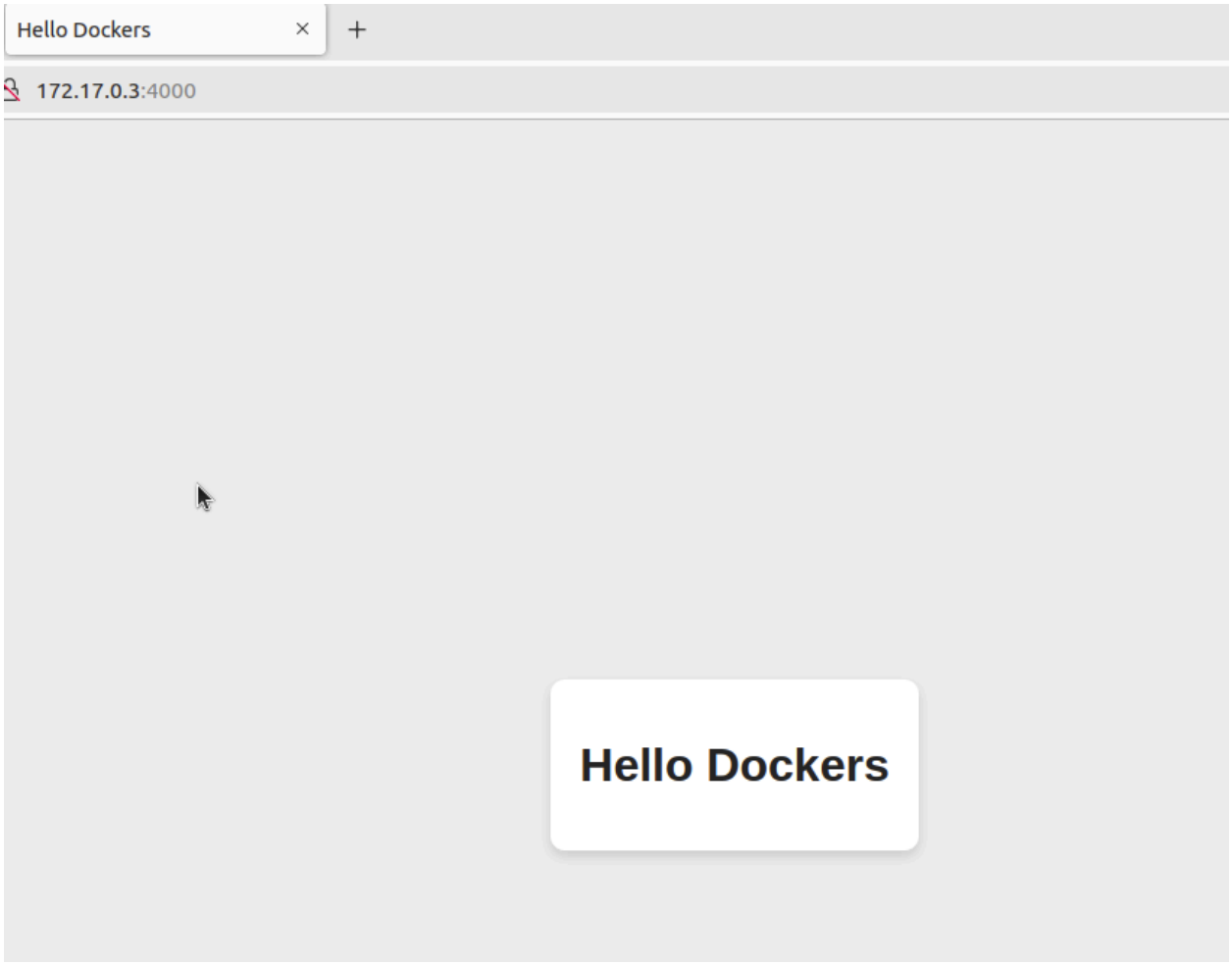
```
#Base image
FROM python:3.11

#Working directory

WORKDIR /python

#Copy the CODE
COPY . /python

# install the dependencies

RUN pip install --upgrade pip
RUN pip install -r requirements.txt

# Run the Code

CMD ["python","app.py"]
~
~
~
```

$ docker build -t flask-app:latest . :--> run this command to build the flash app image.

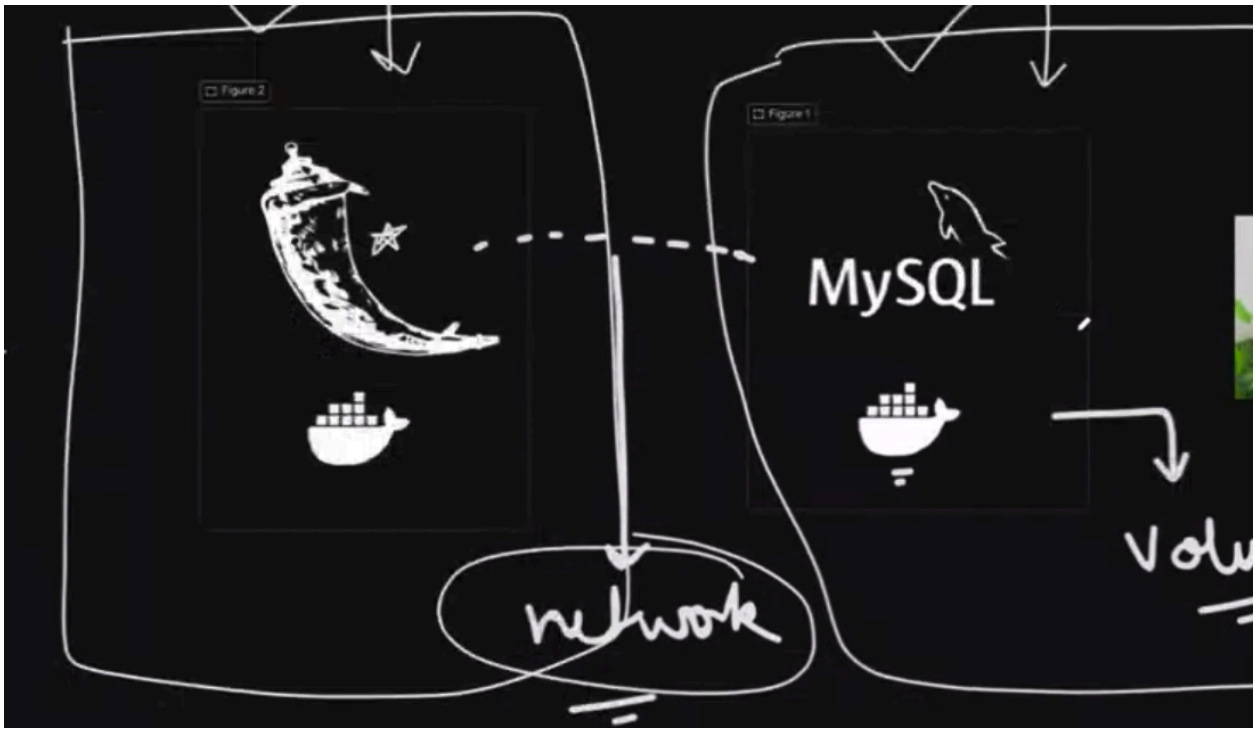
```
dyadav@ubuntu:~/DevOps/flask-app$ docker build -t flask-app:latest .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  70.14kB
Step 1/5 : FROM python:3.11
3.11: Pulling from library/python
0bd1f8180c50: Pull complete
b16610496e73: Pull complete
64a6ab51b82d: Pull complete
222f8241ab9f: Pull complete
7a2d0a549cea: Pull complete
fe8009395e7d: Pull complete
db1b9e3a7d66: Pull complete
78736cfe2b5d: Pull complete
Digest: sha256:c5254471e6073d8942091227f469302f85b30d4b23077226f135360491f5226a
Status: Downloaded newer image for python:3.11
 ---> b721d6003c96
Step 2/5 : WORKDIR /python
 ---> Running in 4b74706100a1
Removing intermediate container 4b74706100a1
 ---> 2b45d857ae42
Step 3/5 : COPY . /python
 ---> 88e5a2bb006b
Step 4/5 : RUN pip install -r requirements.txt
 ---> Running in 70feefdb6c3e
Collecting Flask==2.1.3 (from -r requirements.txt (line 1))
  Downloading Flask-2.1.3-py3-none-any.whl.metadata (3.9 kB)
Collecting Werkzeug==2.2.3 (from -r requirements.txt (line 2))
  Downloading Werkzeug-2.2.3-py3-none-any.whl.metadata (4.4 kB)
Collecting Jinja2>=3.0 (from Flask==2.1.3->-r requirements.txt (line 1))
  Downloading jinja2-3.1.4-py3-none-any.whl.metadata (2.6 kB)
Collecting itsdangerous>=2.0 (from Flask==2.1.3->-r requirements.txt (line 1))
  Downloading itsdangerous-2.2.0-py3-none-any.whl.metadata (1.9 kB)
Collecting click>=8.0 (from Flask==2.1.3->-r requirements.txt (line 1))
  Downloading click-8.1.7-py3-none-any.whl.metadata (3.0 kB)
Collecting MarkupSafe>=2.1.1 (from Werkzeug==2.2.3->-r requirements.txt (line 2))
  Downloading MarkupSafe-2.1.5-cp311-cp311-manylinux_2_17_aarch64.manylinux2014_aarch64.whl.metadata (3.0 kB)
Downloading Flask-2.1.3-py3-none-any.whl (95 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 95.6/95.6 kB 1.8 MB/s eta 0:00:00
Downloading Werkzeug-2.2.3-py3-none-any.whl (233 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 233.6/233.6 kB 2.7 MB/s eta 0:00:00
Downloading click-8.1.7-py3-none-any.whl (97 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 97.9/97.9 kB 3.8 MB/s eta 0:00:00
Downloading itsdangerous-2.2.0-py3-none-any.whl (16 kB)
Downloading jinja2-3.1.4-py3-none-any.whl (133 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 133.3/133.3 kB 2.7 MB/s eta 0:00:00
Downloading MarkupSafe-2.1.5-cp311-cp311-manylinux_2_17_aarch64.manylinux2014_aarch64.whl (29 kB)
Installing collected packages: MarkupSafe, itsdangerous, click, Werkzeug, Jinja2, Flask
Successfully installed Flask-2.1.3 Jinja2-3.1.4 MarkupSafe-2.1.5 Werkzeug-2.2.3 click-8.1.7 itsdangerous-2.2.0
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package mana
ger. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv

[notice] A new release of pip is available: 24.0 -> 24.1.2
[notice] To update, run: pip install --upgrade pip
Removing intermediate container 70feefdb6c3e
 ---> 17aad80ffc5c
Step 5/5 : CMD ["python","app.py"]
 ---> Running in 28aa3be9b322
Removing intermediate container 28aa3be9b322
 ---> e81d118ce607
Successfully built e81d118ce607
Successfully tagged flask-app:latest
```

$ docker run -d -p 4000:4000 flask-app:latest : use this command to run the container.

```
dyadav@ubuntu:~/DevOps/flask-app$ docker run -d -p 4000:4000 flask-app:latest
7442fdc057127487873c678fbf54db20848b963e7e73bd549cd38be964ca5041
dyadav@ubuntu:~/DevOps/flask-app$ docker ps -a
CONTAINER ID   IMAGE              COMMAND           CREATED             STATUS             PORTS
      NAMES
7442fdc05712   flask-app:latest   "python app.py"   6 seconds ago       Up 5 seconds       0.0.0.0:4000->4000/tcp, :::4000->400
0/tcp   great_kilby
1a2bf949dde8   nodejs:V1          "node app.js"     About an hour ago   Up About an hour   0.0.0.0:8000->8000/tcp, :::8000->800
0/tcp   nodejs
dyadav@ubuntu:~/DevOps/flask-app$
dyadav@ubuntu:~/DevOps/flask-app$
```



two-tier Project



b) Clone github project on ubuntu machine.

git clone https://github.com/dheeruyadav54/two-tier-flask-app.git

We are going to deploy a 2 tier application on docker, by default Container runs in isolation mode hence both the container's can't communicate with each other.

Solution : Create a Network & assign both the containers into that Network so they both can communicate.

$ docker network ls : to list all the present Network

```
dyadav@ubuntu:~/DevOps/flask-app$ docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
ef45ec36d531   bridge    bridge    local
05bde5277c59   host      host      local
377e486608a6   none      null      local
```

$ docker network create twotier-Network : Created a new Network for this Project & by Default Network type would be **bridge**

Network Type:

==========

bridge : In this Network type we need to allow the traffic explicitly through variables like -p 81:80 , binding the host port 81 with container port 80.

Host : In this Network type we don't need to allow the traffic , allow everything.

None : Everything is blocked.

```
dyadav@ubuntu:~/DevOps/flask-app$ docker network create twotier-Network
9576d92948208634e69e9b0b0d3c2a9a108818b385b1f9750e7ae6649fd5bbac
dyadav@ubuntu:~/DevOps/flask-app$
dyadav@ubuntu:~/DevOps/flask-app$
dyadav@ubuntu:~/DevOps/flask-app$ docker network ls
NETWORK ID     NAME              DRIVER    SCOPE
ef45ec36d531   bridge            bridge    local
05bde5277c59   host              host      local
377e486608a6   none              null      local
9576d9294820   twotier-Network   bridge    local
dyadav@ubuntu:~/DevOps/flask-app$
```

$ docker volume create twotier-Volume : Created a new Volume for this Project.

```
dyadav@ubuntu:~/DevOps/flask-app$ docker volume create twotier-Volume
twotier-Volume
dyadav@ubuntu:~/DevOps/flask-app$
dyadav@ubuntu:~/DevOps/flask-app$
dyadav@ubuntu:~/DevOps/flask-app$ docker volume ls
DRIVER    VOLUME NAME
local     twotier-Volume
dyadav@ubuntu:~/DevOps/flask-app$
```

$ docker run -d -v twotier-Volume:/var/lib/mysql --name mysql --network twotier-Network -e MYSQL_DATABASE=DevOps -e MYSQL_ROOT_PASSWORD=root -e MYSQL_USER=admin -e MYSQL_PASSWORD=admin mysql:5.7 : Run this command we have given couple of variables here -d, -v, –name , –network , -e etc

mysql database is ready

```
ubuntu@ip-172-31-24-254:~$
ubuntu@ip-172-31-24-254:~$ docker network create twotier-Network
8ad0a472dc5c7528b1a003ef2b08047e3f46693a04d55af355919cadd38542be
ubuntu@ip-172-31-24-254:~$ docker network ls
NETWORK ID     NAME              DRIVER    SCOPE
0d108b37e58b   bridge            bridge    local
1d192d73ea7c   host              host      local
3ee9fae73cfb   none              null      local
8ad0a472dc5c   twotier-Network   bridge    local
ubuntu@ip-172-31-24-254:~$
ubuntu@ip-172-31-24-254:~$
ubuntu@ip-172-31-24-254:~$
ubuntu@ip-172-31-24-254:~$ docker run -d -v twotier-Volume:/var/lib/mysql --name mysql --network twotier-Network -e  MYSQL_DATABASE
=DevOps -e MYSQL_ROOT_PASSWORD=root -e MYSQL_USER=admin -e MYSQL_PASSWORD=admin mysql:5.7
c838a078737d9bdaff76b909539fb3f300789d81f242cfe92263dcadacea6be5
ubuntu@ip-172-31-24-254:~$
ubuntu@ip-172-31-24-254:~$
ubuntu@ip-172-31-24-254:~$
ubuntu@ip-172-31-24-254:~$ docker ps
CONTAINER ID   IMAGE       COMMAND                  CREATED         STATUS         PORTS                  NAMES
c838a078737d   mysql:5.7   "docker-entrypoint.s…"   8 seconds ago   Up 6 seconds   3306/tcp, 33060/tcp   mysql
ubuntu@ip-172-31-24-254:~$ docker images
REPOSITORY                   TAG          IMAGE ID       CREATED        SIZE
two-tier-flask-app-backend   latest       cfab2990ffd7   2 days ago     391MB
java-batch7                  latest       40cbbef8dc91   3 days ago     408MB
python                       3.9-slim     b97320a8c1ca   2 weeks ago    125MB
mysql                        latest       5cde95de907d   3 weeks ago    586MB
nginx                        latest       fffffc90d343   4 weeks ago    188MB
mysql                        5.7          5107333e08a8   7 months ago   501MB
openjdk                      17-jdk-slim  37cb44321d04   2 years ago    408MB
ubuntu@ip-172-31-24-254:~$
```

===> Run the below commands to check the Database.

$ docker exec -it <<Container ID>> bash

bash-4.2# mysql -u root -p

Enter password:

mysql> show databases;

mysql> use DevOps; : select the DevOps Database

mysql> show tables; : Check the Tables

mysql> select * from messages; : to check the Data

```
mysql>
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| DevOps             |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.01 sec)

mysql>
mysql>
mysql> use DevOps;
Database changed
mysql>
mysql>
mysql> show tables;
Empty set (0.01 sec)

mysql>
```

```
mysql> show tables;
+------------------+
| Tables_in_DevOps |
+------------------+
| messages         |
+------------------+
1 row in set (0.00 sec)

mysql> select * from messages;
+----+-----------------+
| id | message         |
+----+-----------------+
|  1 | Amazing-Batch-7 |
|  2 | hello dosto     |
|  3 | Good JOB        |
|  4 | Dheeraj Yadav   |
+----+-----------------+
4 rows in set (0.00 sec)
```

Dockerfile to build the Python flask-App

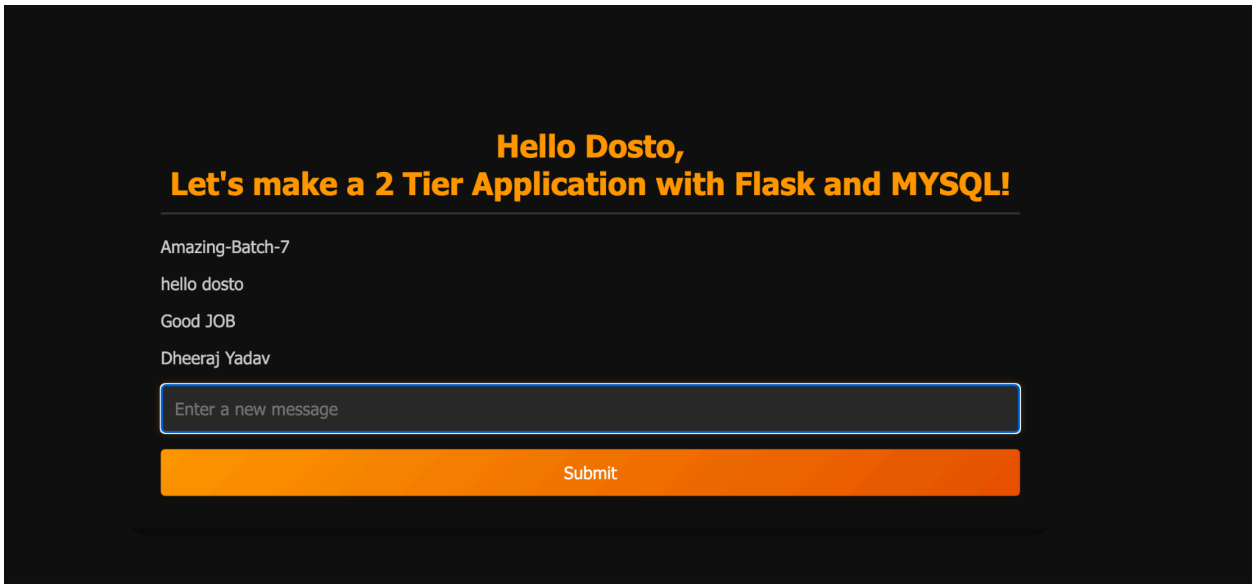
```
# Base Image
FROM python:3.11

# Working Directory
WORKDIR /python

# install required packages for system
RUN apt-get update \
    && apt-get upgrade -y \
    && apt-get install -y gcc default-libmysqlclient-dev pkg-config \
    && rm -rf /var/lib/apt/lists/*

# Copy the requirements file
COPY requirements.txt .

# Copy the Code
COPY . .

# Install app Dependencies
RUN pip install mysqlclient
RUN pip install --no-cache-dir -r requirements.txt

# Copy the reset of application code
COPY . .

# Run the Code
CMD ["python","app.py"]

~
~
~
~
```

$ docker run -d -p 5000:5000 --network twotier-Network --name twotier-Flask -e MYSQL_HOST=mysql -e MYSQL_USER=admin -e MYSQL_PASSWORD=admin -e MYSQL_DB=DevOps twotier-flask:latest : Used this command to run the flash app with mysql Database

```
ubuntu@ip-172-31-24-254:~/DevOps/two-tier-flask-app$ docker run -d -p 5000:5000 --network twotier-Network --name twotier-Flask -e M
YSQL_HOST=mysql -e MYSQL_USER=admin -e MYSQL_PASSWORD=admin -e MYSQL_DB=DevOps twotier-flask:latest
86a77029c60bc761d4a91c0cc3ff08001508abb1ef6ada60b20d60354858da8b
ubuntu@ip-172-31-24-254:~/DevOps/two-tier-flask-app$ docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED          STATUS          PORTS
          NAMES
86a77029c60b   twotier-flask:latest   "python app.py"          3 seconds ago    Up 2 seconds    0.0.0.0:5000->5000/tcp, :::5000->50
00/tcp   twotier-Flask
c838a078737d   mysql:5.7              "docker-entrypoint.s…"   15 minutes ago   Up 15 minutes   3306/tcp, 33060/tcp
          mysql
ubuntu@ip-172-31-24-254:~/DevOps/two-tier-flask-app$
```

Flash app is running on port 5000

http://3.79.149.35:5000



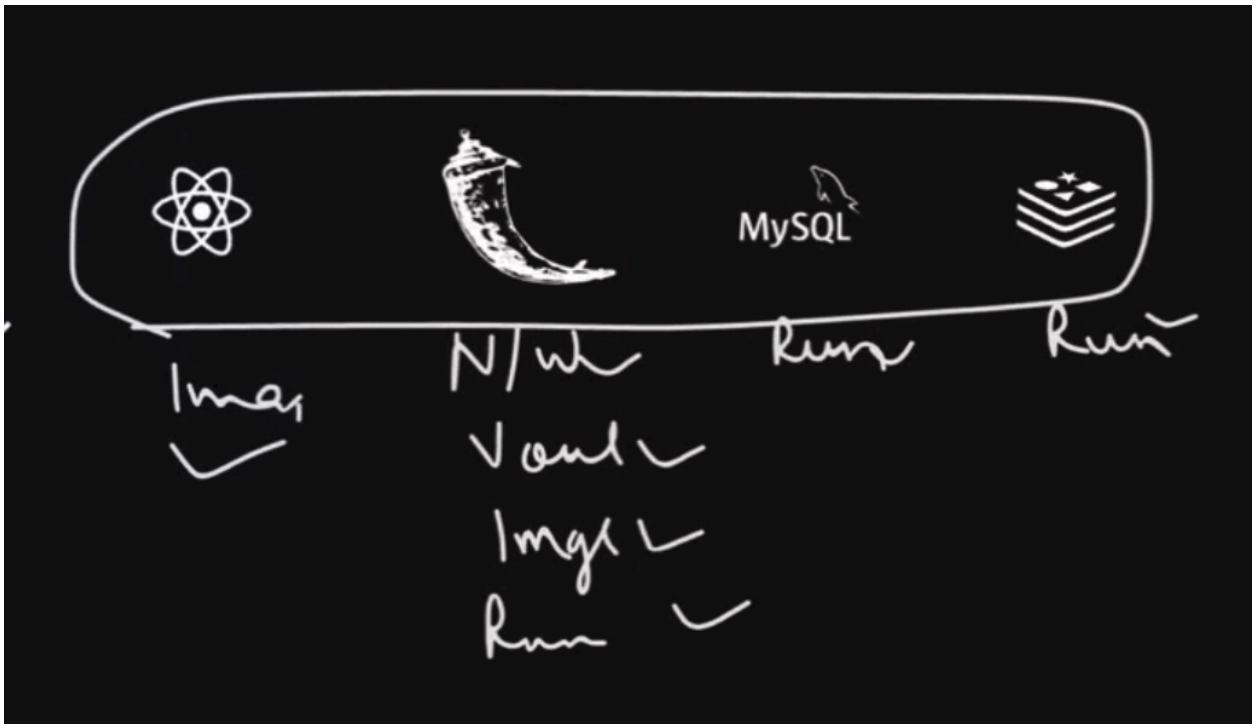

```
ubuntu@ip-172-31-24-254:~/DevOps/two-tier-flask-app$ docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED          STATUS          PORTS                                       NAMES
86a77029c60b   twotier-flask:latest   "python app.py"          5 minutes ago    Up 5 minutes    0.0.0.0:5000->5000/tcp, :::5000->5000/tcp   twotier-Flask
c838a078737d   mysql:5.7            "docker-entrypoint.s…"   21 minutes ago   Up 21 minutes   3306/tcp, 33060/tcp                         mysql
ubuntu@ip-172-31-24-254:~/DevOps/two-tier-flask-app$
```

Docker Compose

It is used to run a single YAML file to build the container.

First install the docker compose:

$ sudo apt-get install docker-compose-v2 : To install the docker-compose.



====> Create a docker-compose.yml file.

$ docker compose up -d : To Create the multiple docker Container with a single file.

$ docker compose down : to remove the created docker container

Services : Under Service we define our container.

build:

Context . → Dot Represent that Dockerfile is here in the current directory.

```yaml
version: '3'
services:

  backend:
    build:
      context: .
    ports:
      - "5000:5000"
    environment:
      MYSQL_HOST: mysql
      MYSQL_USER: admin
      MYSQL_PASSWORD: admin
      MYSQL_DB: myDb
    depends_on:
      - mysql
    restart: always

  mysql:
    image: mysql:5.7
    ports:
      - "3306:3306"
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: myDb
      MYSQL_USER: admin
      MYSQL_PASSWORD: admin
    volumes:
      - ./message.sql:/docker-entrypoint-initdb.d/message.sql   # Mount sql s
      - mysql-data:/var/lib/mysql  # Mount the volume for MySQL data storage

volumes:
  mysql-data:
```

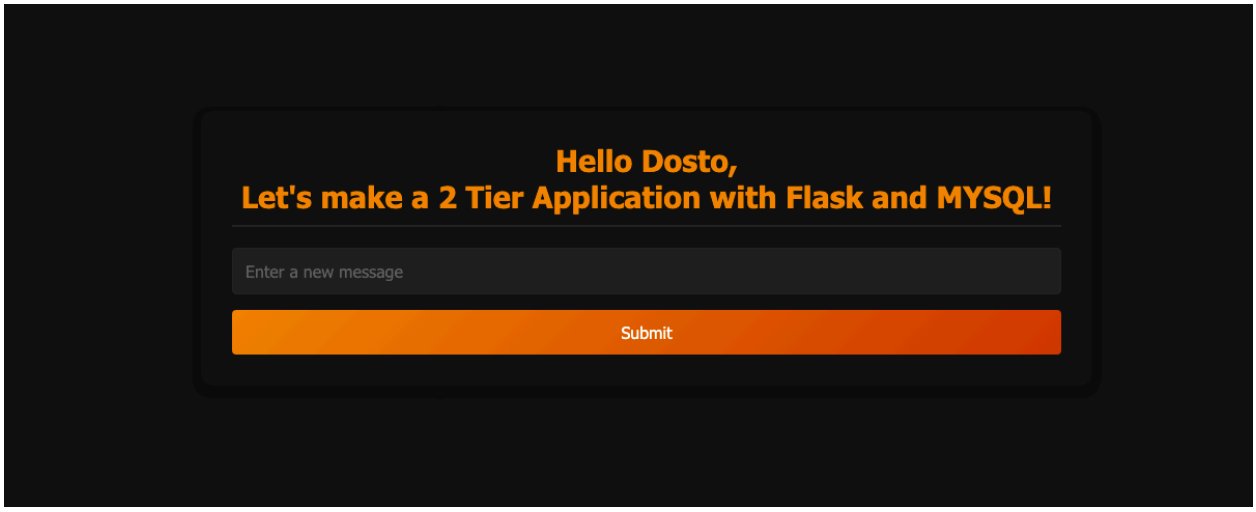
$ docker compose up -d : To Run the Containers.

```
ubuntu@ip-172-31-24-254:~/DevOps/two-tier-flask-app$ docker compose up -d
[+] Running 2/4
 ⠿ Network two-tier-flask-app_default      Created
 ⠿ Volume "two-tier-flask-app_mysql-data"  Created
 ✔ Container two-tier-flask-app-mysql-1    Started
 ✔ Container two-tier-flask-app-backend-1  Started
ubuntu@ip-172-31-24-254:~/DevOps/two-tier-flask-app$
ubuntu@ip-172-31-24-254:~/DevOps/two-tier-flask-app$
ubuntu@ip-172-31-24-254:~/DevOps/two-tier-flask-app$ docker ps
```

$ docker ps : To verify the running containers.

```
ubuntu@ip-172-31-24-254:~/DevOps/two-tier-flask-app$ docker ps
CONTAINER ID   IMAGE                      COMMAND                  CREATED              STATUS              PORTS                                                  NAMES
8a146ddb667a   two-tier-flask-app-backend   "python app.py"          About a minute ago   Up About a minute   0.0.0.0:5000->5000/tcp, :::5000->5000/tcp              two-tier
-flask-app-backend-1
0d1884918012   mysql:5.7                  "docker-entrypoint.s…"   About a minute ago   Up About a minute   0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp   two-tier
-flask-app-mysql-1
```

Tier-2 Application is running on port 5000.



Multi-Stage docker build

“ Will ADD SOON”

Docker scout

Docker scout is to check the vulnerability from the images.

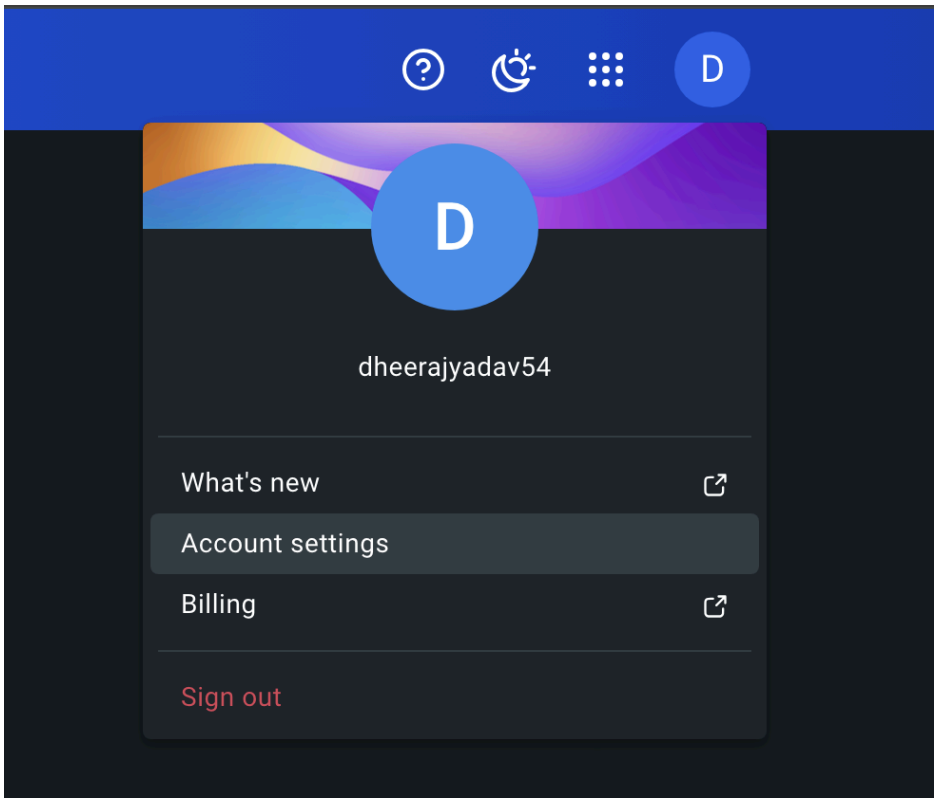
Step1: Create a Directory.
$ mkdir -p $HOME/.docker/scout

Step2: To install the latest version of the plugin
$ curl -fsSL https://raw.githubusercontent.com/docker/scout-cli/main/install.sh -o install-scout.sh
sh install-scout.sh

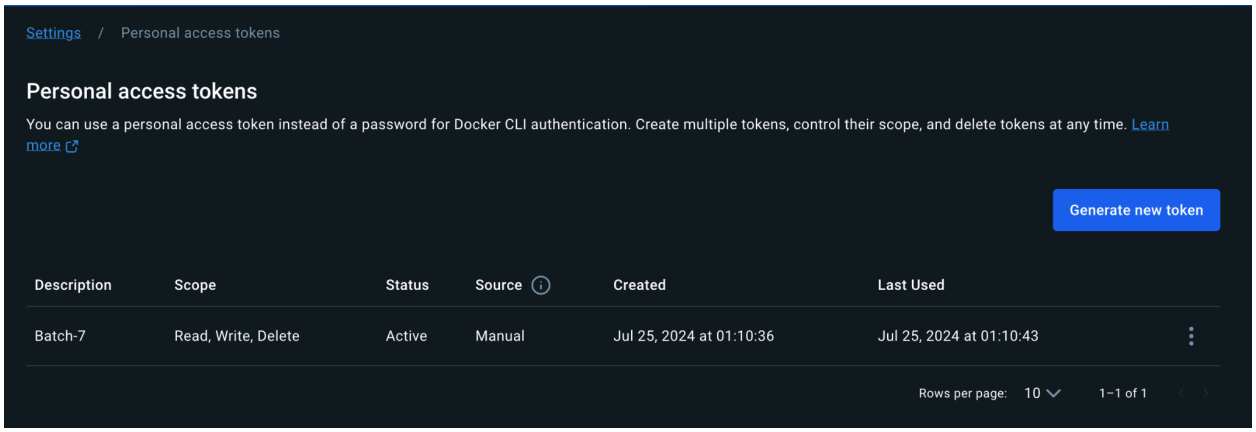
Step3:
$ docker login
Username : dheerajyadav54
Password: Personalized Access Token

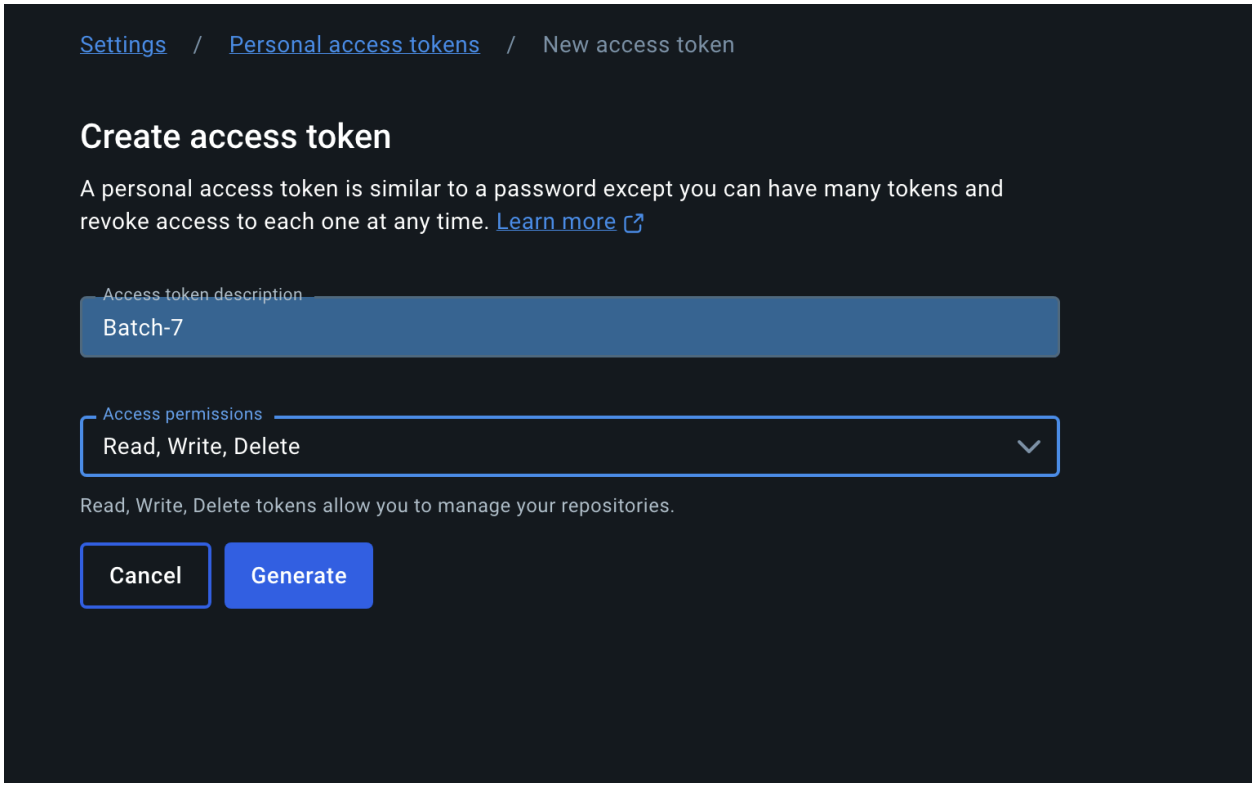
Which can be collected per below method:

a) **Docker Hub Account —-> Account Setting**



b) Personal access tokens → Generate new token





Once it's generated, copy the token & paste into the Password Section.

```
dyadav@ubuntu:~/DevOps$ docker login
Log in with your Docker ID or email address to push and pull images from Docker
Hub. If you don't have a Docker ID, head over to https://hub.docker.com/ to crea
te one.
You can log in with your password or a Personal Access Token (PAT). Using a limi
ted-scope PAT grants better security and is required for organizations using SSO
. Learn more at https://docs.docker.com/go/access-tokens/

Username: dheerajyadav54
Password:
WARNING! Your password will be stored unencrypted in /home/dyadav/.docker/config
.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
dyadav@ubuntu:~/DevOps$
```

Now run the docker scout command to scan the images.

$ docker scout cves mysql:latest : to scan the mysql:latest image

```
dyadav@ubuntu:~/DevOps$ docker scout cves mysql:latest
    ✓ SBOM obtained from attestation, 169 packages found
    ✓ Provenance obtained from attestation
    ✗ Detected 5 vulnerable packages with a total of 55 vulnerabilities


## Overview

                    |                   Analyzed Image
--------------------+------------------------------------------------------
  Target            |  mysql:latest
    digest          |  d0a00c4c73db
    platform        |  linux/arm64
    provenance      |  https://github.com/docker-library/mysql.git
                    |  a482468640c602ccd8a7c86a4c7422f18a307326
    vulnerabilities |    3C     31H     12M     5L     6?
    size            |  168 MB
    packages        |  169
                    |
  Base image        |  oraclelinux:9-slim
                    |  226abc66ae59
```

$ docker scout quickview mysql:latest : you can quick view of the image

```
dyadav@ubuntu:~/DevOps$ docker scout quickview mysql:latest
    ✓ SBOM obtained from attestation, 169 packages found
    ✓ Provenance obtained from attestation

  Target       │  mysql:latest       │  3C  31H  12M  5L  6?
    digest     │  d0a00c4c73db       │
  Base image   │  oraclelinux:9-slim │  0C  0H  0M  0L

What's next:
    View vulnerabilities → docker scout cves mysql:latest
    Include policy results in your quickview by supplying an organization → dock
er scout quickview mysql:latest --org <organization>

dyadav@ubuntu:~/DevOps$
```

Docker Hub Login:

$ docker login

Username : dheerajyadav54

Password: Personalized Access Token

```
dyadav@ubuntu:~/DevOps$ docker login
Log in with your Docker ID or email address to push and pull images from Docker
Hub. If you don't have a Docker ID, head over to https://hub.docker.com/ to crea
te one.
You can log in with your password or a Personal Access Token (PAT). Using a limi
ted-scope PAT grants better security and is required for organizations using SSO
. Learn more at https://docs.docker.com/go/access-tokens/

Username: dheerajyadav54
Password:
WARNING! Your password will be stored unencrypted in /home/dyadav/.docker/config
.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
dyadav@ubuntu:~/DevOps$
```

$ docker image tag flask-app:latest dheerajyadav54/flask-app:latest : create a image tag with docker hub username i.e: dheerajyadav54

```
dyadav@ubuntu:~/DevOps$ docker image tag flask-app:latest dheerajyadav54/flask-a
pp:latest
dyadav@ubuntu:~/DevOps$ docker images
REPOSITORY                TAG             IMAGE ID       CREATED       SIZE
mysql                     latest          2010e93e56ca   2 days ago    608MB
dheerajyadav54/flask-app  latest          e81d118ce607   2 days ago    1.03GB
flask-app                 latest          e81d118ce607   2 days ago    1.03GB
nodejs                    V1              9206c2fe43f5   2 days ago    104MB
python                    3.11            b721d6003c96   2 weeks ago   1.01GB
node                      12.2.0-alpine   82b4eda28a0b   5 years ago   77.6MB
dyadav@ubuntu:~/DevOps$
```

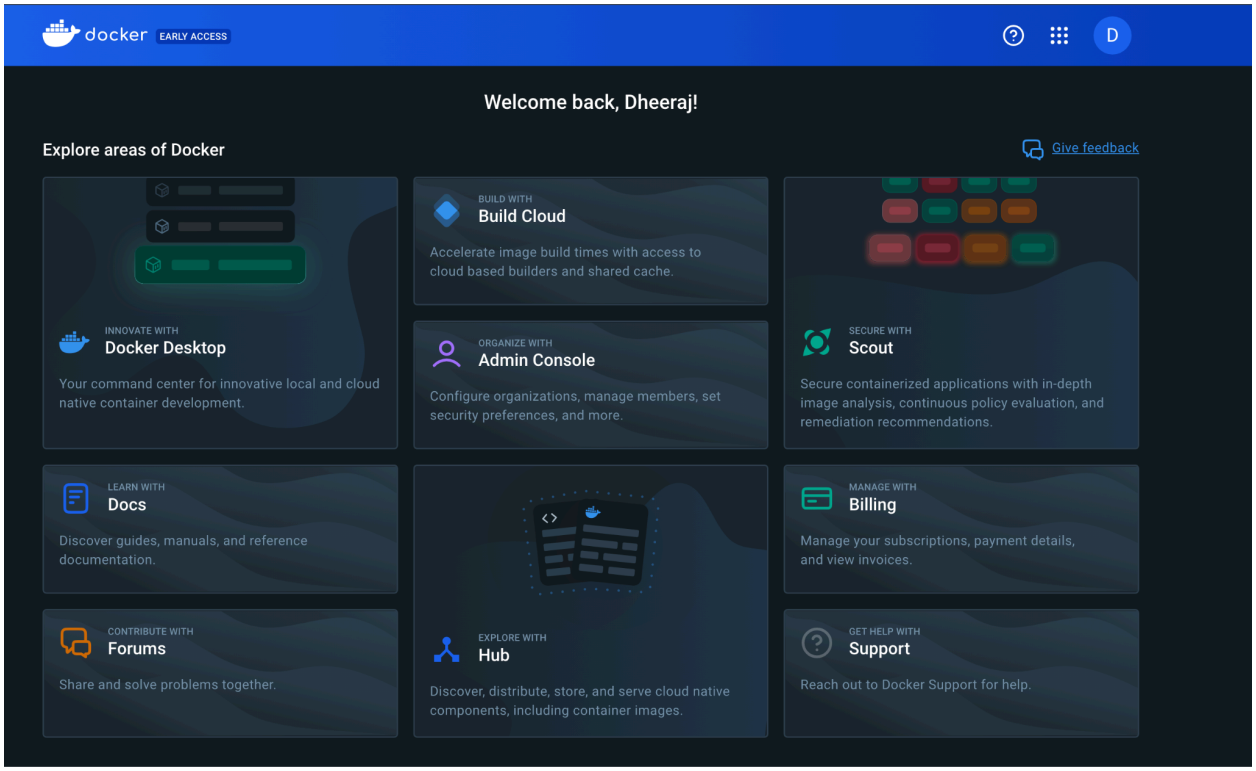
$ docker push dheerajyadav54/flask-app:latest : Use Docker Push Command to push the image from local to Docker Hub.

```
dyadav@ubuntu:~/DevOps$ docker push dheerajyadav54/flask-app:latest
The push refers to repository [docker.io/dheerajyadav54/flask-app]
d4adf981141c: Pushed
88e68907b16a: Pushed
cddbf5579256: Pushed
e8044085b055: Mounted from library/python
9b983e8dcfb2: Mounted from library/python
30c380a64572: Mounted from library/python
c33547265a2e: Mounted from library/python
b7db16820666: Mounted from library/python
daa3e00c13a2: Mounted from library/python
61dff208cd2f: Mounted from library/python
d1660adccd2b: Mounted from library/python
latest: digest: sha256:95057c718697982cfe6456a93ef16b47c6ca2df64b957cd7a9df58ebc
1d9c2b1 size: 2633
dyadav@ubuntu:~/DevOps$
```

To Check on docker hub :

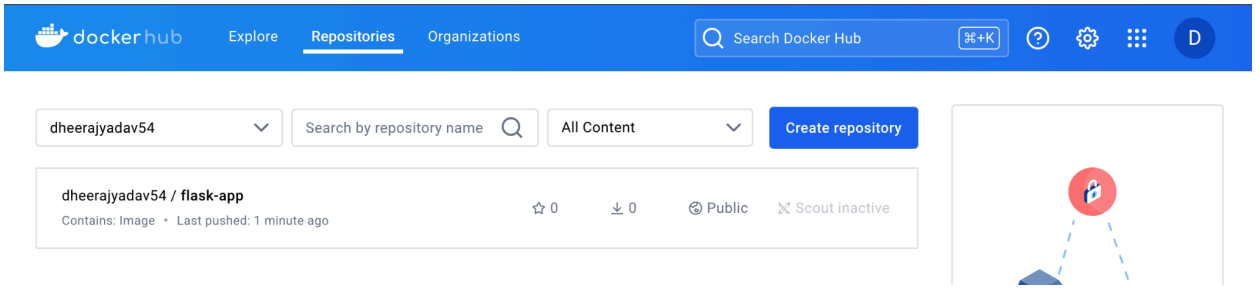
Go to URL : https://app.docker.com/

Click on Hub:



It will redirect to URL https://hub.docker.com/

Here we can see the pushed Docker image from Local to Docker HUB.



Bonus Commands