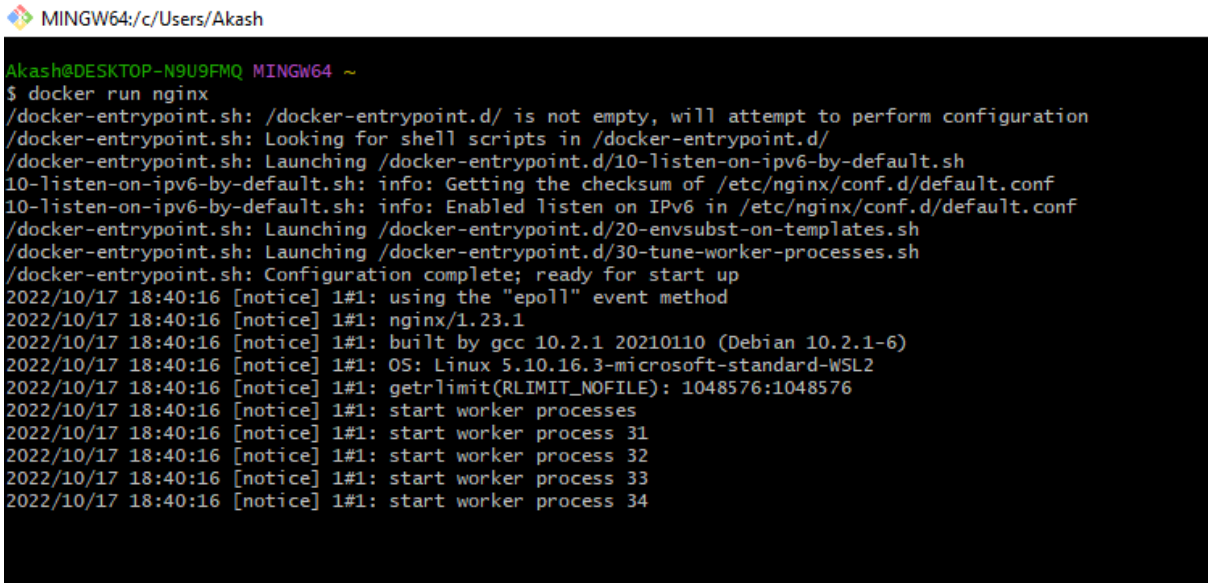
Docker & Docker Hub

Assignment 1:

Demonstrate minimum 15 basic docker command with explanation and screenshot.

1. Start the container

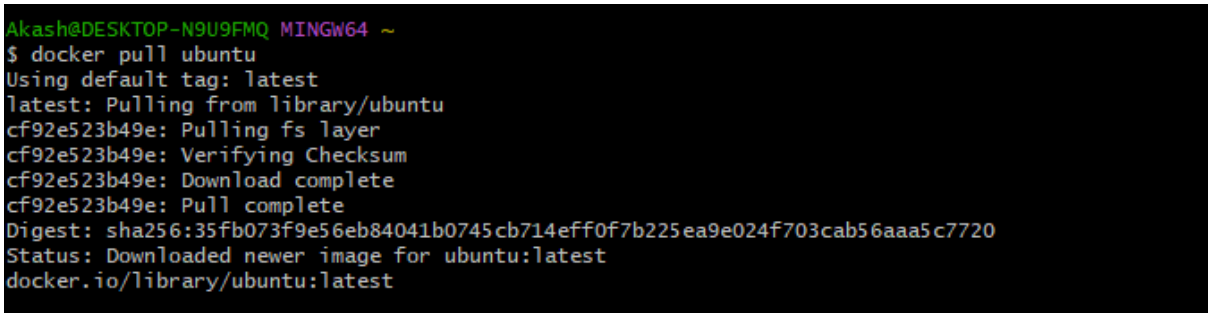
   - **docker run nginx**



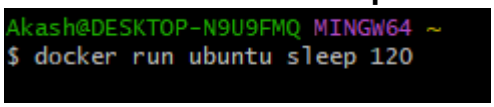
   If the image is not available locally it will try to pull image from docker hub repository and then running it

2. Only pull and do not start the container

   - **docker pull ubuntu**



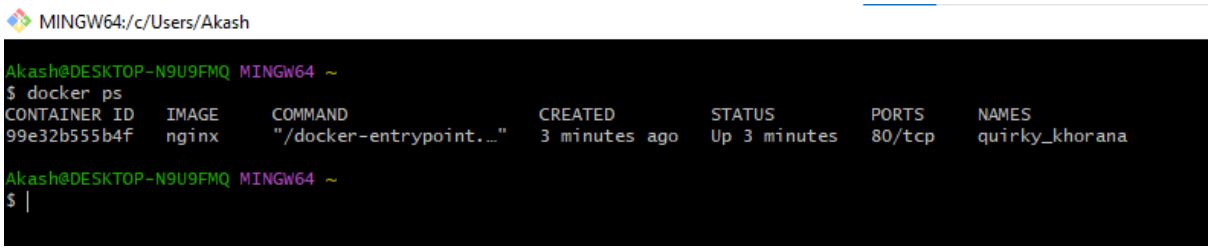
   - **docker run ubuntu sleep 120**



   Container will sleep after 120 seconds

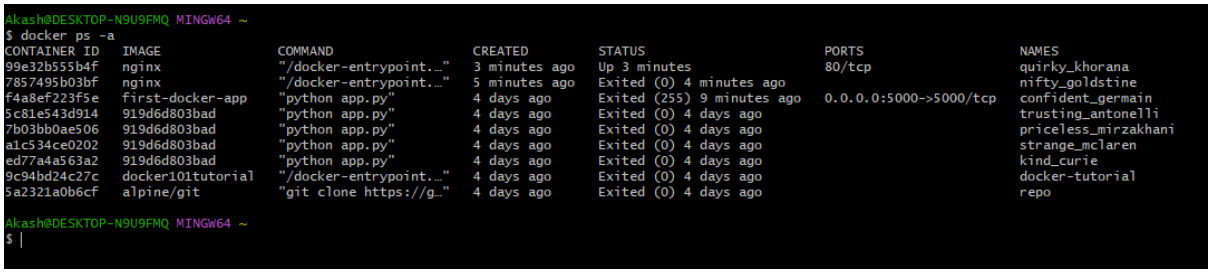
3. Listing the containers running

   - **docker ps**



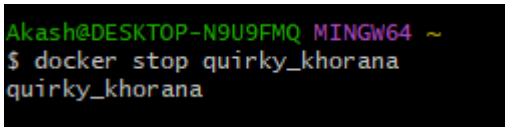
4. Listing the running as well as stopped containers

   - **docker ps -a**



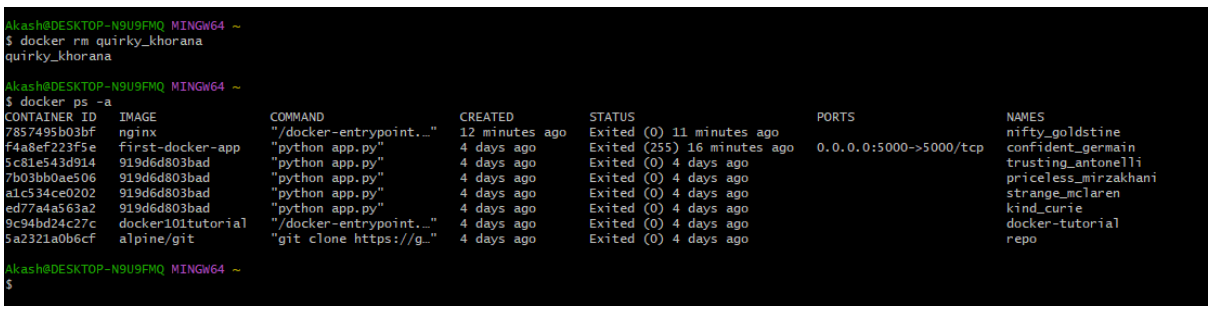
5. Stopping a running container

   - `docker stop quirky_khorana`



6. deleting the container.
   It is very essential that a container must be stopped before removing.

   - `docker rm quirky_khorana`



7. check docker images

   - `docker images`

```
Akash@DESKTOP-N9U9FMQ MINGW64 ~
$ docker images
REPOSITORY                    TAG       IMAGE ID       CREATED        SIZE
aakashsoni/first-docker-app   latest    6f07b6dcd25a   4 days ago     1.32GB
first-docker-app              latest    6f07b6dcd25a   4 days ago     1.32GB
docker101tutorial             latest    7e90f0a63bcb   4 days ago     28.9MB
alpine/git                    latest    b80d2cac43e4   10 days ago    43.6MB
nginx                         latest    51086ed63d8c   12 days ago    142MB

Akash@DESKTOP-N9U9FMQ MINGW64 ~
$
```

8. Deleting docker image

   - `docker rmi nginx`

```
Akash@DESKTOP-N9U9FMQ MINGW64 ~
$ docker rm nifty_goldstine
nifty_goldstine

Akash@DESKTOP-N9U9FMQ MINGW64 ~
$ docker rmi nginx
Untagged: nginx:latest
Untagged: nginx@sha256:2f770d2fe27bc85f68fd7fe6a63900ef7076bc703022fe81b980377fe3d27b70
Deleted: sha256:51086ed63d8cba3a6a3d94ecd103e9638b4cb8533bb896caf2cda04fb79b862f
Deleted: sha256:c22f011a5c63a718e3155ef21b930f5583102384c8e333299913ed660baa230c
Deleted: sha256:1235ee8acd48a34c389280d8192ae79ef241d546eeea2f3416b64608d68d8538
Deleted: sha256:80ab7667b1007f2ed4b5387e7585e18d3ca1899c76449240e2890373a8e77285
Deleted: sha256:4833b18722fc3d06feafaa0f61726b1b11baa1daa0ea455e6e2ab66a7c8db283
Deleted: sha256:98b8d2ed046082a8f6c2fb2f34430f5142fea7a7078326d980b323d71640d8ff
Deleted: sha256:fe7b1e9bf7922fbc22281bcc6b4f5ac8f1a7b4278929880940978c42fc9d0229
```

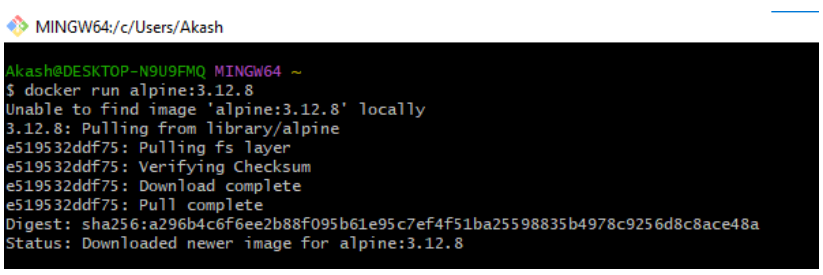
   It is very important to remove all the container references to an image before deleting the image. Check link

   https://stackoverflow.com/questions/33907835/docker-error-cannot-delete-docker-container-conflict-unable-to-remove-reposito

9. Docker run

   Get only specific image based on tag form docker hub

   - `docker run alpine:3.12.8`



Get only specific image based on tag form docker hub

```
Akash@DESKTOP-N9U9FMQ MINGW64 ~
$ docker images
REPOSITORY                    TAG       IMAGE ID       CREATED          SIZE
aakashsoni/first-docker-app   latest    6f07b6dcd25a   4 days ago       1.32GB
first-docker-app              latest    6f07b6dcd25a   4 days ago       1.32GB
docker101tutorial             latest    7e90f0a63bcb   4 days ago       28.9MB
alpine/git                    latest    b80d2cac43e4   10 days ago      43.6MB
ubuntu                        latest    216c552ea5ba   12 days ago      77.8MB
alpine                        3.12.8    48b8ec4ed9eb   13 months ago    5.58MB
```

10. Port mapping

Lets try to run some previous made application whose image we have created

```
Akash@DESKTOP-N9U9FMQ MINGW64 ~
$ docker run aakashsoni/first-docker-app
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.17.0.2:5000
Press CTRL+C to quit
```

This IP : 172.17.0.2:5000 is only accessible inside the container and if we directly run it using browser it will throw an time out error

So now we will map it to other port

- `docker run -p 8000:5000 aakashsoni/first-docker-app`

hello world

So using mapping we can run multiple instances at the same time

11. inspect

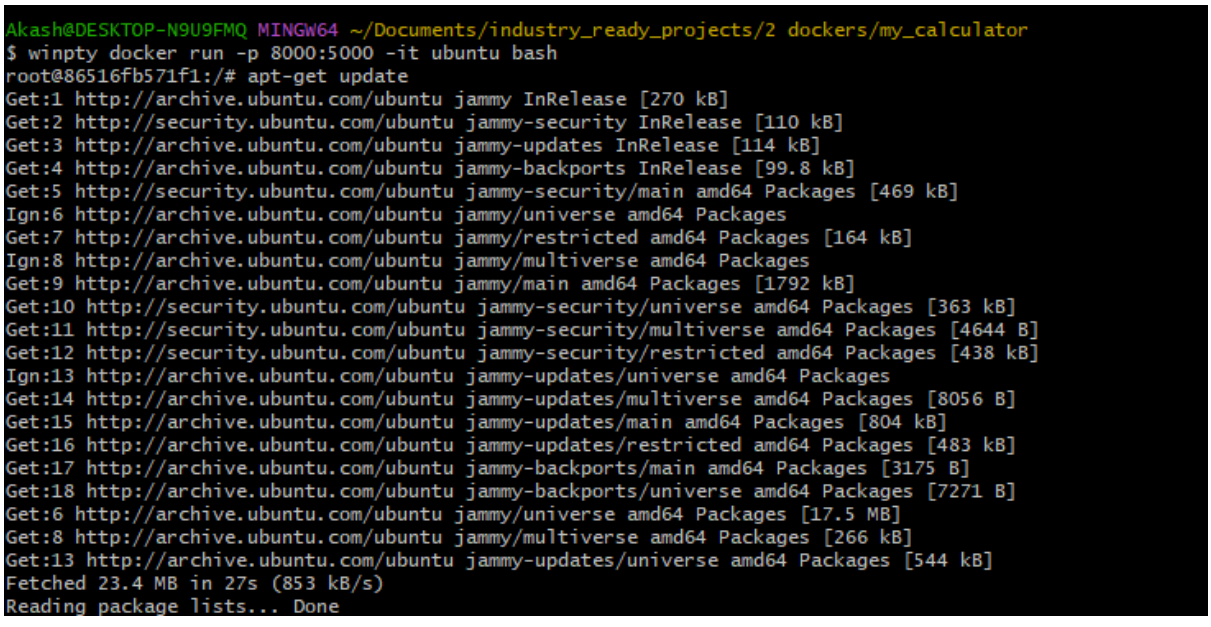
    Get full information about the containers
    - `docker inspect <container name>`

12. Running application directly into a container on ubuntu OS by installing all the docker layers
    - Running ubuntu OS image container

      `winpty docker run -p 8000:5000 -it ubuntu bash`

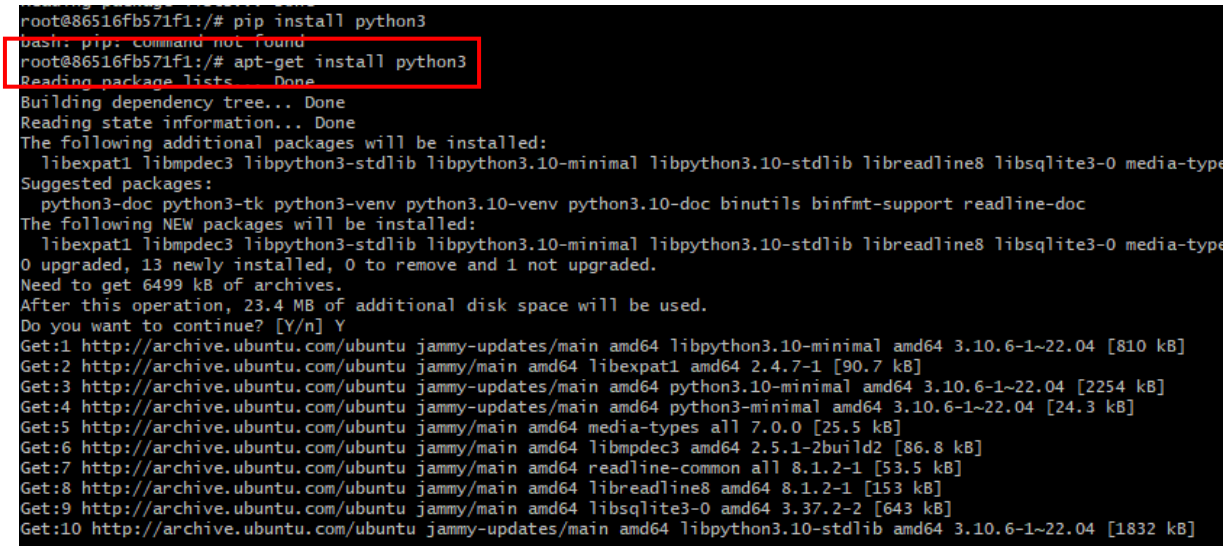
    - Update linux contatiner machine

      `root@86516fb571f1:/# apt-get update`



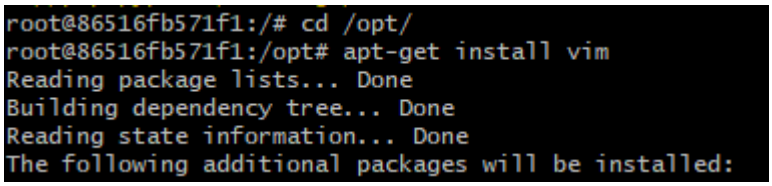
- Install python layer

```
root@86516fb571f1:/# apt-get install python3
```



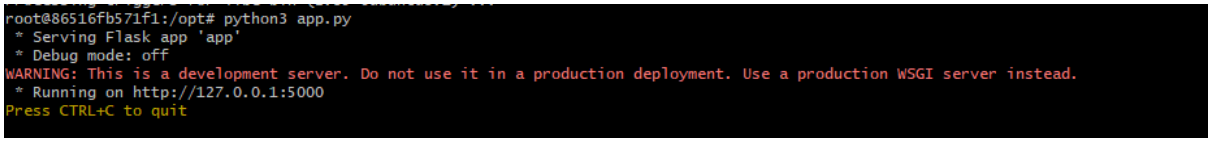
- Install requirements in the container

```
root@86516fb571f1:/# pip install flask
root@86516fb571f1:/# cd /opt/
root@86516fb571f1:/opt# apt-get install vim
root@86516fb571f1:/opt# apt install curl
```



- Lets try to create a file app.py inside the container and run it

```
root@86516fb571f1:/opt# vi app.py
root@86516fb571f1:/opt# python3 app.py
```



- Got to the terminal in docker

```
curl --location -request GET 'http://127.0.0.1:5000'
```

```
docker exec -it 86516fb571f1caacb379a8a916946d143c0e4260aafdd9eebe2db5fdc3ba7acb /bin/sh
# curl --location --request GET 'http://127.0.0.1:5000/'
curl: (3) Failed to convert -req--request to ACE; could not convert string to UTF-8
curl: (6) Could not resolve host: GET
My First Image!# curl --location --request GET 'http://127.0.0.1:5000/container'
Container Working Successfully#
```

Container working successful means our application is working fine.

We have simply done a trial just in order to observe that every thing is running fine

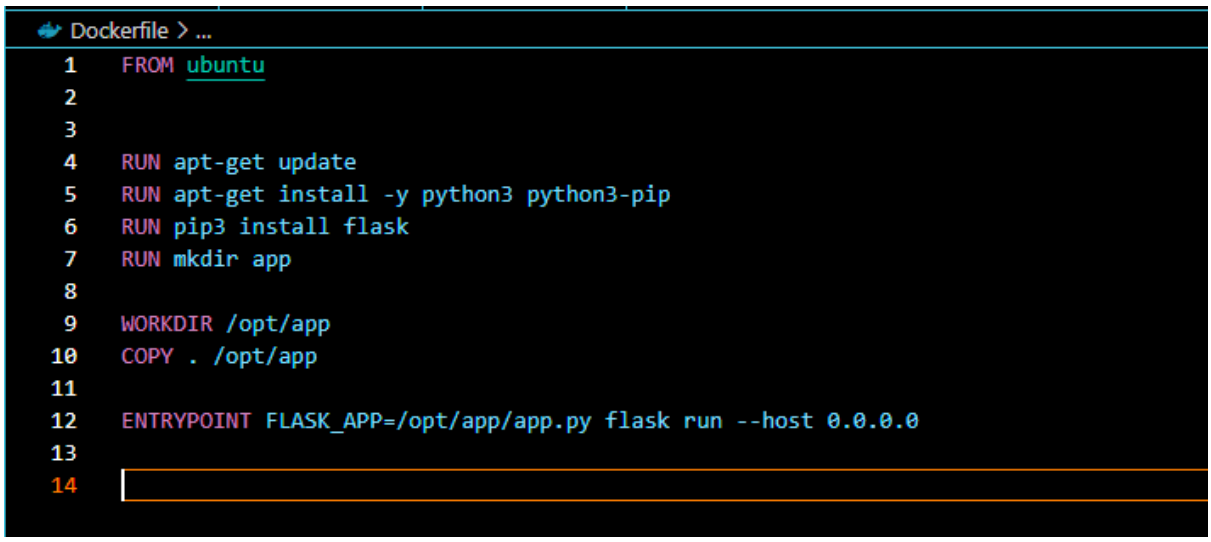
Simply run "***History***" command to get and the commands which we ran recently and using these we will create a docker file for our calculator app.

```
^Croot@86516fb571f1:/opt# history
    1  apt-get update
    2  pip install python3
    3  apt-get install python3
    4  pip install flask
    5  pip3 install flask
    6  sudo apt-get install python3
    7  apt-get install -y python3
    8  pip3 install flask
    9  apt-get install -y python3-pip
   10  pip3 install flask
   11  cd /opt/
   12  apt-get install vim
   13  vi
   14  vi app.py
   15  python3 app.py
   16  vi app.py
   17  python3 app.py
   18  apt install curl
   19  python3 app.py
   20  history
root@86516fb571f1:/opt#
```

Assignment 2:

Building a Local docker image

13. Now we will create a docker file in our app and will use the commands which are required for creating docker image



Now the final docker file is created

Lets build the local Image

- `docker build . -t cal`

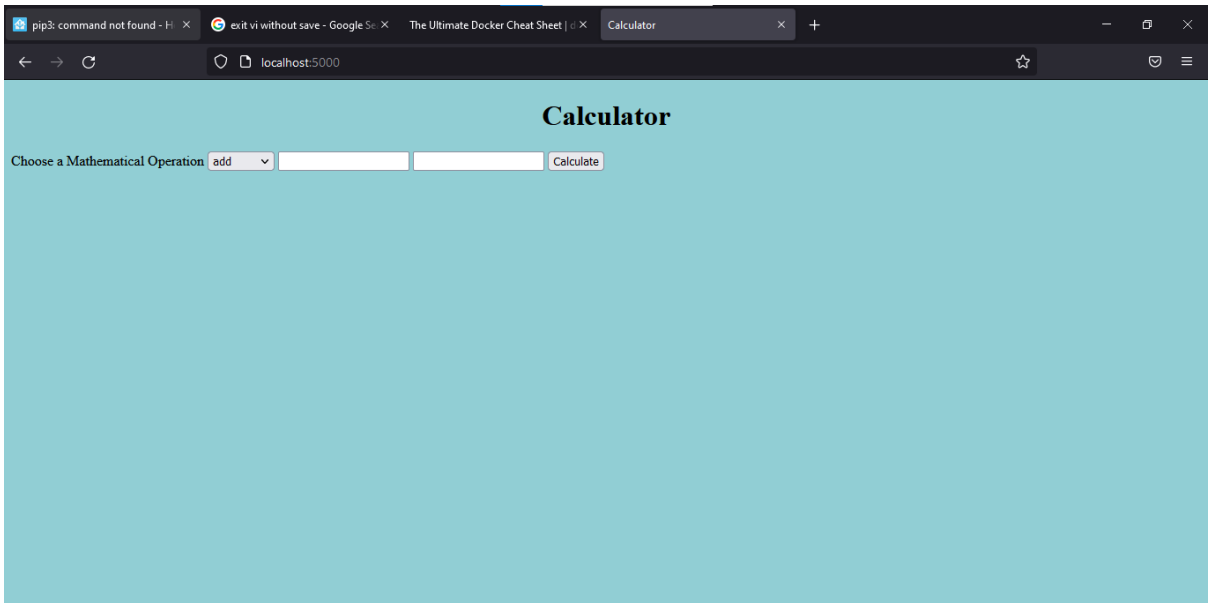
This command will build a docker image by downloading all the libraries and run time dependencies

Now a docker image has been created with name "***cal***"

```
Akash@DESKTOP-N9U9FMQ MINGW64 ~/Documents/industry_ready_projects/2 dockers/my_calculator
$ docker images
REPOSITORY                      TAG       IMAGE ID       CREATED          SIZE
cal                             latest    3cddf541084a   10 seconds ago   468MB
<none>                          <none>    9f254bf7e457   4 minutes ago    468MB
first-docker-app                latest    6f07b6dcd25a   4 days ago       1.32GB
aakashsoni/first-docker-app     latest    6f07b6dcd25a   4 days ago       1.32GB
docker101tutorial               latest    7e90f0a63bcb   5 days ago       28.9MB
alpine/git                      latest    b80d2cac43e4   11 days ago      43.6MB
ubuntu                          latest    216c552ea5ba   13 days ago      77.8MB
alpine                          3.12.8    48b8ec4ed9eb   13 months ago    5.58MB
```

Let's Run the local image

- `docker run -p 5000:5000 cal`



So the image is running locally

Assignment 3:

Now lets create a image and push it on the docker hub

14. Initially We will do a Login to docker hub first

    - `docker login`

```
Akash@DESKTOP-N9U9FMQ MINGW64 ~/Documents/industry_ready_projects/2 dockers/my_calculator
$ docker login
Authenticating with existing credentials...
Login Succeeded

Logging in with your password grants your terminal complete access to your account.
For better security, log in with a limited-privilege personal access token. Learn more at https://docs.docker.com/go/access-toke
ns/

Akash@DESKTOP-N9U9FMQ MINGW64 ~/Documents/industry_ready_projects/2 dockers/my_calculator
$
```

15. Building the image and pushing it onto the docker Hub

    - `docker build . -t aakashsoni/calculatorapp`

```
Akash@DESKTOP-N9U9FMQ MINGW64 ~/Documents/industry_ready_projects/2 dockers/my_calculator
$ docker build . -t aakashsoni/calculatorapp
[+] Building 0.6s (12/12) FINISHED
 => [internal] load build definition from Dockerfile                                0.1s
 => => transferring dockerfile: 32B                                                 0.0s
 => [internal] load .dockerignore                                                   0.0s
 => => transferring context: 2B                                                     0.0s
 => [internal] load metadata for docker.io/library/ubuntu:latest                    0.0s
 => [1/7] FROM docker.io/library/ubuntu                                             0.0s
 => [internal] load build context                                                   0.1s
 => => transferring context: 2.86kB                                                 0.0s
 => CACHED [2/7] RUN apt-get update                                                 0.0s
 => CACHED [3/7] RUN apt-get install -y python3 python3-pip                         0.0s
 => CACHED [4/7] RUN pip3 install flask                                             0.0s
 => CACHED [5/7] RUN mkdir app                                                      0.0s
```

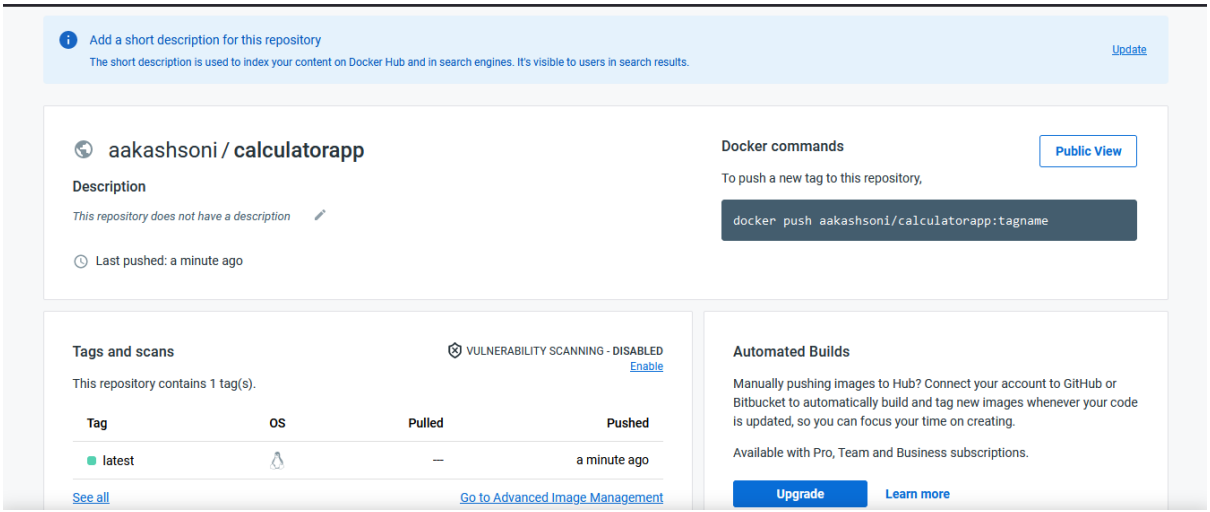
    - `docker images`

```
$ docker images
REPOSITORY                  TAG       IMAGE ID       CREATED             SIZE
aakashsoni/calculatorapp    latest    280a8eddda70   About a minute ago  468MB
cal                         latest    a7709cd89fec   33 minutes ago      468MB
<none>                      <none>    3cddf541084a   36 minutes ago      468MB
<none>                      <none>    9f254bf7e457   40 minutes ago      468MB
aakashsoni/first-docker-app latest    6f07b6dcd25a   4 days ago          1.32GB
first-docker-app            latest    6f07b6dcd25a   4 days ago          1.32GB
docker101tutorial           latest    7e90f0a63bcb   5 days ago          28.9MB
alpine/git                  latest    b80d2cac43e4   11 days ago         43.6MB
ubuntu                      latest    216c552ea5ba   13 days ago         77.8MB
alpine                      3.12.8    48b8ec4ed9eb   13 months ago       5.58MB
```

Pushing image form local to docker hub

- `docker push aakashsoni/calculatorapp`

```
Akash@DESKTOP-N9U9FMQ MINGW64 ~/Documents/industry_ready_projects/2 dockers/my_calculator
$ docker push aakashsoni/calculatorapp
Using default tag: latest
The push refers to repository [docker.io/aakashsoni/calculatorapp]
895d855fecf0: Pushed
29ea141b8032: Pushed
c5494d5d4189: Pushed
fe22de897214: Pushed
26e654c38998: Pushed
5c42d08536ca: Pushed
17f623af01e2: Mounted from library/ubuntu
latest: digest: sha256:a5edea1c5e410e0e88ff21ab06be9916b70369e912d4ae501ebbe667e760e45c size: 1787
```



Now this repository is visible globally.

Assignment 4:

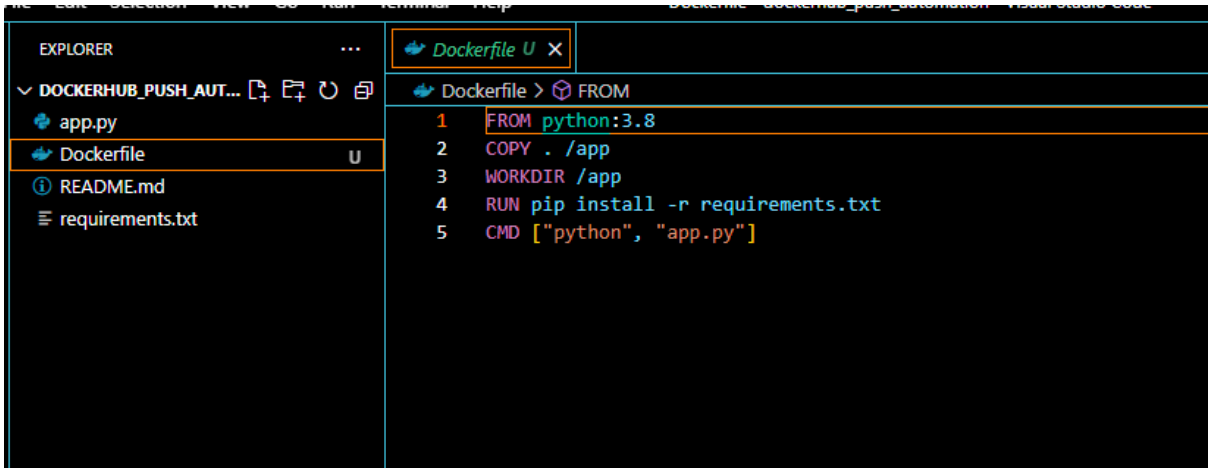
Automate Assignment below task using github action.

1. Build Docker Image
2. Push Docker Image to Docker hub.

Step 1: Create a new repo on github https://github.com/akash-soni/dockerhub_push_automation.git

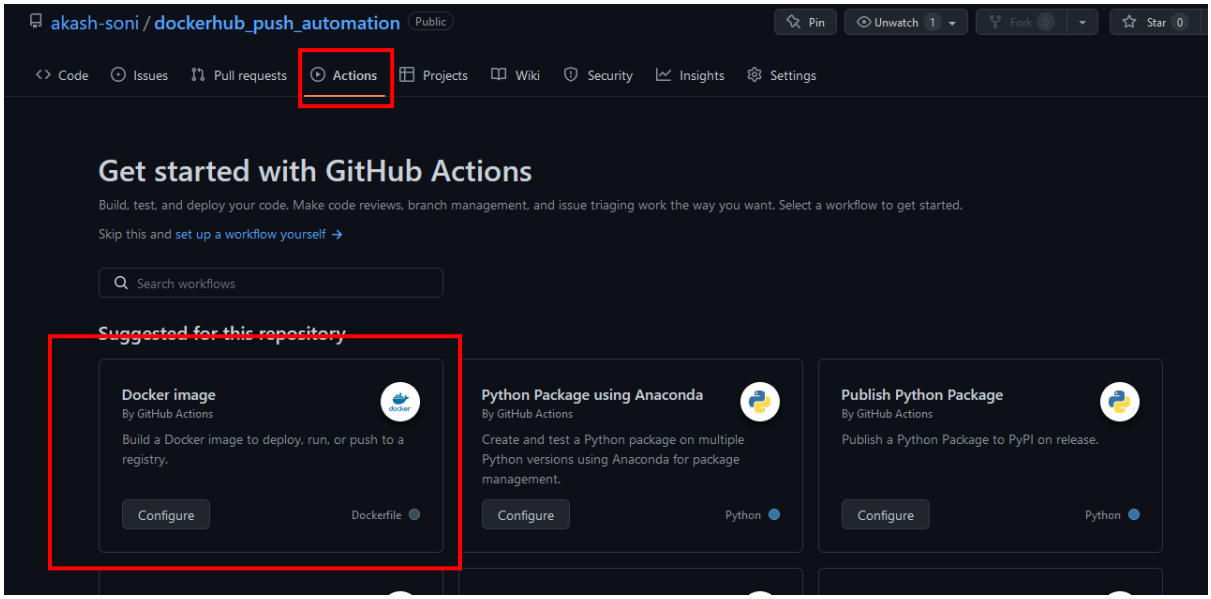
Step 2: Git clone and create a small “Hello world” application

Step 3: Create a docker file

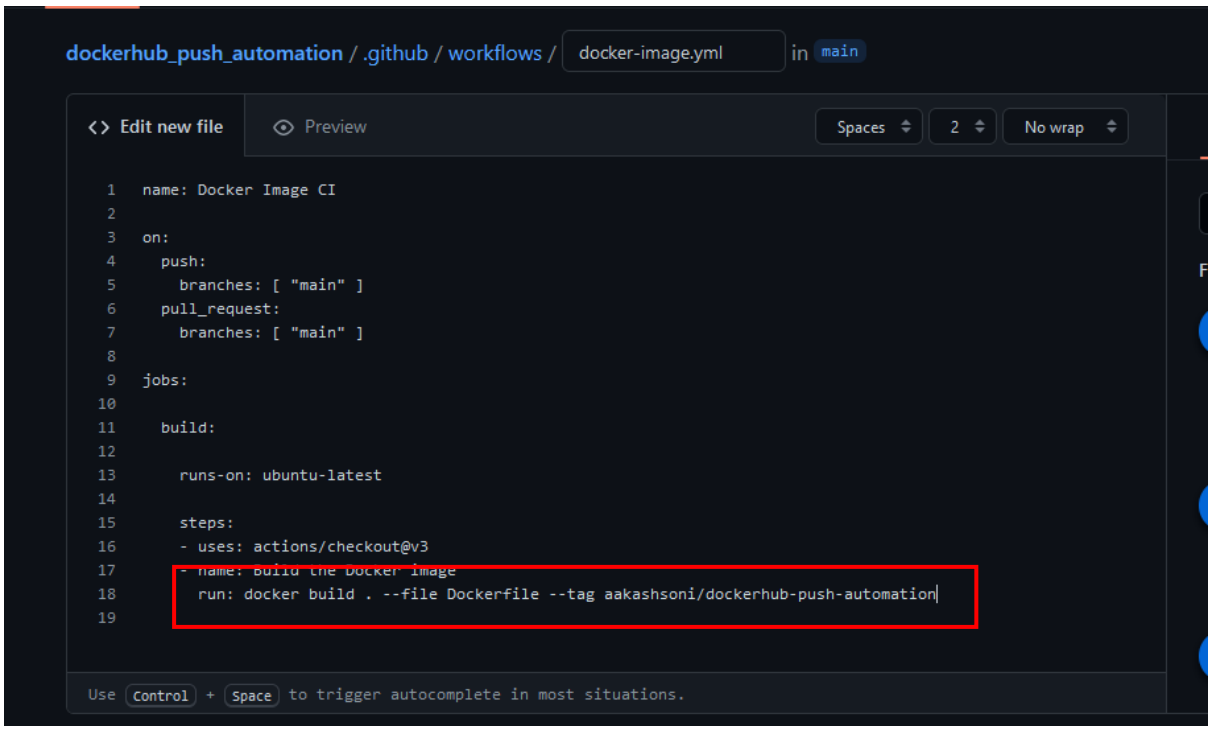


Step 4 : Now push all these changes onto Repository

Step 5 : Github Actions provide Docker Configuration



Step 6: Github already provides workflow template for pushing with a build command, here simply change give the name with which you want to push image on docker hub



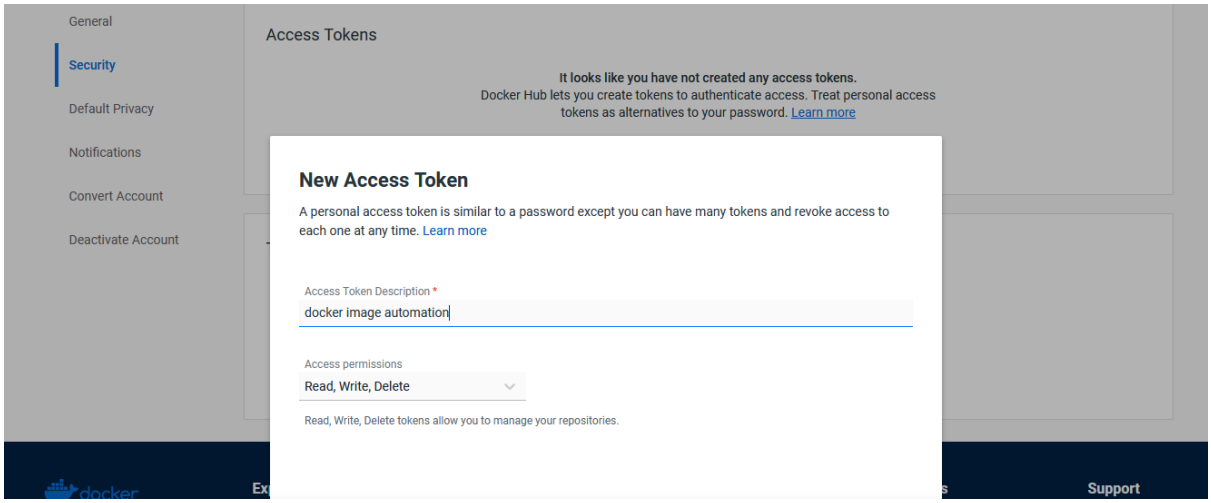
Observe that template only provides code to build the image but not to push the image.

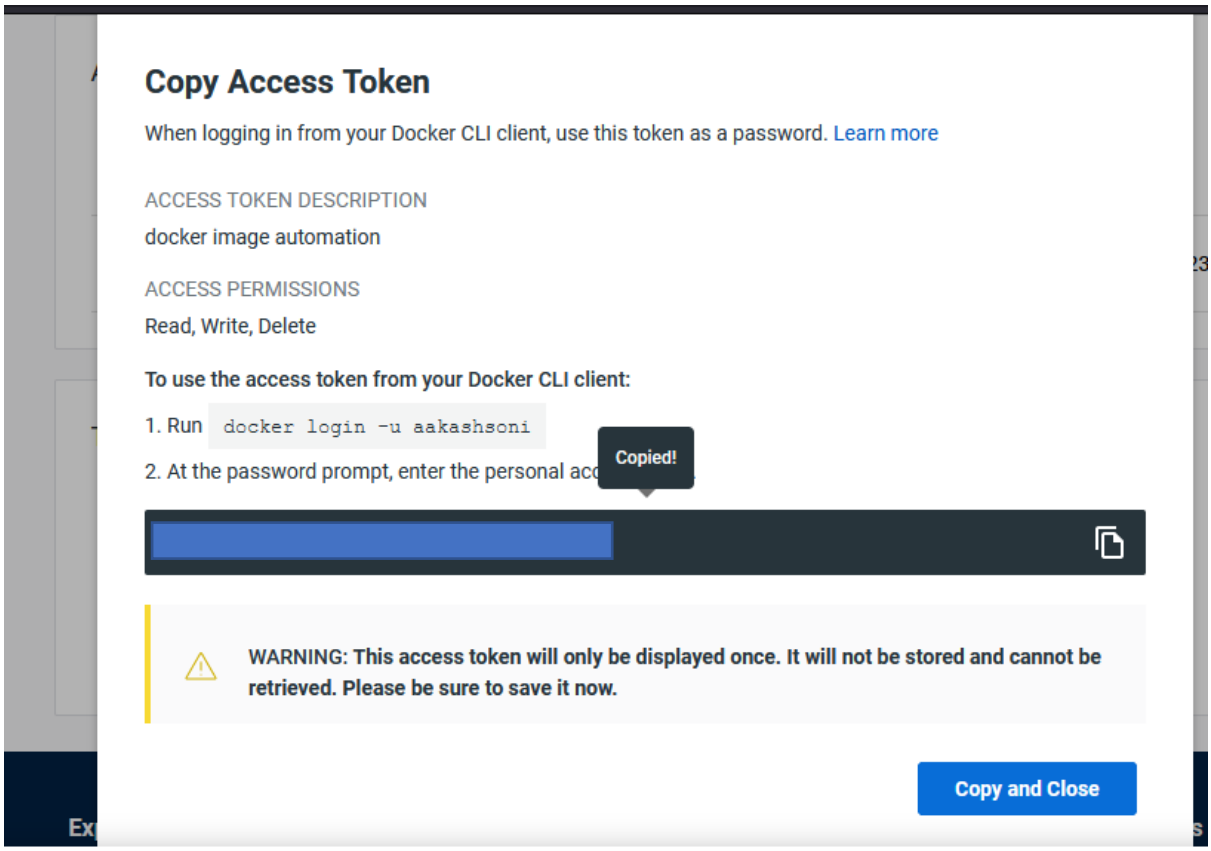
Step 7: We manually write step to login into docker hub and push image into docker hub.

Now we will create and commit this file.

Step 8: Obtain docker hub token

Go to docker hub→Profile → Account Settings → Security → New access token → give description and obtain the token. Make sure to keep token safe as it is like a password

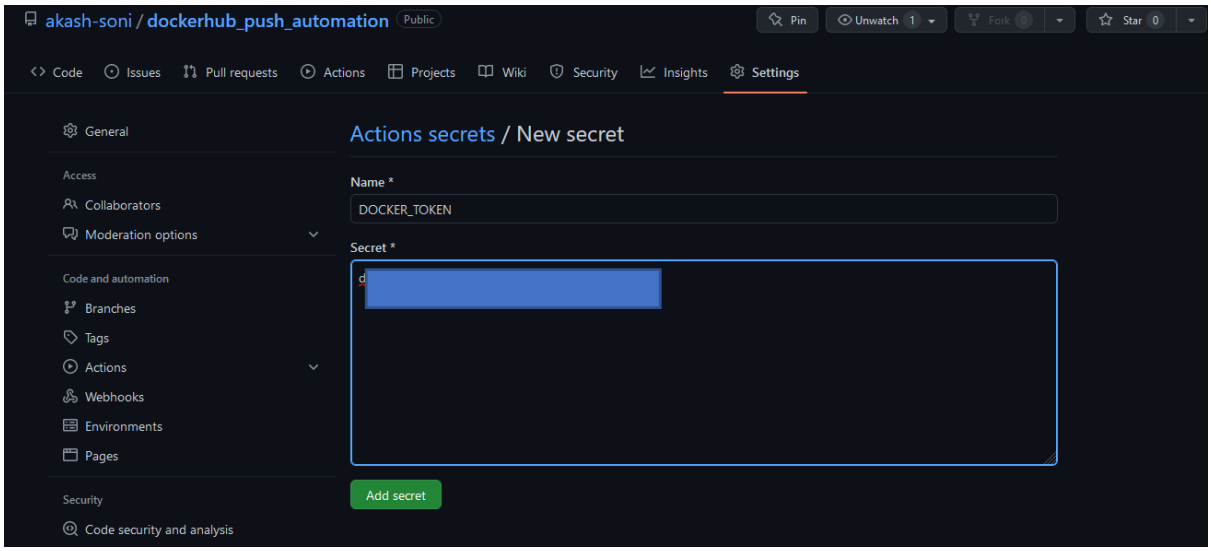




Copy the token and keep it some place as it will not be available next time

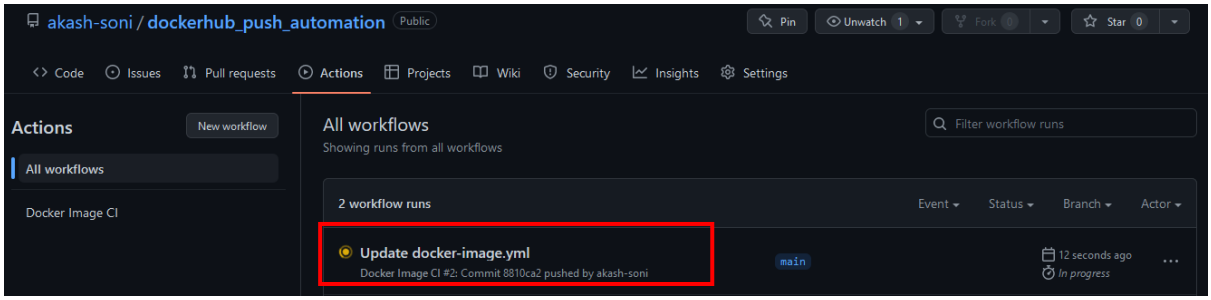
Step 9: Get back to Github → Repository Setting → Serets

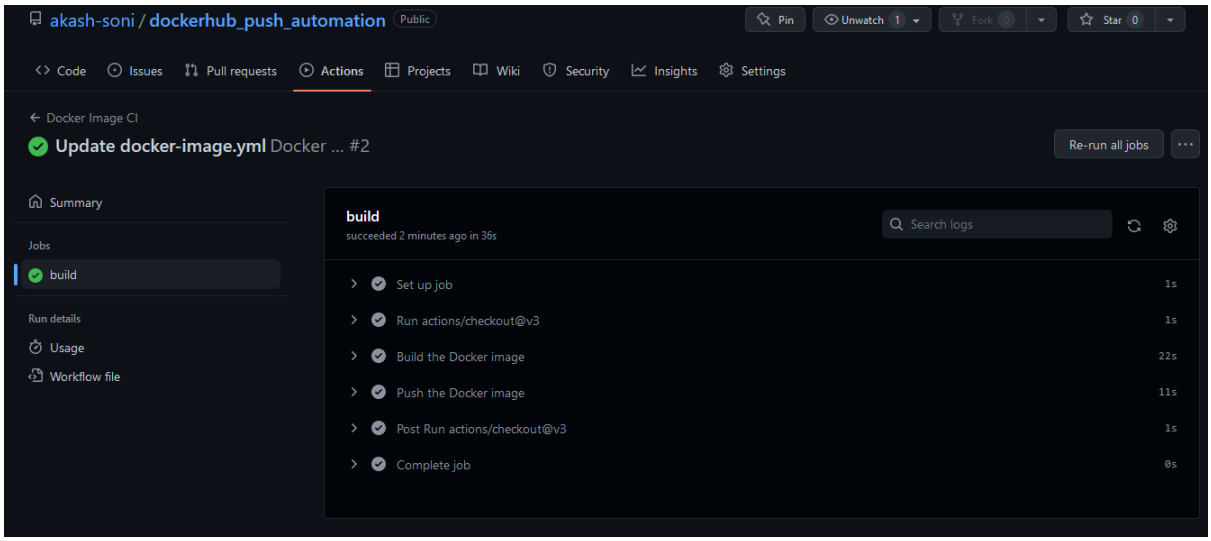
Create the Secret token and paste the secret key.



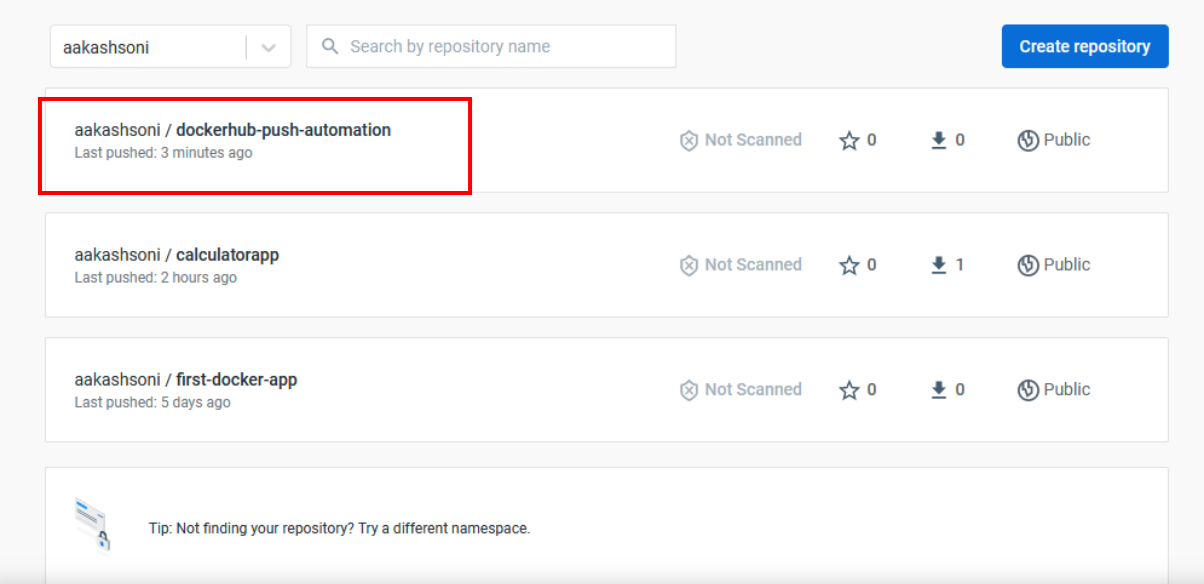
Click on Add secret.

Step 10: Now go to Actions tab, there we will notice that the docker image has started building





Now lets check on docker hub, whether the image has been pushed or not



So clearly, we can observe that the image has been pushed.

Hence, we have automated docker image pushes to docker hub.

Step 11: We do not want to trigger docker push for every commit so we can add a manual button "***Run workflow***"

Simply add line "workflow dispatch" into docker-image.yml file and commit the changes