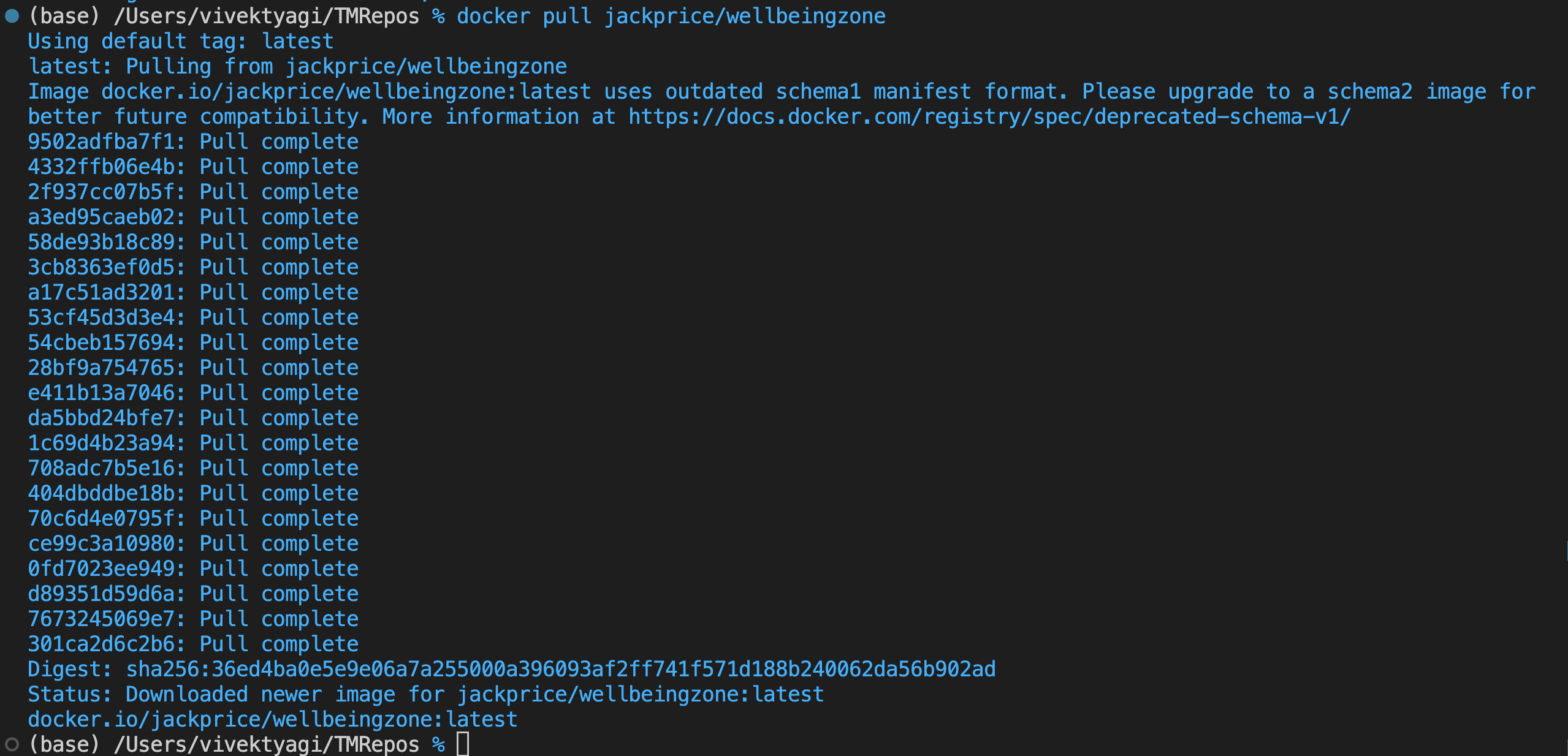
**Docker & Docker Hub: Assignment 2**

**Assignment 1:** Demonstrate minimum 15 basic docker command with explanation and screenshot.

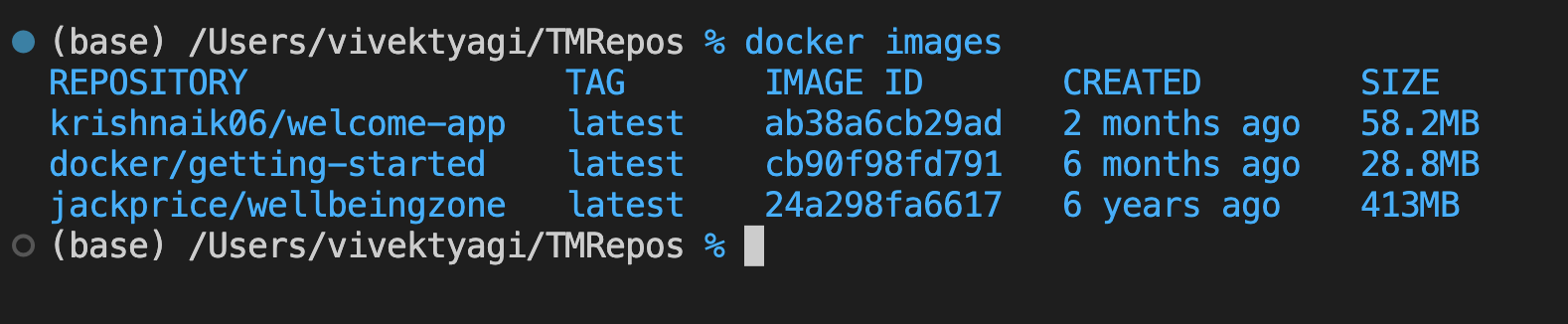
1. $ docker pull jackprice/wellbeingzone

Pulls image from docker hub. Giving account specific information in the image name, image from any account (accessible) can be downloaded.



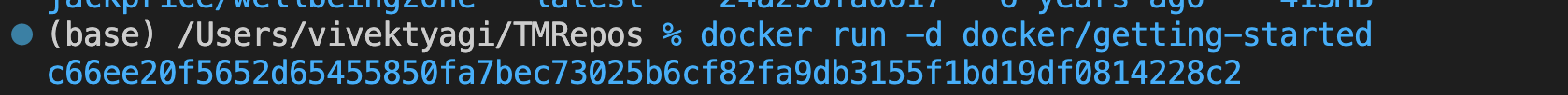
1. $ docker images

List out all the docker images on the host



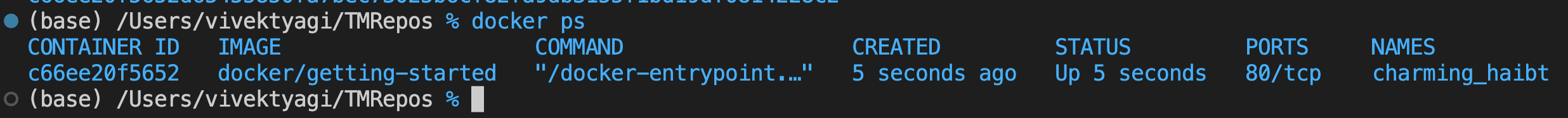
1. $ docker run docker/getting-started

Docker first looks for local availability of the imageName, if not found locally, then downloads from DockerHub and runs in container.



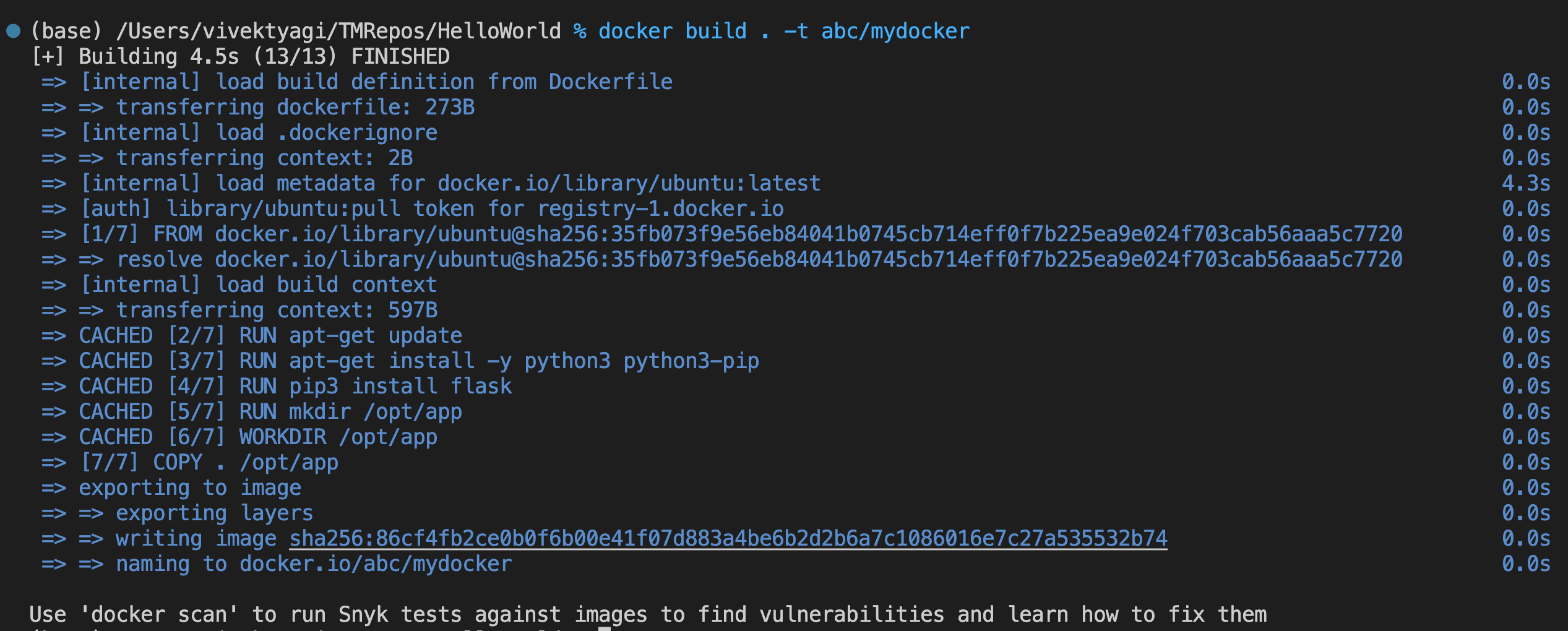
1. $ docker ps

Lists all the processes (container running)



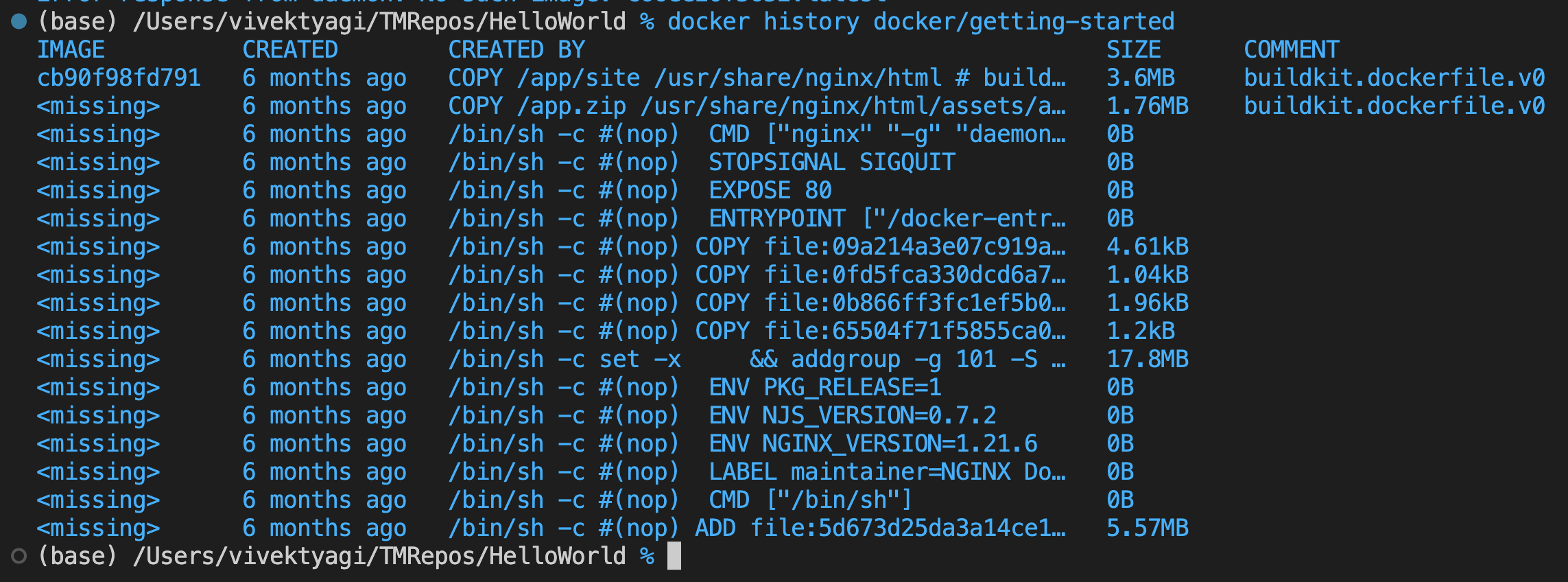
1. $ docker build . -t abc/mydocker

Build a docker image from a given Dockerfile



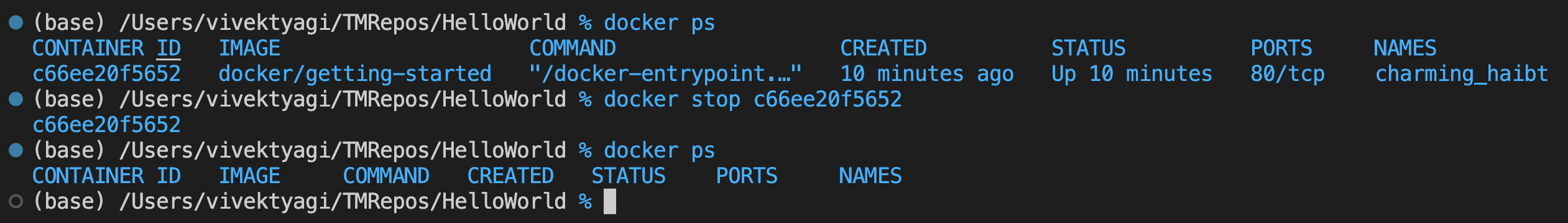
1. $ docker history docker/getting-started

Shows history of layer formation, time, size etc. of a container.



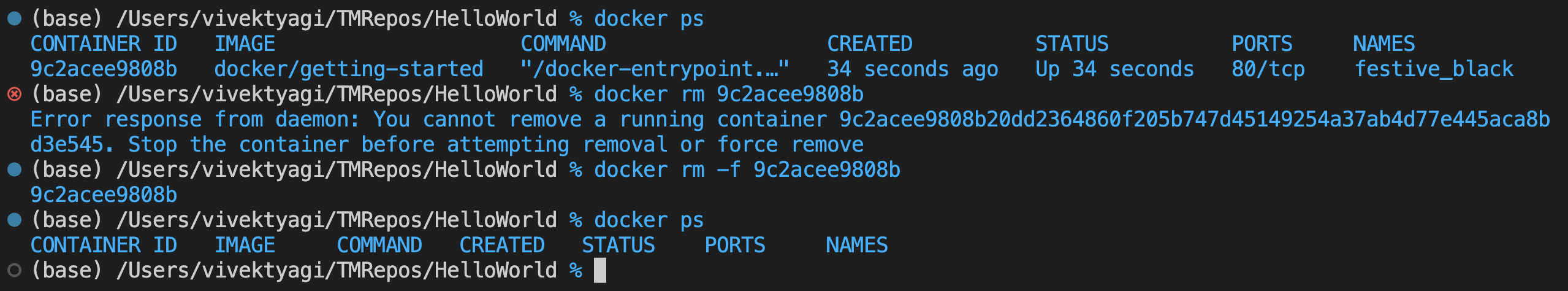
1. $ docker stop docker/getting-started

Used to stop the run of a container. Unique container name or the ImageID is required to identify the container.



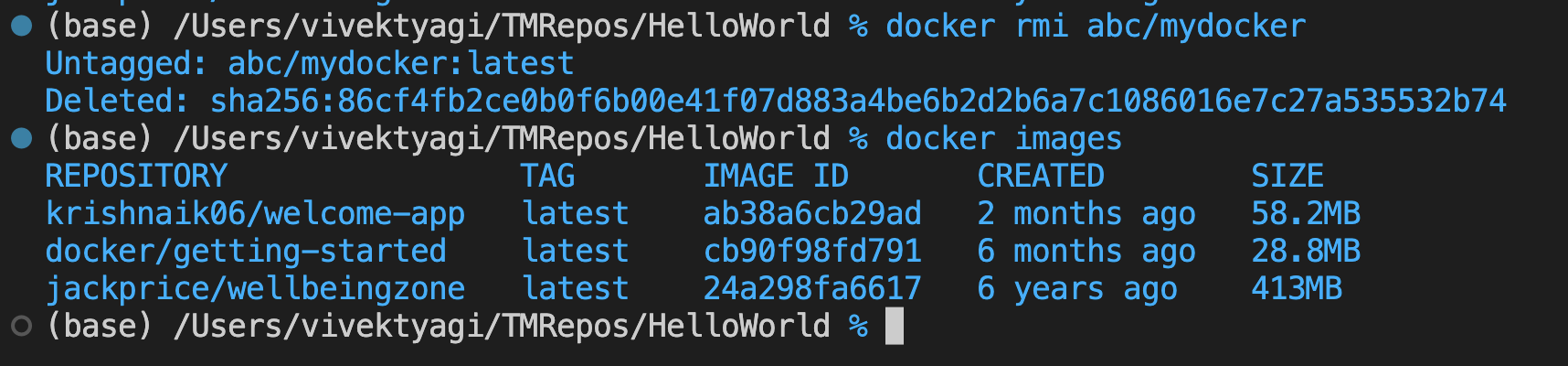
1. $ docker rm -f docker/getting-started

Force remove container (not image) from the containers list.



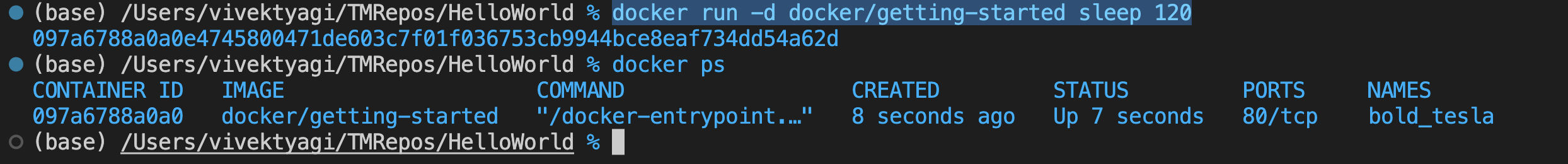
1. $ docker rmi abc/mydocker

Removes image from the host.



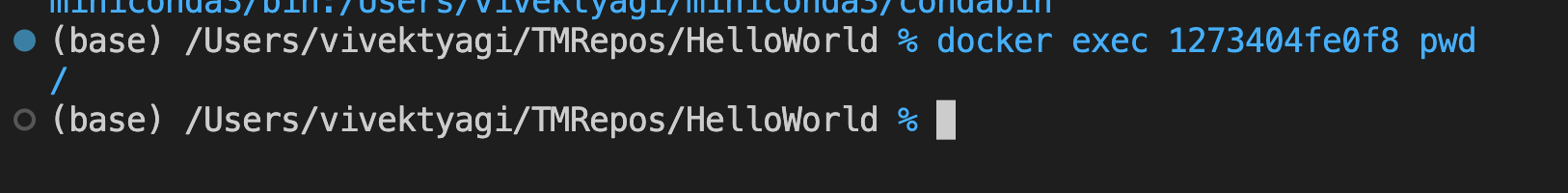
1. docker run -d docker/getting-started sleep 120

Passing a parameter to the container. Here sleep is passed a value 120.



1. $ docker exec docker/getting-started echo $PATH

The docker exec command runs a new command in a running container.



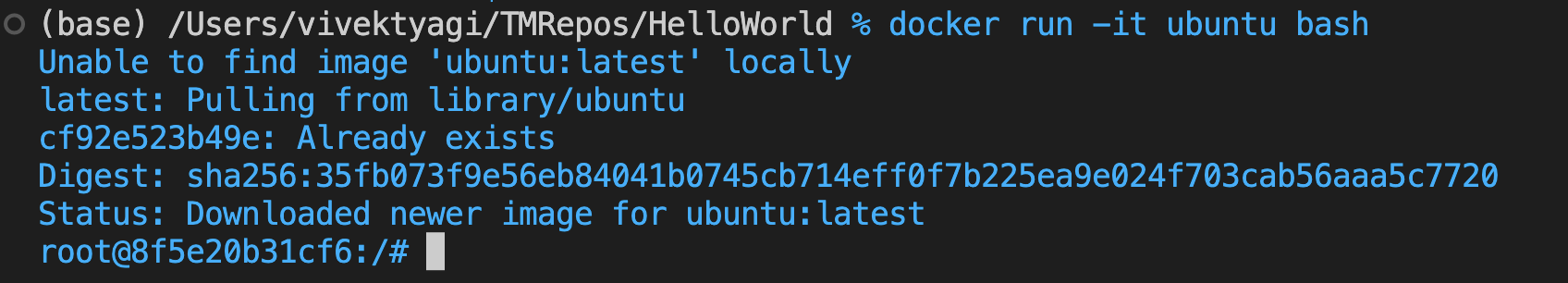
1. $ docker run -it ubuntu bash

To run commands inside a container and in interactive mode.

-i : interactive -> Keeps STDIN open even if not attached.

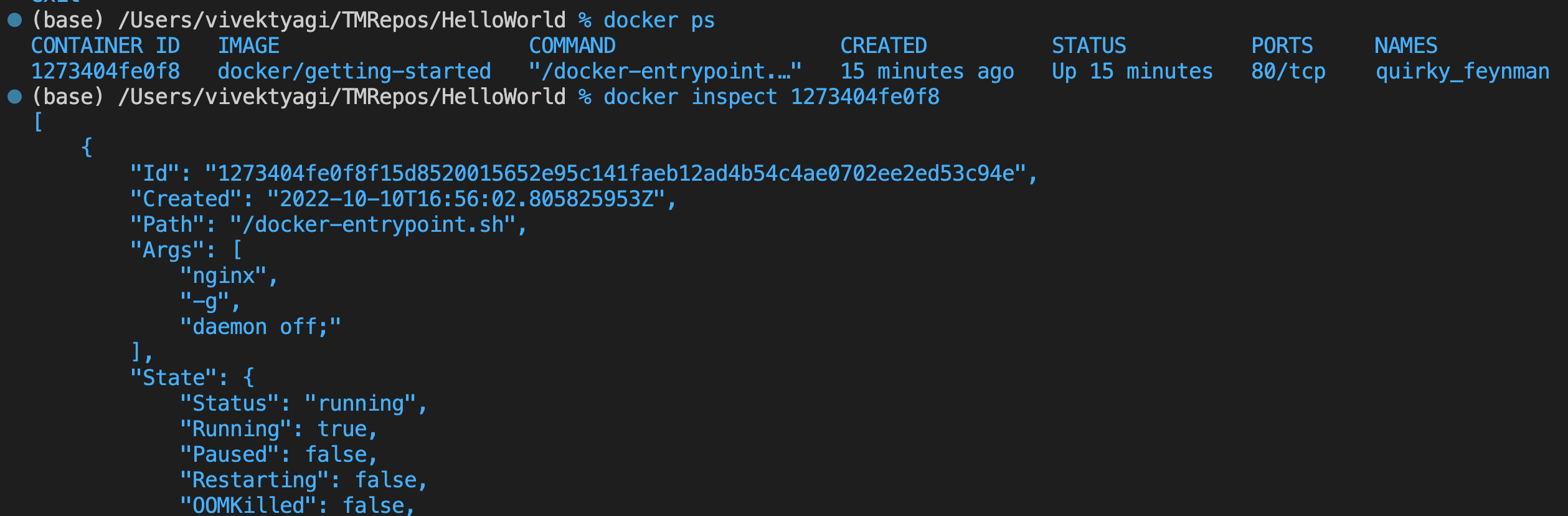
-t : Allocates pseudo terminal

The following command launches a bash shell inside the container. User can run shell commands here.



1. $ docker inspect 1273404fe0f8

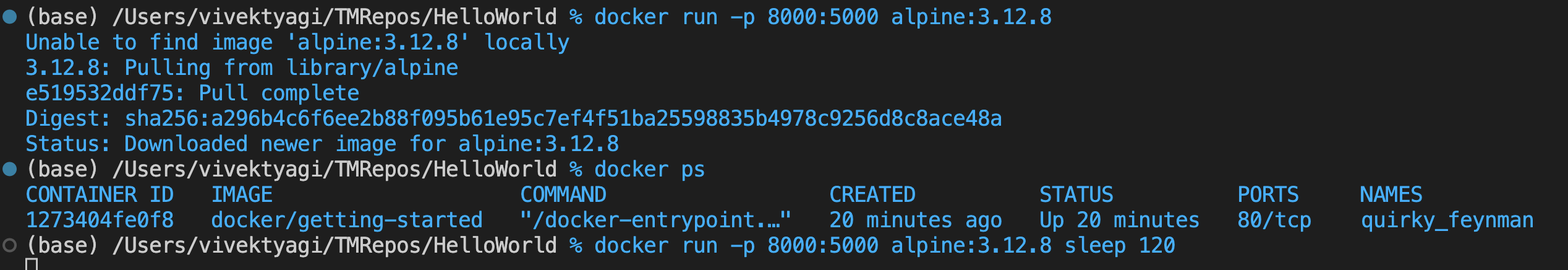
To inspect a particular container details. Can also do this using Docker Desktop.



1. $ docker run -p 8000:5000 alpine:3.12.8

Container ports are not accessible to outside world unless mapped to the host's port. '-p' flag options maps that for us.

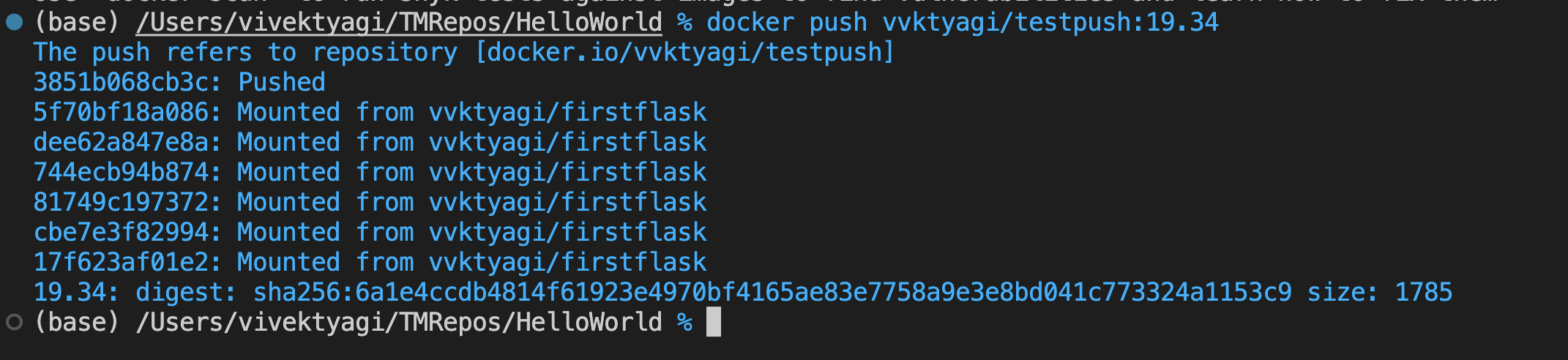
Here port 8000 of the host is mapped to port 5000 of the container. Now we can access the service/API using browser to test.

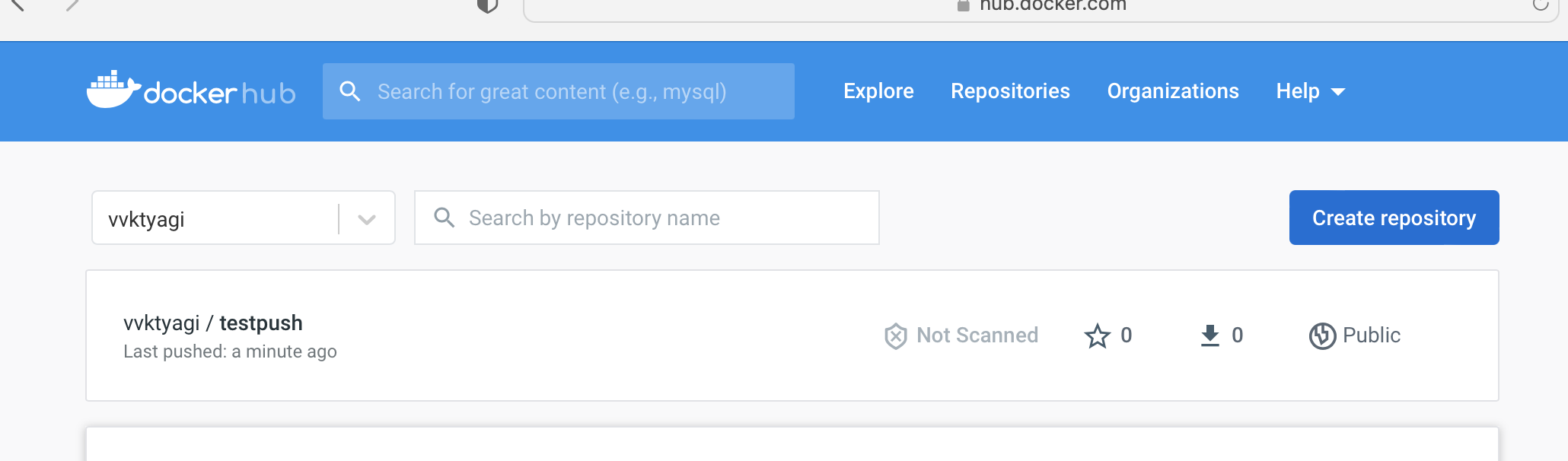


1. $ docker push vvktyagi/testpush:19.34

This is the way to publish your local docker image into docker-hub account.

This would need username and password for docker hub. If tag name is not given 'latest' would be assumed.



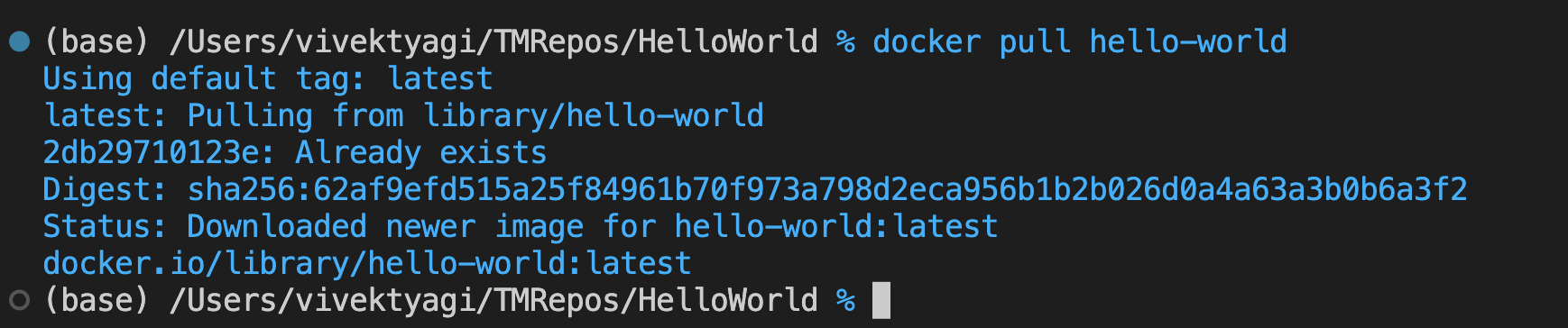


**Assignment 2:**

[Hello World Docker Image](https://hub.docker.com/_/hello-world) Run Hello World Docker Image Locally.

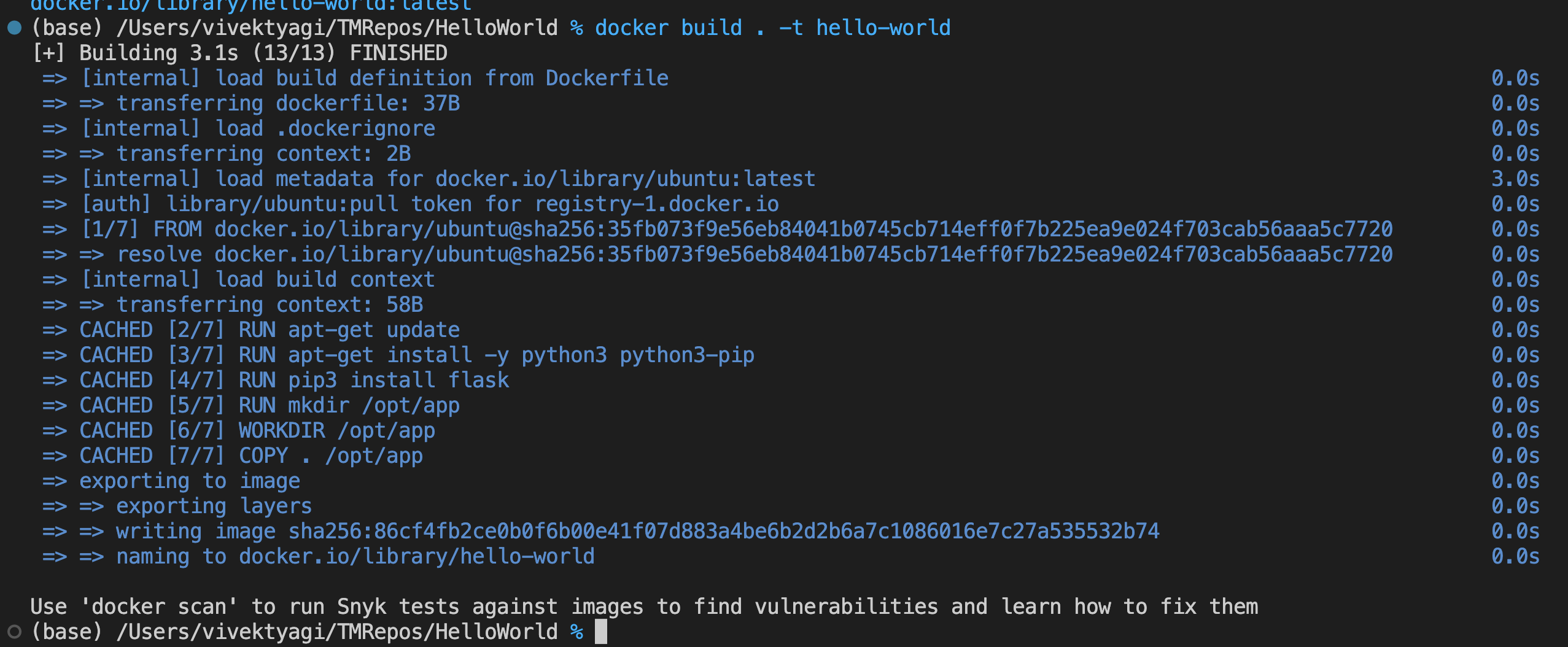
1. Get the docker locally first

$ docker pull hello-world



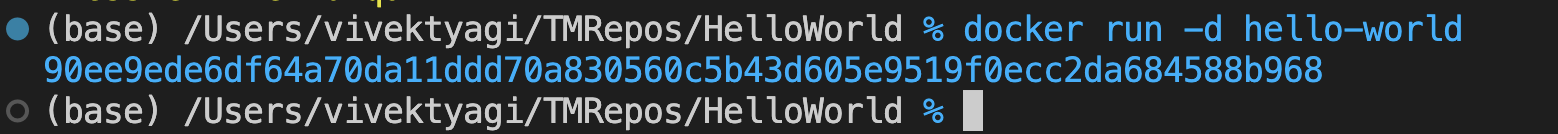
1. Build the docker image

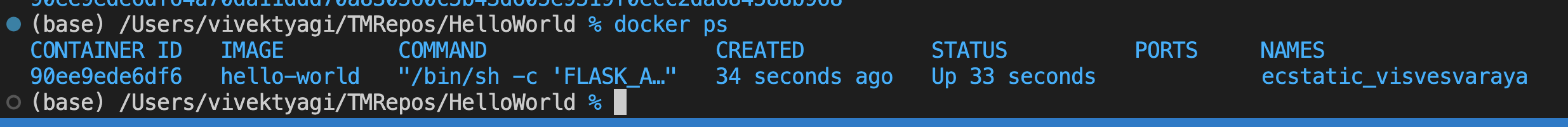
$ docker build . -t hello-world



1. Run the container having this image.

$ docker run hello-world





**Assignment 3:**

Create a hello world fastapi application. Create a Dockerfile for your fastapi hello world application. Build Docker image using Docker file. Run docker image build in previous step. Push your Docker image to Docker Hub.

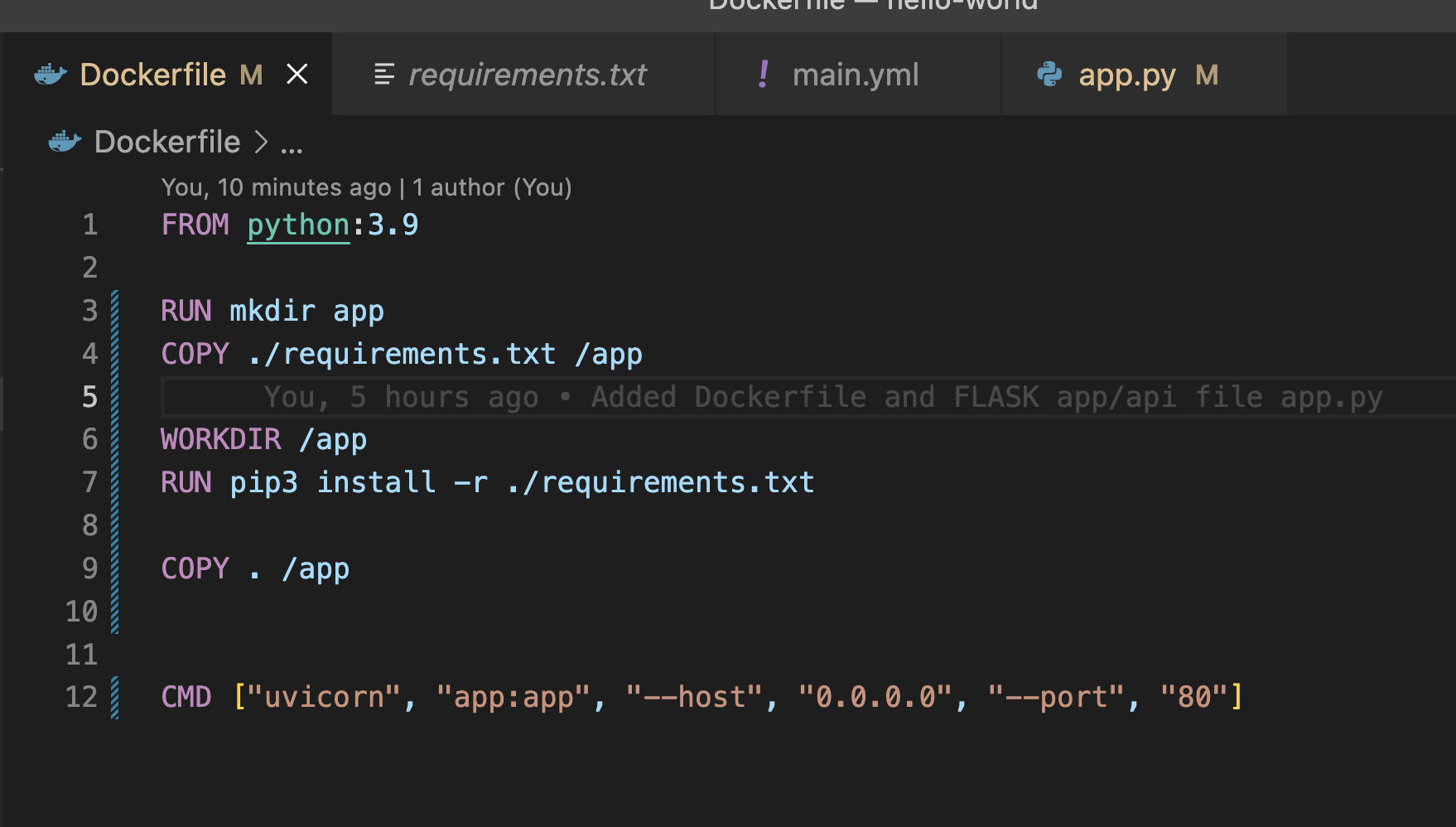
DockerHub: <https://hub.docker.com/repository/docker/vvktyagi/hello-world>

Git: <https://github.com/Data-Gyan/hello-world>

A: App code:

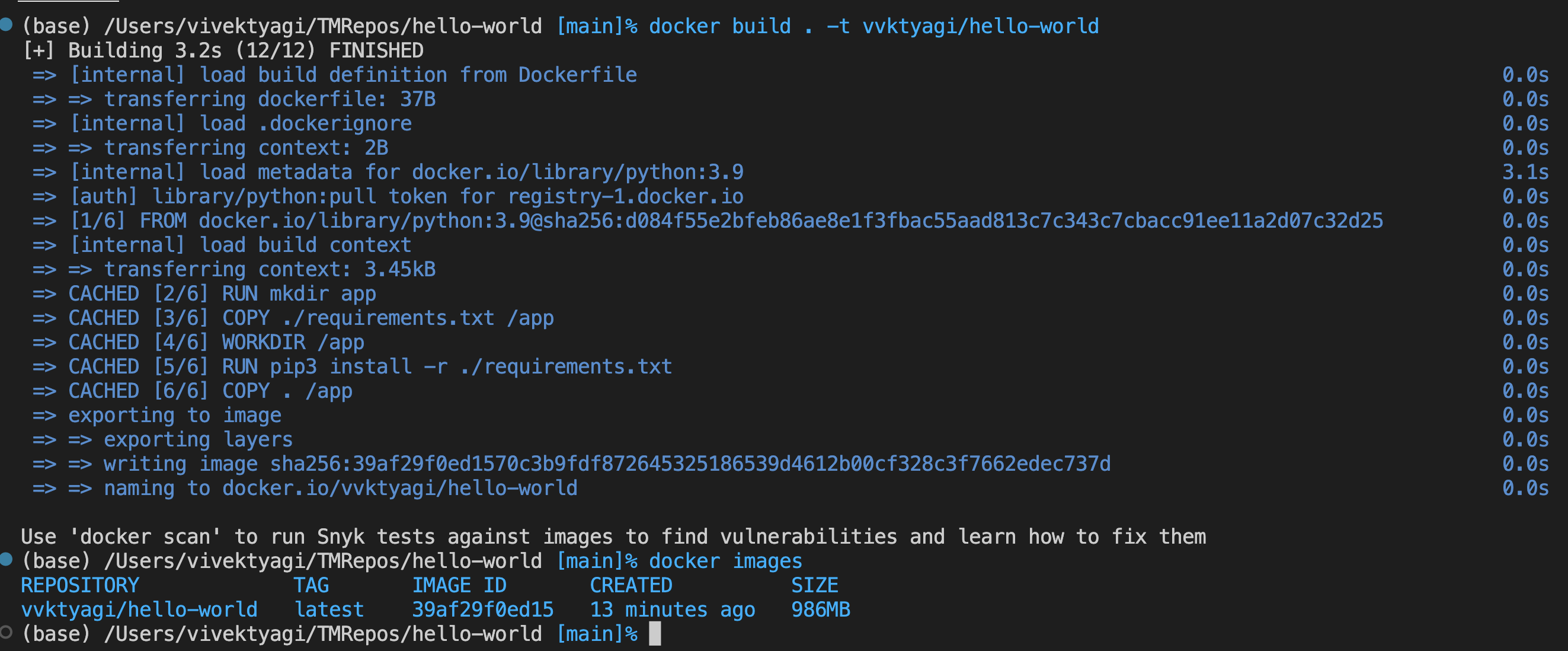


B: Dockerfile



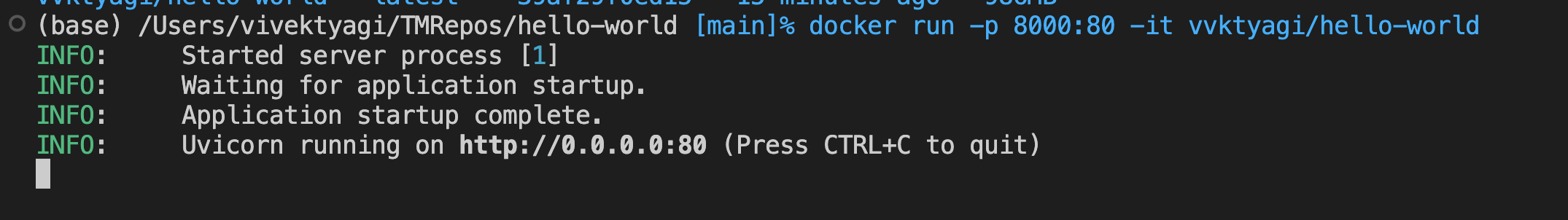
C: Build the Dockerfile to create docker image

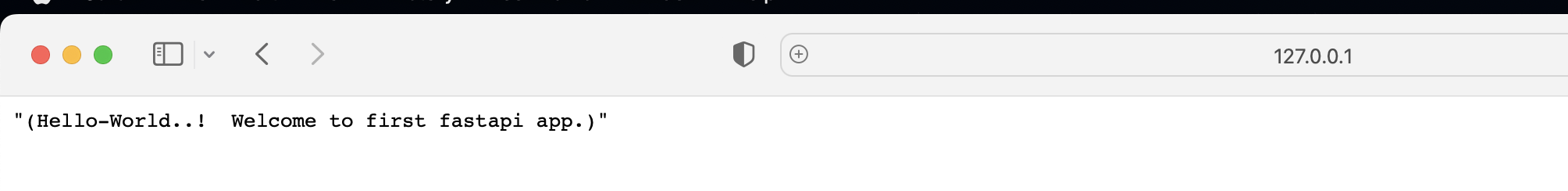
$ docker build . -t vvktyagi/hello-world



D: Run the docker image as container

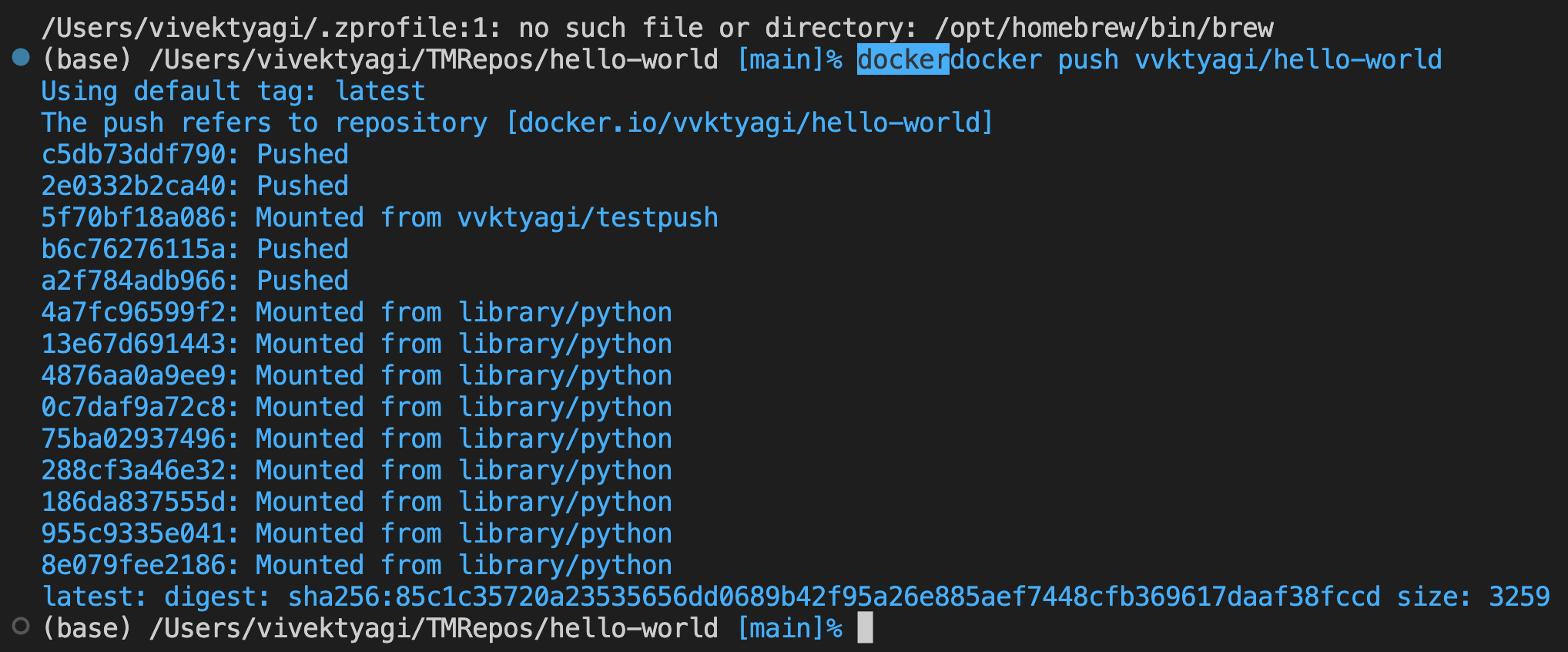
$ docker run -p 8000:80 -it vvktyagi/hello-world

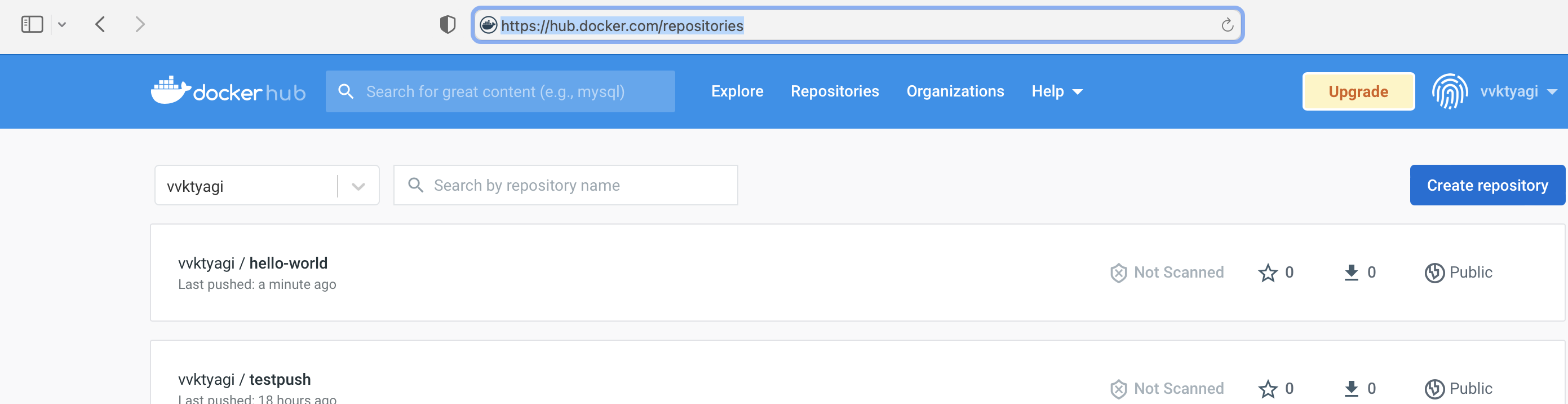




E: Push the docker to docker-hub repository

$ docker push vvktyagi/hello-world





**Assignment 4:**

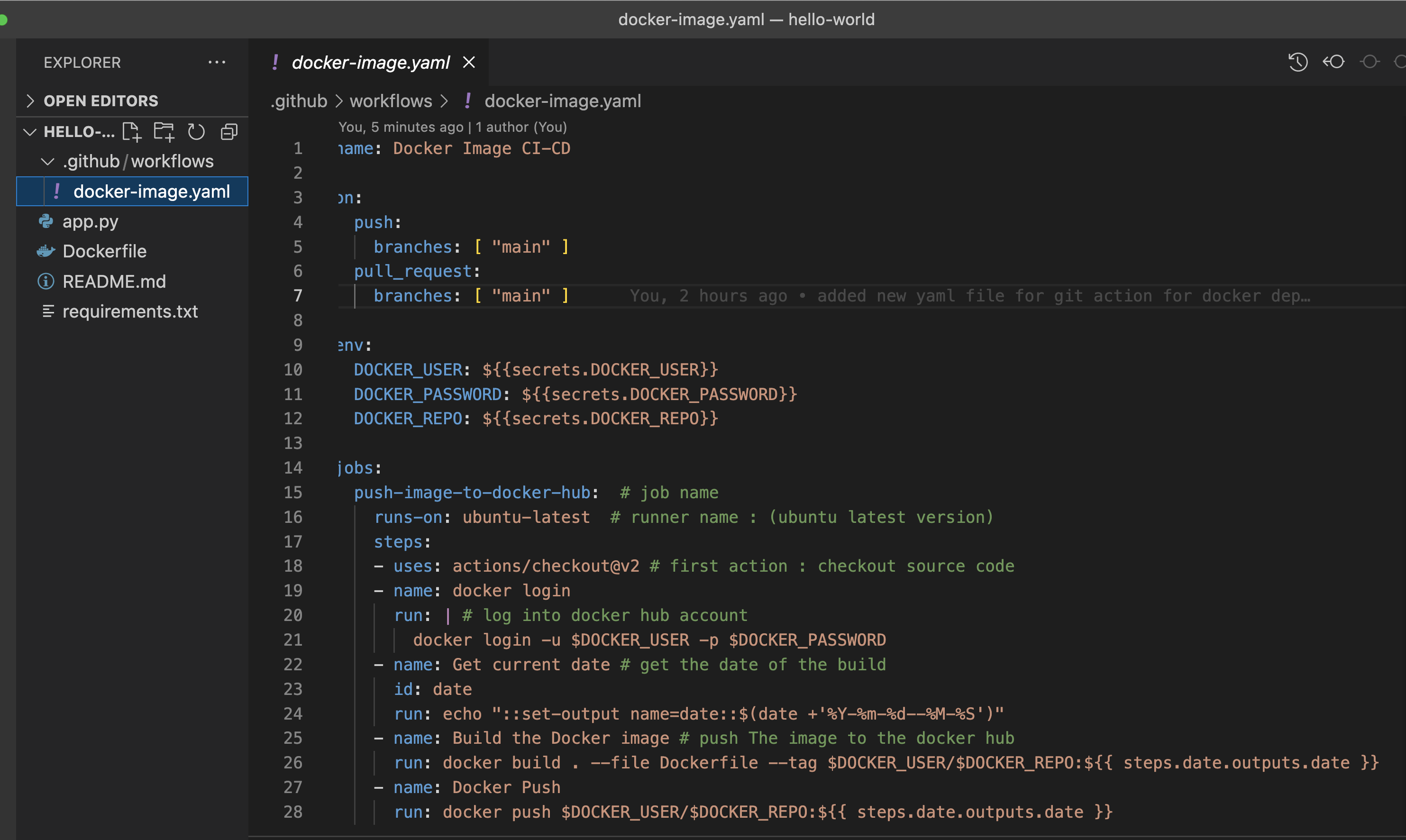
Automate Assignment below task using github action.

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

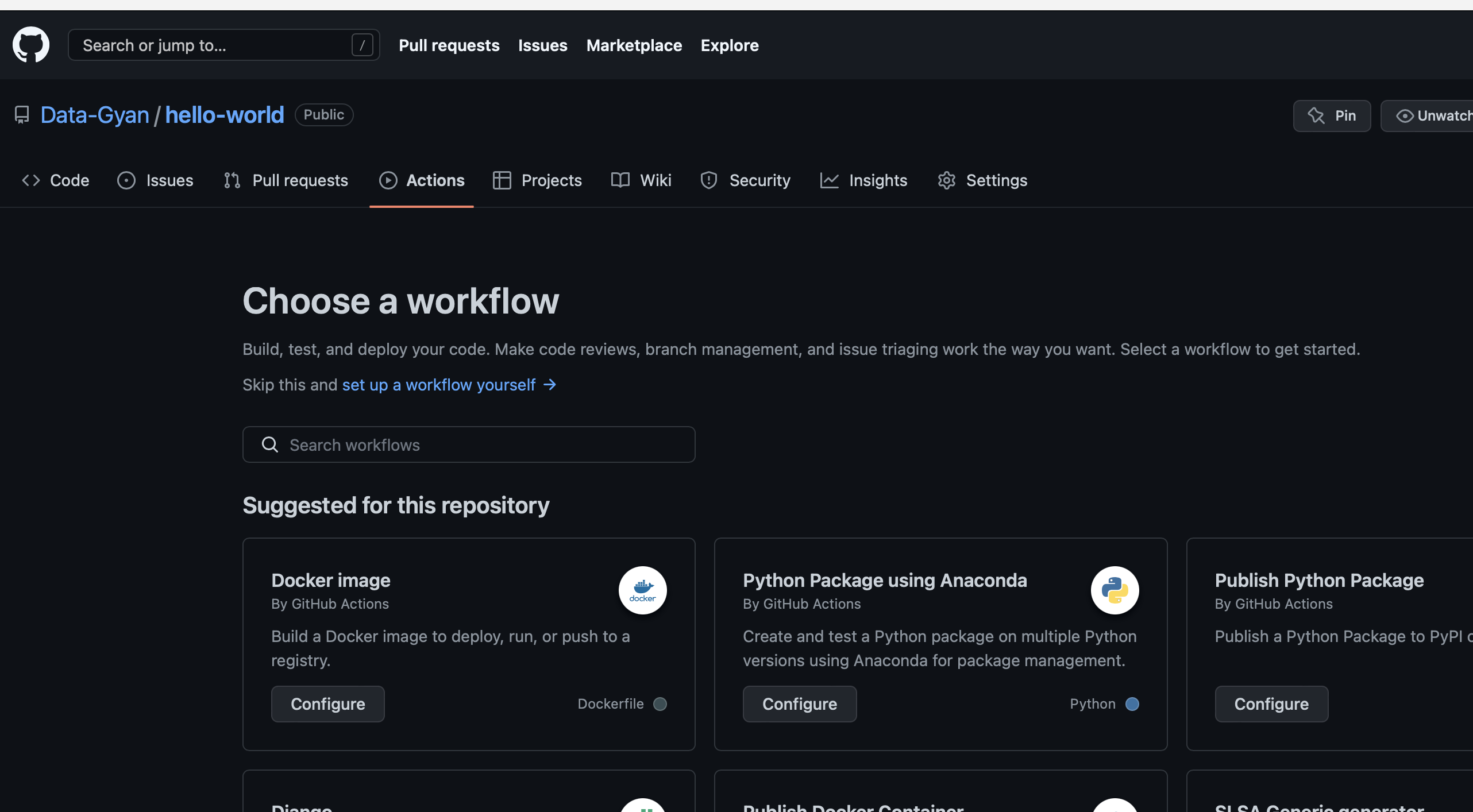
DockerHub image: <https://hub.docker.com/repository/docker/vvktyagi/hello-world>

Git repository: <https://github.com/Data-Gyan/hello-world>

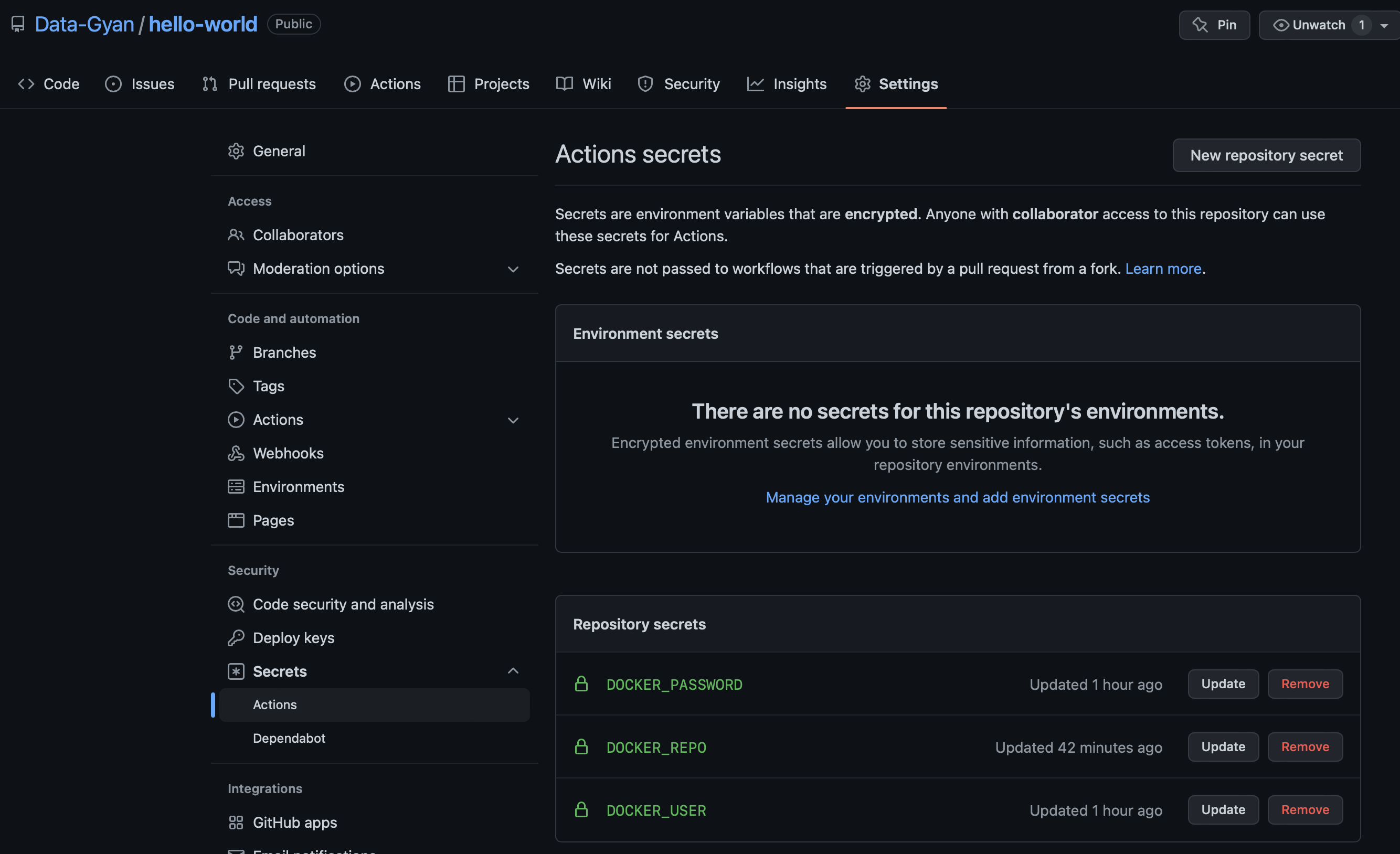
1. Add yaml file to trigger git actions in “.github/workflows”.
2. Name it as docker-image.yaml
3. Type-in the following text as shown in the picture below.



1. This can be done triggering Git Actions in GitHub repository. Select DockerImage for the workflow. It generates default yaml file in correct path.



1. Add secrets in git repository so that Git can access Docker account.

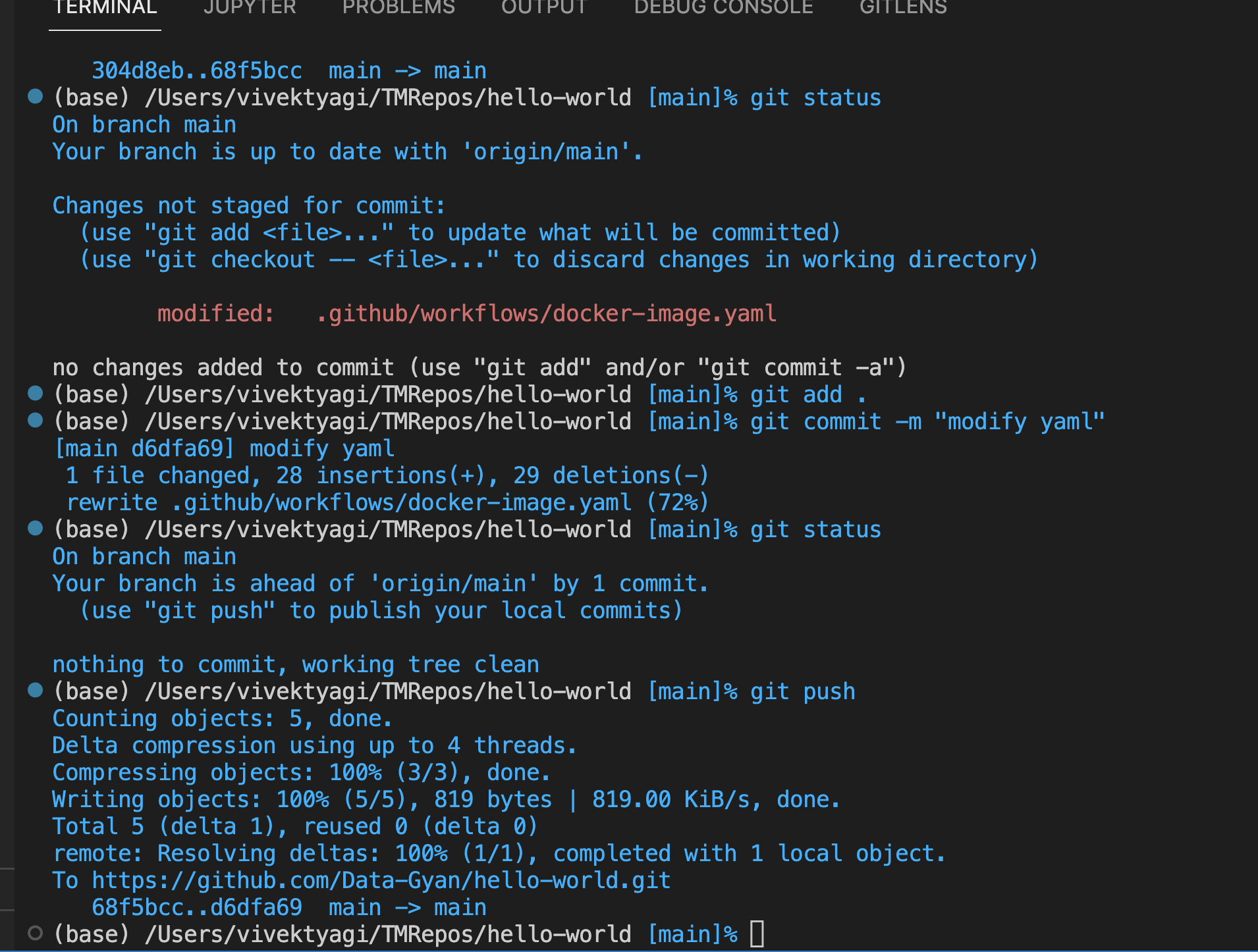


1. Our actions on git are defined for push or merge. Hence, we make code changes locally and push that to git repository.

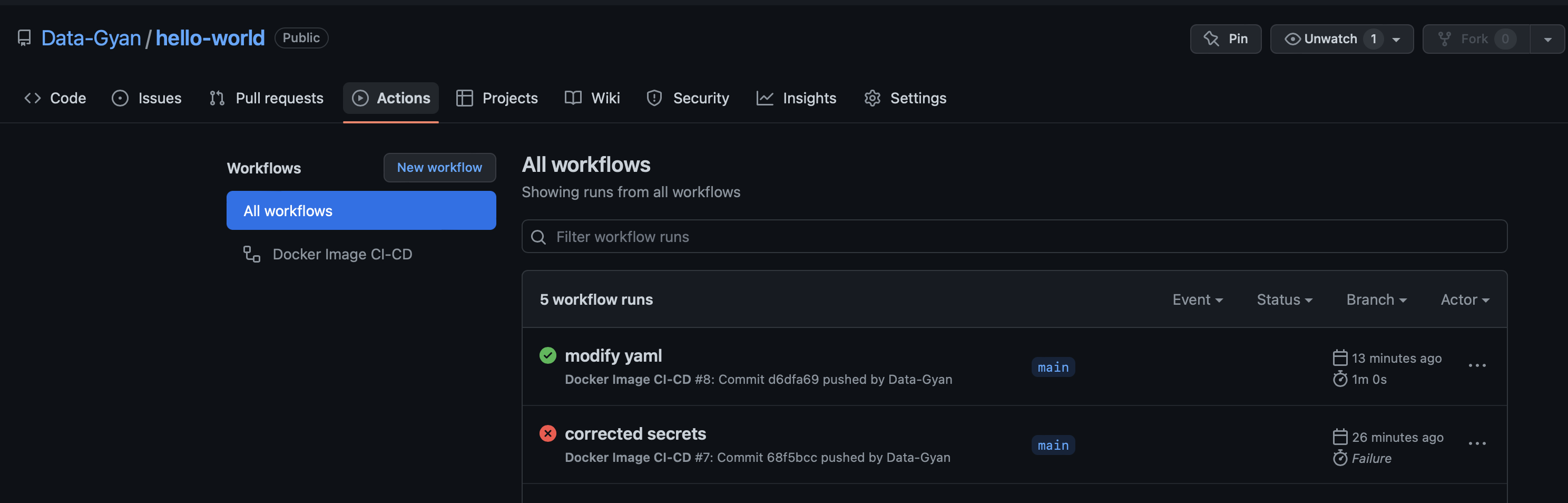
$ git add .

$ git commit -m “modify yaml”

$ git push



1. This change triggers action on the git repository



1. Resulting execution of git Action, builds and pushes the docker image on DockerHub with the repository name “hello-world”. The identifier tag is date-time.

