1) Background
   a) You don't need to submit any screenshots for the Background section.
   b) Familiarize yourself with Docker Playground: https://labs.play-with-docker.com/
   c) Familiarize yourself with Kubernetes Playground: https://labs.play-with-k8s.com/

2) Docker
   a) Go through the Python exercise in https://docs.docker.com/language/python/
   b) This teaches you how to create a containerized Python application using Docker. Do all the steps of this tutorial except for the last step "Deploy your app". You will do this step in the Kubernetes section.
   c) Take screenshots of each step and paste them into a Microsoft Word document entitled Docker-K8s-Lab.doc under a Docker section
3) Kubernetes
   a) Install Kubernetes on your laptop. You can either use Minikube (https://minikube.sigs.k8s.io/docs/start/) or Docker Desktop (you will need to follow the instructions to enable Kubernetes).
   b) Go through the exercises in https://kubernetes.io/docs/tutorials/kubernetes-basics/
   c) Deploy the Flask app you built in the Docker section on your cluster. Your deployment should have 2 pods and your service can use ClusterIP or NodePort.
   d) Take screenshots of each step and paste them into a separate K8s section in the doc Docker-K8s-Lab.doc.
   e) Submit the document using file upload option in this assignment.

Docker

Containerize a Python application:

Prerequisites
You have installed the latest version of Docker Desktop.
You have a git client. The examples in this section use a command-line based git client, but you can use any client.

Overview
This section walks you through containerizing and running a Python application.

Get the sample application
The sample application uses the popular Flask framework.
Clone the sample application to use with this guide. Open a terminal, change directory to a directory that you want to work in, and run the following command to clone the repository:

```
$ git clone https://github.com/docker/python-docker
```

```
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/CC-Lab5$ git clone https://github.com/docker/python
-docker
Cloning into 'python-docker'...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 11 (delta 3), reused 1 (delta 0), pack-reused 3
Receiving objects: 100% (11/11), done.
Resolving deltas: 100% (3/3), done.
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/CC-Lab5$ ll
total 0
drwxrwxrwx 1 jay jay 4096 May 10 15:29 ./
drwxrwxrwx 1 jay jay 4096 May  9 16:25 ../
drwxrwxrwx 1 jay jay 4096 May 10 15:29 python-docker/
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/CC-Lab5$ ls .
python-docker
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/CC-Lab5$ ls ./python-docker/
README.md  app.py  requirements.txt
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/CC-Lab5$
```

Initialize Docker assets
Now that you have an application, you can use docker init to create the necessary Docker assets to containerize your application. Inside the python-docker directory, run the docker init command. docker init provides some default configuration, but you'll need to answer a few questions about your application. For example, this application uses Flask to run. Refer to the following example to answer the prompts from docker init and use the same answers for your prompts.

```
$ docker init
Welcome to the Docker Init CLI!
This utility will walk you through creating the following files with sensible defaults for your project:
  - .dockerignore
  - Dockerfile
  - compose.yaml
  - README.Docker.md
Let's get started!
? What application platform does your project use? Python
```

```
? What version of Python do you want to use? 3.11.4
? What port do you want your app to listen on? 5000
? What is the command to run your app? python3 -m flask run --host=0.0.0.0
```

You should now have the following contents in your python-docker directory.

```
├── python-docker/
│   ├── app.py
│   ├── requirements.txt
│   ├── .dockerignore
│   ├── compose.yaml
│   ├── Dockerfile
│   ├── README.Docker.md
│   └── README.md
```

To learn more about the files that docker init added, see the following:

```
Dockerfile
.dockerignore
compose.yaml
```

```
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/CC-Lab5/python-docker$ docker init

Welcome to the Docker Init CLI!

This utility will walk you through creating the following files with sensible defaults for your projec
t:
  - .dockerignore
  - Dockerfile
  - compose.yaml
  - README.Docker.md

Let's get started!

! Warning → The following Docker files already exist in this directory:
  - .dockerignore
  - Dockerfile
  - compose.yaml
  - README.Docker.md

? Do you want to overwrite them? Yes
? What application platform does your project use? Python
? What version of Python do you want to use? 3.10.12
? What port do you want your app to listen on? 52333
? What is the command you use to run your app? python3 -m flask run --host=0.0.0.0 --port=52333

✓Created → .dockerignore
✓Created → Dockerfile
✓Created → compose.yaml
✓Created → README.Docker.md

→ Your Docker files are ready!
  Review your Docker files and tailor them to your application.
  Consult README.Docker.md for information about using the generated files.

What's next?
  Start your application by running → docker compose up --build
  Your application will be available at http://localhost:52333
```

Command = python3 —m flask run -—host=0.0.0.0 -—port=52333

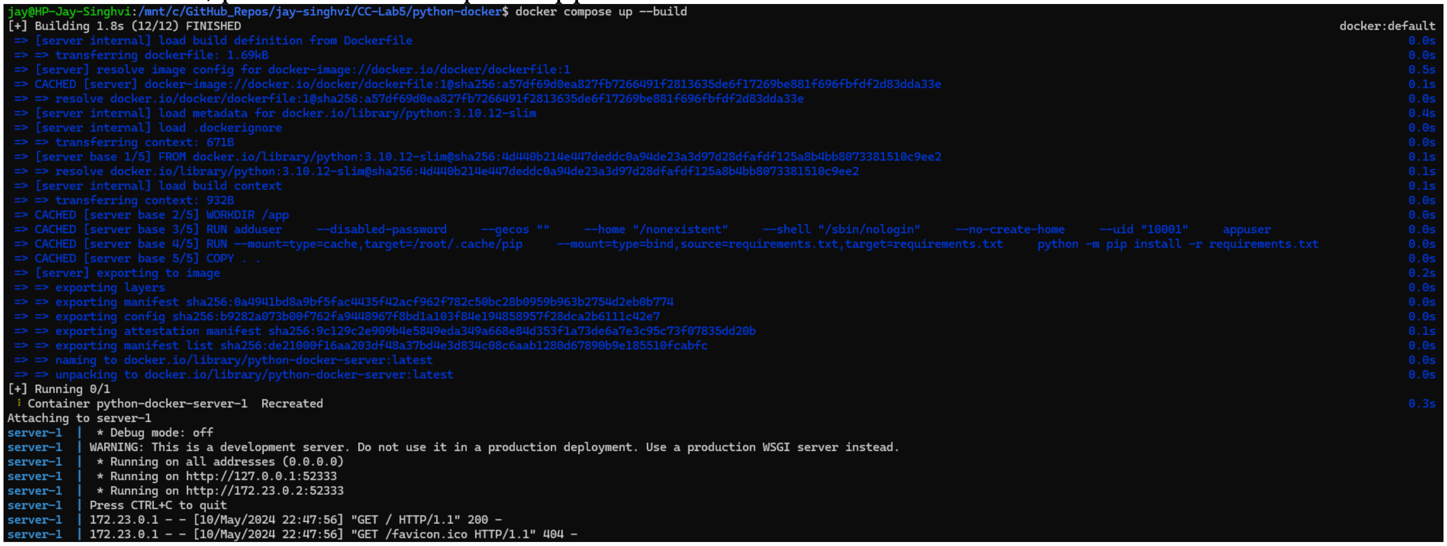
Run the application

Inside the python-docker directory, run the following command in a terminal.

```
$ docker compose up --build
```

Open a browser and view the application at http://localhost:5000. You should see a simple Flask application.

In the terminal, press ctrl+c to stop the application.

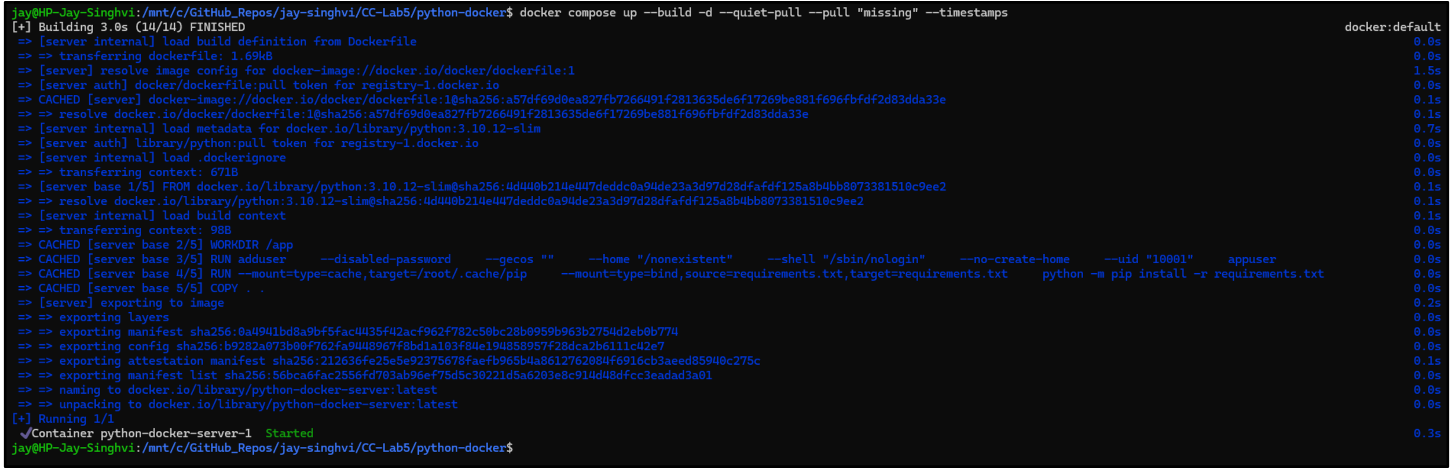


```
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/CC-Lab5/python-docker$ docker compose up --build
[+] Building 1.8s (12/12) FINISHED                                                                        docker:default
 => [server internal] load build definition from Dockerfile                                                          0.0s
 => => transferring dockerfile: 1.69kB                                                                               0.0s
 => [server] resolve image config for docker-image://docker.io/docker/dockerfile:1                                   0.5s
 => CACHED [server] docker-image://docker.io/docker/dockerfile:1@sha256:a57df69d0ea827fb7266491f2813635de6f17269be881f696fbfdf2d83dda33e    0.1s
 => => resolve docker.io/docker/dockerfile:1@sha256:a57df69d0ea827fb7266491f2813635de6f17269be881f696fbfdf2d83dda33e   0.0s
 => [server internal] load metadata for docker.io/library/python:3.10.12-slim                                        0.4s
 => [server internal] load .dockerignore                                                                             0.0s
 => => transferring context: 671B                                                                                    0.0s
 => [server base 1/5] FROM docker.io/library/python:3.10.12-slim@sha256:4d440b214e447deddc0a94de23a3d97d28dfafdf125a8b4bb8073381510c9ee2    0.1s
 => => resolve docker.io/library/python:3.10.12-slim@sha256:4d440b214e447deddc0a94de23a3d97d28dfafdf125a8b4bb8073381510c9ee2   0.1s
 => [server internal] load build context                                                                             0.1s
 => => transferring context: 932B                                                                                    0.0s
 => CACHED [server base 2/5] WORKDIR /app                                                                            0.0s
 => CACHED [server base 3/5] RUN adduser     --disabled-password     --gecos ""     --home "/nonexistent"     --shell "/sbin/nologin"     --no-create-home     --uid "10001"     appuser    0.0s
 => CACHED [server base 4/5] RUN --mount=type=cache,target=/root/.cache/pip     --mount=type=bind,source=requirements.txt,target=requirements.txt     python -m pip install -r requirements.txt    0.0s
 => CACHED [server base 5/5] COPY . .                                                                                0.0s
 => [server] exporting to image                                                                                      0.2s
 => => exporting layers                                                                                              0.0s
 => => exporting manifest sha256:0a4941bd8a9bf5fac4435f42acf962f782c50bc28b0959b963b2754d2eb0b774                    0.0s
 => => exporting config sha256:b9282a073b00f762fa9448967f8bd1a103f84e194858957f28dca2b6111c42e7                      0.0s
 => => exporting attestation manifest sha256:9c129c2e909b4e5849eda349a668e84d353f1a73de6a7e3c95c73f07835dd20b        0.1s
 => => exporting manifest list sha256:de21000f16aa203df48a37bd4e3d834c08c6aab1280d67890b9e185510fcabfc               0.0s
 => => naming to docker.io/library/python-docker-server:latest                                                       0.0s
 => => unpacking to docker.io/library/python-docker-server:latest                                                    0.0s
[+] Running 0/1
 ⁝ Container python-docker-server-1  Recreated                                                                       0.3s
Attaching to server-1
server-1  |  * Debug mode: off
server-1  | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
server-1  |  * Running on all addresses (0.0.0.0)
server-1  |  * Running on http://127.0.0.1:52333
server-1  |  * Running on http://172.23.0.2:52333
server-1  | Press CTRL+C to quit
server-1  | 172.23.0.1 - - [10/May/2024 22:47:56] "GET / HTTP/1.1" 200 -
server-1  | 172.23.0.1 - - [10/May/2024 22:47:56] "GET /favicon.ico HTTP/1.1" 404 -
```

Run the application in the background

You can run the application detached from the terminal by adding the -d option. Inside the python-docker directory, run the following command in a terminal.

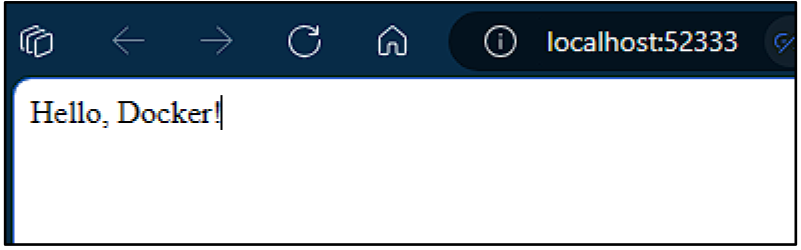
```
$ docker compose up --build -d
```



```
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/CC-Lab5/python-docker$ docker compose up --build -d --quiet-pull --pull "missing" --timestamps
[+] Building 3.0s (14/14) FINISHED                                                                        docker:default
 => [server internal] load build definition from Dockerfile                                                          0.0s
 => => transferring dockerfile: 1.69kB                                                                               0.0s
 => [server] resolve image config for docker-image://docker.io/docker/dockerfile:1                                   1.5s
 => [server auth] docker/dockerfile:pull token for registry-1.docker.io                                              0.0s
 => CACHED [server] docker-image://docker.io/docker/dockerfile:1@sha256:a57df69d0ea827fb7266491f2813635de6f17269be881f696fbfdf2d83dda33e    0.1s
 => => resolve docker.io/docker/dockerfile:1@sha256:a57df69d0ea827fb7266491f2813635de6f17269be881f696fbfdf2d83dda33e   0.1s
 => [server internal] load metadata for docker.io/library/python:3.10.12-slim                                        0.7s
 => [server auth] library/python:pull token for registry-1.docker.io                                                 0.0s
 => [server internal] load .dockerignore                                                                             0.0s
 => => transferring context: 671B                                                                                    0.0s
 => [server base 1/5] FROM docker.io/library/python:3.10.12-slim@sha256:4d440b214e447deddc0a94de23a3d97d28dfafdf125a8b4bb8073381510c9ee2    0.1s
 => => resolve docker.io/library/python:3.10.12-slim@sha256:4d440b214e447deddc0a94de23a3d97d28dfafdf125a8b4bb8073381510c9ee2   0.1s
 => [server internal] load build context                                                                             0.1s
 => => transferring context: 98B                                                                                     0.0s
 => CACHED [server base 2/5] WORKDIR /app                                                                            0.0s
 => CACHED [server base 3/5] RUN adduser     --disabled-password     --gecos ""     --home "/nonexistent"     --shell "/sbin/nologin"     --no-create-home     --uid "10001"     appuser    0.0s
 => CACHED [server base 4/5] RUN --mount=type=cache,target=/root/.cache/pip     --mount=type=bind,source=requirements.txt,target=requirements.txt     python -m pip install -r requirements.txt    0.0s
 => CACHED [server base 5/5] COPY . .                                                                                0.0s
 => [server] exporting to image                                                                                      0.2s
 => => exporting layers                                                                                              0.0s
 => => exporting manifest sha256:0a4941bd8a9bf5fac4435f42acf962f782c50bc28b0959b963b2754d2eb0b774                    0.0s
 => => exporting config sha256:b9282a073b00f762fa9448967f8bd1a103f84e194858957f28dca2b6111c42e7                      0.0s
 => => exporting attestation manifest sha256:212636fe25e5e92375678faefb965b4a8612762084f6916cb3aeed85940c275c        0.1s
 => => exporting manifest list sha256:56bca6fac2556fd703ab96ef75d5c30221d5a6203e8c914d48dfcc3eadad3a01               0.0s
 => => naming to docker.io/library/python-docker-server:latest                                                       0.0s
 => => unpacking to docker.io/library/python-docker-server:latest                                                    0.0s
[+] Running 1/1
 ✔Container python-docker-server-1  Started                                                                          0.3s
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/CC-Lab5/python-docker$
```

Open a browser and view the application at http://localhost:5000.

You should see a simple Flask application.



Hello, Docker!

In the terminal, run the following command to stop the application.

```
$ docker compose down
```

For more information about Compose commands, see the Compose CLI reference.

```
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/CC-Lab5/python-docker$ docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED              STATUS              PORTS                      NAMES
2b5b62e4f2d6   python-docker-server   "/bin/sh -c 'python3…"   About a minute ago   Up About a minute   0.0.0.0:52333->52333/tcp   python-docker-server-1
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/CC-Lab5/python-docker$ docker compose down
[+] Running 2/2
 ✔Container python-docker-server-1  Removed
 ✔Network python-docker_default     Removed
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/CC-Lab5/python-docker$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/CC-Lab5/python-docker$
```

Use containers for Python development

Prerequisites

Complete Containerize a Python application.

Overview

In this section, you'll learn how to set up a development environment for your containerized application. This includes:
Adding a local database and persisting data
Configuring Compose to automatically update your running Compose services as you edit and save your code

Get the sample application

You'll need to clone a new repository to get a sample application that includes logic to connect to the database.

Change to a directory where you want to clone the repository and run the following command.

```
git clone https://github.com/docker/python-docker-dev
```

```
jay@HP-Jay-Singhvi: /mnt/c/(

jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/CC-Lab5$ git clone https://github.com/docker/python-docker-dev
Cloning into 'python-docker-dev'...
remote: Enumerating objects: 18, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 18 (delta 2), reused 1 (delta 1), pack-reused 11
Receiving objects: 100% (18/18), 6.03 KiB | 441.00 KiB/s, done.
Resolving deltas: 100% (4/4), done.
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/CC-Lab5$ ls ./python-docker-dev
README.md  app.py  requirements.txt
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/CC-Lab5$
```

In the cloned repository's directory, run docker init to create the necessary Docker files. Refer to the following example to answer the prompts from docker init.

```
docker init
Welcome to the Docker Init CLI!

This utility will walk you through creating the following files with sensible defaults for your project:
  - .dockerignore
  - Dockerfile
  - compose.yaml
  - README.Docker.md
```

```
Let's get started!

? What application platform does your project use? Python
? What version of Python do you want to use? 3.11.4
? What port do you want your app to listen on? 5000
? What is the command to run your app? python3 -m flask run --host=0.0.0.0
```

```
jay@HP-Jay-Singhvi: /mnt/c/(   Administrator: PowerShell

jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/CC-Lab5/python-docker-dev$ docker init

Welcome to the Docker Init CLI!

This utility will walk you through creating the following files with sensible defaults for your project:
  - .dockerignore
  - Dockerfile
  - compose.yaml
  - README.Docker.md

Let's get started!

? What application platform does your project use? Python
? What version of Python do you want to use? 3.10.12
? What port do you want your app to listen on? 5000
? What is the command you use to run your app? python -m flask run --host=0.0.0.0 --port=5000

✓ Created → .dockerignore
✓ Created → Dockerfile
✓ Created → compose.yaml
✓ Created → README.Docker.md

→ Your Docker files are ready!
  Review your Docker files and tailor them to your application.
  Consult README.Docker.md for information about using the generated files.

What's next?
  Start your application by running → docker compose up --build
  Your application will be available at http://localhost:5000
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/CC-Lab5/python-docker-dev$
```

Add a local database and persist data

You can use containers to set up local services, like a database. In this section, you'll update the compose.yaml file to define a database service and a volume to persist data.

In the cloned repository's directory, open the compose.yaml file in an IDE or text editor. docker init handled creating most of the instructions, but you'll need to update it for your unique application.

In the compose.yaml file, you need to uncomment all of the database instructions. In addition, you need to add the database password file as an environment variable to the server service and specify the secret file to use .

The following is the updated compose.yaml file.

```
services:
 server:
  build:
   context: .
  ports:
   - 5000:5000
  environment:
   - POSTGRES_PASSWORD_FILE=/run/secrets/db-password
  depends_on:
```

```
      db:
        condition: service_healthy
    secrets:
      - db-password
  db:
    image: postgres
    restart: always
    user: postgres
    secrets:
      - db-password
    volumes:
      - db-data:/var/lib/postgresql/data
    environment:
      - POSTGRES_DB=example
      - POSTGRES_PASSWORD_FILE=/run/secrets/db-password
    expose:
      - 5432
    healthcheck:
      test: [ "CMD", "pg_isready" ]
      interval: 10s
      timeout: 5s
      retries: 5
volumes:
  db-data:
secrets:
  db-password:
    file: db/password.txt
```

compose.yaml

python-docker-dev > compose.yaml > {} secrets > {} db-password > file

docker-compose.yml - The Compose specification establishes a standard for the definition of mu

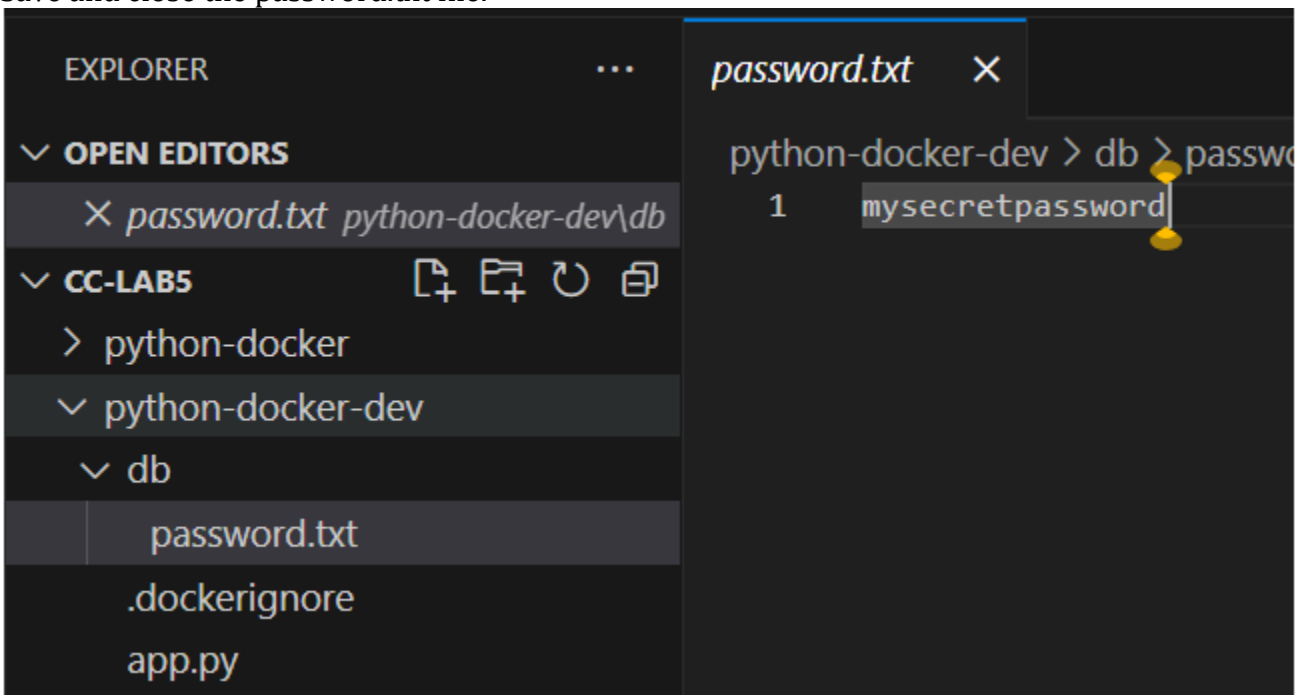
```yaml
services:
  server:
    build:
      context: .
    ports:
      - 5000:5000
    environment:
      - POSTGRES_PASSWORD_FILE=/run/secrets/db-password
    depends_on:
      db:
        condition: service_healthy
    secrets:
      - db-password
  db:
    image: postgres
    restart: always
    user: postgres
    secrets:
      - db-password
    volumes:
      - db-data:/var/lib/postgresql/data
    environment:
      - POSTGRES_DB=example
      - POSTGRES_PASSWORD_FILE=/run/secrets/db-password
    expose:
      - 5432
    healthcheck:
      test: [ "CMD", "pg_isready" ]
      interval: 10s
      timeout: 5s
      retries: 5
volumes:
  db-data:
secrets:
  db-password:
    file: db/password.txt
```

Before you run the application using Compose, notice that this Compose file specifies a password.txt file to hold the database's password. You must create this file as it's not included in the source repository.

In the cloned repository's directory, create a new directory named db and inside that directory create a file named password.txt that contains the password for the database. Using your favorite IDE or text editor, add the following contents to the password.txt file.

```
mysecretpassword
```

Save and close the password.txt file.

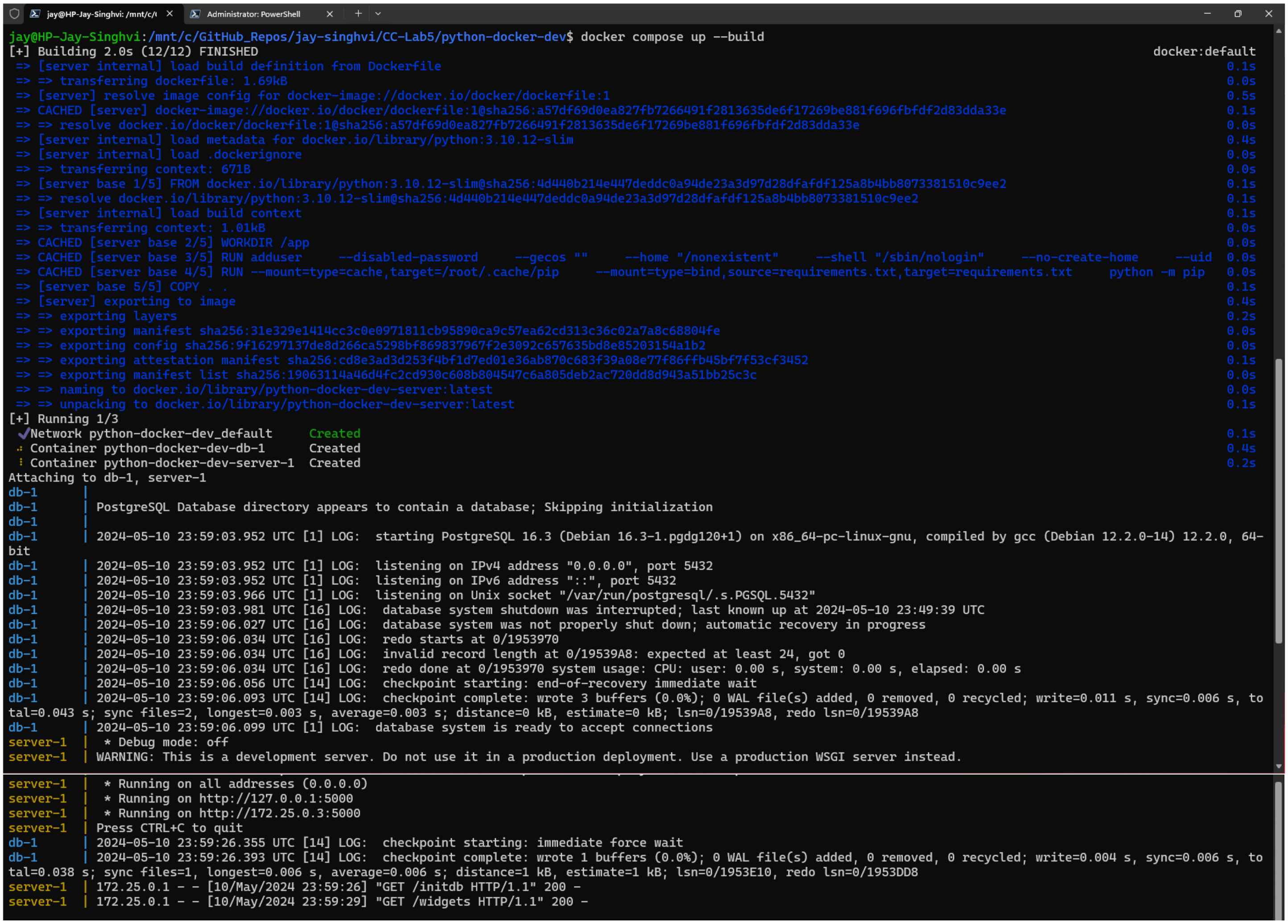


You should now have the following contents in your python-docker-dev directory.

```
├── python-docker-dev/
│   ├── db/
│   │   └── password.txt
│   ├── app.py
│   ├── requirements.txt
│   ├── .dockerignore
│   ├── compose.yaml
│   ├── Dockerfile
│   ├── README.Docker.md
│   └── README.md
```

Now, run the following docker compose up command to start your application.

```
docker compose up --build
```



```
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/CC-Lab5/python-docker-dev$ docker compose up --build
[+] Building 2.0s (12/12) FINISHED                                                                                                                          docker:default
 => [server internal] load build definition from Dockerfile                                                                                                           0.1s
 => => transferring dockerfile: 1.69kB                                                                                                                                0.0s
 => [server] resolve image config for docker-image://docker.io/docker/dockerfile:1                                                                                    0.5s
 => CACHED [server] docker-image://docker.io/docker/dockerfile:1@sha256:a57df69d0ea827fb7266491f2813635de6f17269be881f696fbfdf2d83dda33e                              0.1s
 => => resolve docker.io/docker/dockerfile:1@sha256:a57df69d0ea827fb7266491f2813635de6f17269be881f696fbfdf2d83dda33e                                                  0.0s
 => [server internal] load metadata for docker.io/library/python:3.10.12-slim                                                                                         0.4s
 => [server internal] load .dockerignore                                                                                                                              0.0s
 => => transferring context: 671B                                                                                                                                     0.0s
 => [server base 1/5] FROM docker.io/library/python:3.10.12-slim@sha256:4d440b214e447deddc0a94de23a3d97d28dfafdf125a8b4bb8073381510c9ee2                              0.1s
 => => resolve docker.io/library/python:3.10.12-slim@sha256:4d440b214e447deddc0a94de23a3d97d28dfafdf125a8b4bb8073381510c9ee2                                          0.1s
 => [server internal] load build context                                                                                                                              0.1s
 => => transferring context: 1.01kB                                                                                                                                   0.0s
 => CACHED [server base 2/5] WORKDIR /app                                                                                                                             0.0s
 => CACHED [server base 3/5] RUN adduser     --disabled-password     --gecos ""     --home "/nonexistent"     --shell "/sbin/nologin"     --no-create-home     --uid  0.0s
 => CACHED [server base 4/5] RUN --mount=type=cache,target=/root/.cache/pip     --mount=type=bind,source=requirements.txt,target=requirements.txt     python -m pip   0.0s
 => [server base 5/5] COPY . .                                                                                                                                        0.1s
 => [server] exporting to image                                                                                                                                       0.4s
 => => exporting layers                                                                                                                                               0.2s
 => => exporting manifest sha256:31e329e1414cc3c0e0971811cb95890ca9c57ea62cd313c36c02a7a8c68804fe                                                                     0.0s
 => => exporting config sha256:9f16297137de8d266ca5298bf869837967f2e3092c657635bd8e85203154a1b2                                                                       0.0s
 => => exporting attestation manifest sha256:cd8e3ad3d253f4bf1d7ed01e36ab870c683f39a08e77f86ffb45bf7f53cf3452                                                         0.1s
 => => exporting manifest list sha256:19063114a46d4fc2cd930c608b804547c6a805deb2ac720dd8d943a51bb25c3c                                                                0.0s
 => => naming to docker.io/library/python-docker-dev-server:latest                                                                                                    0.0s
 => => unpacking to docker.io/library/python-docker-dev-server:latest                                                                                                 0.1s
[+] Running 1/3
 ✔Network python-docker-dev_default     Created                                                                                                                       0.1s
 ⠴ Container python-docker-dev-db-1      Created                                                                                                                      0.4s
 ⠇ Container python-docker-dev-server-1  Created                                                                                                                      0.2s
Attaching to db-1, server-1
db-1      |
db-1      | PostgreSQL Database directory appears to contain a database; Skipping initialization
db-1      |
db-1      | 2024-05-10 23:59:03.952 UTC [1] LOG:  starting PostgreSQL 16.3 (Debian 16.3-1.pgdg120+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 12.2.0-14) 12.2.0, 64-
bit
db-1      | 2024-05-10 23:59:03.952 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5432
db-1      | 2024-05-10 23:59:03.952 UTC [1] LOG:  listening on IPv6 address "::", port 5432
db-1      | 2024-05-10 23:59:03.966 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
db-1      | 2024-05-10 23:59:03.981 UTC [16] LOG:  database system shutdown was interrupted; last known up at 2024-05-10 23:49:39 UTC
db-1      | 2024-05-10 23:59:06.027 UTC [16] LOG:  database system was not properly shut down; automatic recovery in progress
db-1      | 2024-05-10 23:59:06.034 UTC [16] LOG:  redo starts at 0/1953970
db-1      | 2024-05-10 23:59:06.034 UTC [16] LOG:  invalid record length at 0/19539A8: expected at least 24, got 0
db-1      | 2024-05-10 23:59:06.034 UTC [16] LOG:  redo done at 0/1953970 system usage: CPU: user: 0.00 s, system: 0.00 s, elapsed: 0.00 s
db-1      | 2024-05-10 23:59:06.056 UTC [14] LOG:  checkpoint starting: end-of-recovery immediate wait
db-1      | 2024-05-10 23:59:06.093 UTC [14] LOG:  checkpoint complete: wrote 3 buffers (0.0%); 0 WAL file(s) added, 0 removed, 0 recycled; write=0.011 s, sync=0.006 s, to
tal=0.043 s; sync files=2, longest=0.003 s, average=0.003 s; distance=0 kB, estimate=0 kB; lsn=0/19539A8, redo lsn=0/19539A8
db-1      | 2024-05-10 23:59:06.099 UTC [1] LOG:  database system is ready to accept connections
server-1  |  * Debug mode: off
server-1  | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
server-1  |  * Running on all addresses (0.0.0.0)
server-1  |  * Running on http://127.0.0.1:5000
server-1  |  * Running on http://172.25.0.3:5000
server-1  | Press CTRL+C to quit
db-1      | 2024-05-10 23:59:26.355 UTC [14] LOG:  checkpoint starting: immediate force wait
db-1      | 2024-05-10 23:59:26.393 UTC [14] LOG:  checkpoint complete: wrote 1 buffers (0.0%); 0 WAL file(s) added, 0 removed, 0 recycled; write=0.004 s, sync=0.006 s, to
tal=0.038 s; sync files=1, longest=0.006 s, average=0.006 s; distance=1 kB, estimate=1 kB; lsn=0/1953E10, redo lsn=0/1953DD8
server-1  | 172.25.0.1 - - [10/May/2024 23:59:26] "GET /initdb HTTP/1.1" 200 -
server-1  | 172.25.0.1 - - [10/May/2024 23:59:29] "GET /widgets HTTP/1.1" 200 -
```

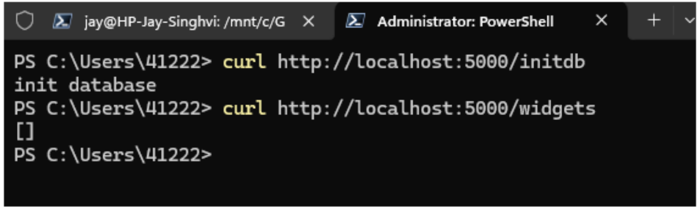
Now test your API endpoint. Open a new terminal then make a request to the server using the curl commands:

```
curl http://localhost:5000/initdb
 curl http://localhost:5000/widgets
```

You should receive the following response:

```
[]
```

The response is empty because your database is empty.



```
PS C:\Users\41222> curl http://localhost:5000/initdb
init database
PS C:\Users\41222> curl http://localhost:5000/widgets
[]
PS C:\Users\41222>
```

Press ctrl+c in the terminal to stop your application.

Automatically update services

Use Compose Watch to automatically update your running Compose services as you edit and save your code. For more details about Compose Watch, see Use Compose Watch.

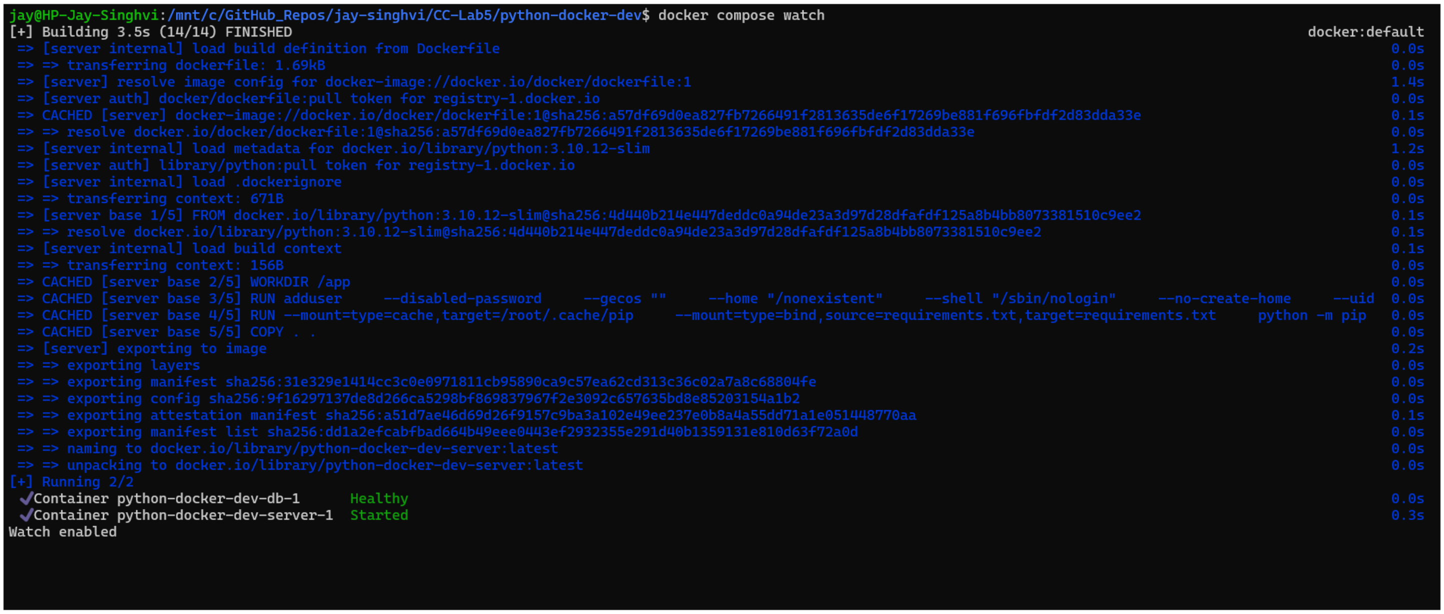
Open your compose.yaml file in an IDE or text editor and then add the Compose Watch instructions. The following is the updated compose.yaml file.

```
services:
```

```
  server:
    build:
      context: .
    ports:
      - 5000:5000
    environment:
      - POSTGRES_PASSWORD_FILE=/run/secrets/db-password
    depends_on:
      db:
        condition: service_healthy
    secrets:
      - db-password
    develop:
      watch:
        - action: rebuild
          path: .
  db:
    image: postgres
    restart: always
    user: postgres
    secrets:
      - db-password
    volumes:
      - db-data:/var/lib/postgresql/data
    environment:
      - POSTGRES_DB=example
      - POSTGRES_PASSWORD_FILE=/run/secrets/db-password
    expose:
      - 5432
    healthcheck:
      test: [ "CMD", "pg_isready" ]
      interval: 10s
      timeout: 5s
      retries: 5
volumes:
  db-data:
secrets:
  db-password:
    file: db/password.txt
```

Run the following command to run your application with Compose Watch.

```
docker compose watch
```



```
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/CC-Lab5/python-docker-dev$ docker compose watch
[+] Building 3.5s (14/14) FINISHED                                                                                                     docker:default
 => [server internal] load build definition from Dockerfile                                                                                       0.0s
 => => transferring dockerfile: 1.69kB                                                                                                            0.0s
 => [server] resolve image config for docker-image://docker.io/docker/dockerfile:1                                                                1.4s
 => [server auth] docker/dockerfile:pull token for registry-1.docker.io                                                                           0.0s
 => CACHED [server] docker-image://docker.io/docker/dockerfile:1@sha256:a57df69d0ea827fb7266491f2813635de6f17269be881f696fbfdf2d83dda33e           0.1s
 => => resolve docker.io/docker/dockerfile:1@sha256:a57df69d0ea827fb7266491f2813635de6f17269be881f696fbfdf2d83dda33e                              0.0s
 => [server internal] load metadata for docker.io/library/python:3.10.12-slim                                                                     1.2s
 => [server auth] library/python:pull token for registry-1.docker.io                                                                              0.0s
 => [server internal] load .dockerignore                                                                                                          0.0s
 => => transferring context: 671B                                                                                                                 0.0s
 => [server base 1/5] FROM docker.io/library/python:3.10.12-slim@sha256:4d440b214e447deddc0a94de23a3d97d28dfafdf125a8b4bb8073381510c9ee2           0.1s
 => => resolve docker.io/library/python:3.10.12-slim@sha256:4d440b214e447deddc0a94de23a3d97d28dfafdf125a8b4bb8073381510c9ee2                      0.1s
 => [server internal] load build context                                                                                                          0.1s
 => => transferring context: 156B                                                                                                                 0.0s
 => CACHED [server base 2/5] WORKDIR /app                                                                                                         0.0s
 => CACHED [server base 3/5] RUN adduser     --disabled-password     --gecos ""     --home "/nonexistent"     --shell "/sbin/nologin"     --no-create-home     --uid  0.0s
 => CACHED [server base 4/5] RUN --mount=type=cache,target=/root/.cache/pip     --mount=type=bind,source=requirements.txt,target=requirements.txt     python -m pip   0.0s
 => CACHED [server base 5/5] COPY . .                                                                                                             0.0s
 => [server] exporting to image                                                                                                                   0.2s
 => => exporting layers                                                                                                                           0.0s
 => => exporting manifest sha256:31e329e1414cc3c0e0971811cb95890ca9c57ea62cd313c36c02a7a8c68804fe                                                 0.0s
 => => exporting config sha256:9f16297137de8d266ca5298bf869837967f2e3092c657635bd8e85203154a1b2                                                   0.0s
 => => exporting attestation manifest sha256:a51d7ae46d69d26f9157c9ba3a102e49ee237e0b8a4a55dd71a1e051448770aa                                     0.1s
 => => exporting manifest list sha256:dd1a2efcabfbad664b49eee0443ef2932355e291d40b1359131e810d63f72a0d                                            0.0s
 => => naming to docker.io/library/python-docker-dev-server:latest                                                                                0.0s
 => => unpacking to docker.io/library/python-docker-dev-server:latest                                                                             0.0s
[+] Running 2/2
 ✓Container python-docker-dev-db-1      Healthy                                                                                                   0.0s
 ✓Container python-docker-dev-server-1  Started                                                                                                   0.3s
Watch enabled
```

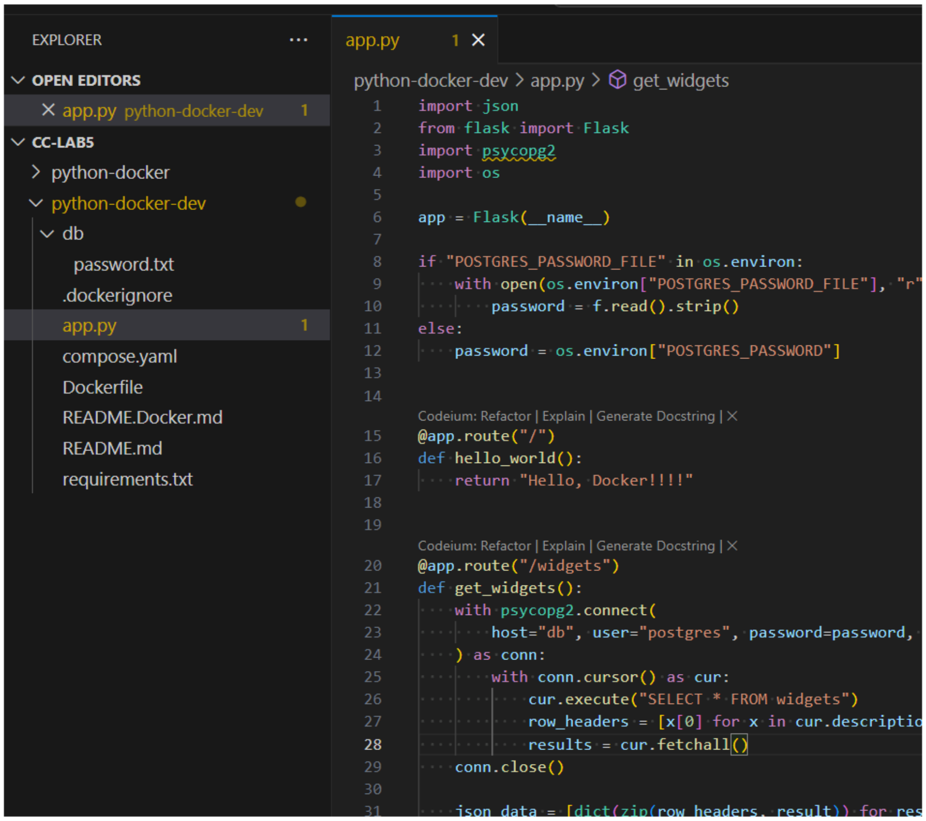
In a terminal, curl the application to get a response.

```
curl http://localhost:5000
Hello, Docker!
```

Any changes to the application's source files on your local machine will now be immediately reflected in the running container.

Open python-docker-dev/app.py in an IDE or text editor and update the Hello, Docker! string by adding a few more exclamation marks.

```
-   return 'Hello, Docker!'
+   return 'Hello, Docker!!!'
```



```
import json
from flask import Flask
import psycopg2
import os

app = Flask(__name__)

if "POSTGRES_PASSWORD_FILE" in os.environ:
    with open(os.environ["POSTGRES_PASSWORD_FILE"], "r"
        password = f.read().strip()
else:
    password = os.environ["POSTGRES_PASSWORD"]


Codeium: Refactor | Explain | Generate Docstring | ×
@app.route("/")
def hello_world():
    return "Hello, Docker!!!!"


Codeium: Refactor | Explain | Generate Docstring | ×
@app.route("/widgets")
def get_widgets():
    with psycopg2.connect(
        host="db", user="postgres", password=password,
    ) as conn:
        with conn.cursor() as cur:
            cur.execute("SELECT * FROM widgets")
            row_headers = [x[0] for x in cur.descriptio
            results = cur.fetchall()
    conn.close()

    json_data = [dict(zip(row_headers, result)) for res
```

Save the changes to app.py and then wait a few seconds for the application to rebuild. Curl the application again and verify that the updated text appears.

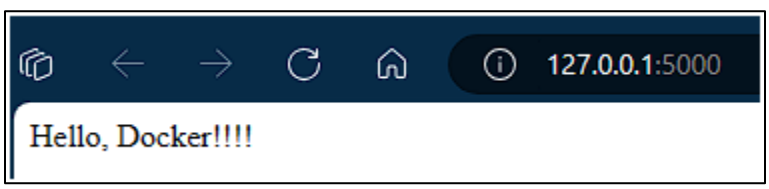
```
curl http://localhost:5000
Hello, Docker!!!
```

```
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/python-docker-and-kubernetes$ docker compose up --build
 --watch
[+] Building 9.0s (14/14) FINISHED
                                                                                                  docker:default
 => [server internal] load build definition from Dockerfile
                                                                                                            0.1s
 => => transferring dockerfile: 1.74kB
                                                                                                            0.1s
 => [server] resolve image config for docker-image://docker.io/docker/dockerfile:1
                                                                                                            1.2s
 => [server auth] docker/dockerfile:pull token for registry-1.docker.io
                                                                                                            0.0s
 => CACHED [server] docker-image://docker.io/docker/dockerfile:1@sha256:a57df69d0ea827fb7266491f2813635de6
f17269be881f696fbfdf2d83dda33e                                                                              0.1s
 => => resolve docker.io/docker/dockerfile:1@sha256:a57df69d0ea827fb7266491f2813635de6f17269be881f696fbfdf
2d83dda33e                                                                                                  0.1s
 => [server internal] load metadata for docker.io/library/python:3.10.12-slim
                                                                                                            0.7s
 => [server auth] library/python:pull token for registry-1.docker.io
                                                                                                            0.0s
 => [server internal] load .dockerignore
                                                                                                            0.1s
 => => transferring context: 705B
                                                                                                            0.0s
 => [server base 1/5] FROM docker.io/library/python:3.10.12-slim@sha256:4d440b214e447deddc0a94de23a3d97d28
dfafdf125a8b4bb8073381510c9ee2                                                                              0.1s
 => => resolve docker.io/library/python:3.10.12-slim@sha256:4d440b214e447deddc0a94de23a3d97d28dfafdf125a8b
4bb8073381510c9ee2                                                                                          0.1s
 => [server internal] load build context
                                                                                                            0.1s
 => => transferring context: 5.85kB
                                                                                                            0.1s
 => CACHED [server base 2/5] WORKDIR /app
                                                                                                            0.0s
 => CACHED [server base 3/5] RUN adduser     --disabled-password     --gecos ""     --home "/nonexistent"
    --shell "/sbin/nologin"     --no-create-home     --uid "10001"     appuser                            0.0s
 => [server base 4/5] RUN --mount=type=cache,target=/root/.cache/pip     --mount=type=bind,source=requirem
ents.txt,target=requirements.txt     python -m pip install -r requirements.txt                              4.4s
 => [server base 5/5] COPY . .
                                                                                                            0.1s
 => [server] exporting to image
                                                                                                            1.5s
 => => exporting layers
                                                                                                            0.8s
 => => exporting manifest sha256:7e0e4a3b7f1a945c4bf53585fa6d0b8121bce8fb242ecf6f5c80b7f5d7ef5f28
                                                                                                            0.0s
 => => exporting config sha256:a4a8ed5d0de6101e6f86dd82d126434f94d38c21495ec4eb0cf5b35b4eb695f1
                                                                                                            0.0s
 => => exporting attestation manifest sha256:6436355d7a87c451deec35814552a395e11998c12deee8c3b8546f57f9aea
a0b                                                                                                         0.1s
 => => exporting manifest list sha256:fc3419d82542651c987a00496cf31e39a174d91206c87aaa525a565f5cb209c8
                                                                                                            0.0s
 => => naming to docker.io/library/python-docker-and-kubernetes-server:latest
                                                                                                            0.0s
 => => unpacking to docker.io/library/python-docker-and-kubernetes-server:latest
                                                                                                            0.4s
[+] Running 4/4
```

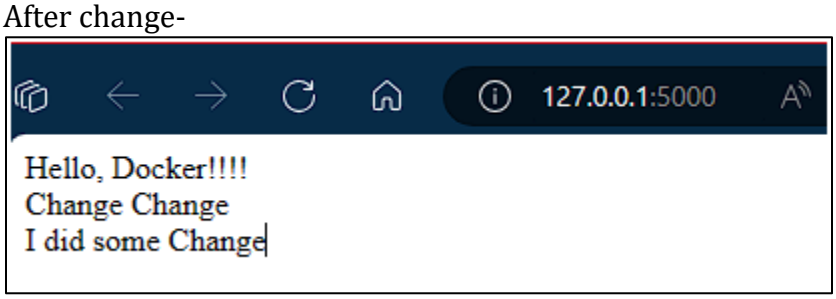
```
[+] Running 4/4
 ✔Network python-docker-and-kubernetes_default     Created
                                                                                                    0.1s
 ✔Volume "python-docker-and-kubernetes_db-data"    Created
                                                                                                    0.0s
 ✔Container python-docker-and-kubernetes-db-1      Created
                                                                                                    0.4s
 ✔Container python-docker-and-kubernetes-server-1  Created
                                                                                                    0.4s
        ⦿ Watch enabled
Attaching to db-1, server-1
db-1      | The files belonging to this database system will be owned by user "postgres".
db-1      | This user must also own the server process.
db-1      |
db-1      | The database cluster will be initialized with locale "en_US.utf8".
db-1      | The default database encoding has accordingly been set to "UTF8".
db-1      | The default text search configuration will be set to "english".
db-1      |
db-1      | Data page checksums are disabled.
db-1      |
db-1      | fixing permissions on existing directory /var/lib/postgresql/data ... ok
db-1      | creating subdirectories ... ok
db-1      | selecting dynamic shared memory implementation ... posix
db-1      | selecting default max_connections ... 100
db-1      | selecting default shared_buffers ... 128MB
db-1      | selecting default time zone ... Etc/UTC
db-1      | creating configuration files ... ok
db-1      | running bootstrap script ... ok
db-1      | performing post-bootstrap initialization ... ok
db-1      | syncing data to disk ... ok
db-1      |
db-1      |
db-1      | Success. You can now start the database server using:
db-1      |
db-1      |     pg_ctl -D /var/lib/postgresql/data -l logfile start
db-1      |
db-1      | initdb: warning: enabling "trust" authentication for local connections
db-1      | initdb: hint: You can change this by editing pg_hba.conf or using the option -A, or --auth-loc
al and --auth-host, the next time you run initdb.
db-1      | waiting for server to start....2024-05-11 05:26:13.365 UTC [35] LOG:  starting PostgreSQL 16.3
 (Debian 16.3-1.pgdg120+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 12.2.0-14) 12.2.0, 64-bit
db-1      | 2024-05-11 05:26:13.371 UTC [35] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.
5432"
db-1      | 2024-05-11 05:26:13.391 UTC [38] LOG:  database system was shut down at 2024-05-11 05:26:11 UT
C
db-1      | 2024-05-11 05:26:13.401 UTC [35] LOG:  database system is ready to accept connections
db-1      |  done
db-1      | server started
db-1      | CREATE DATABASE
db-1      |
db-1      |
db-1      | /usr/local/bin/docker-entrypoint.sh: ignoring /docker-entrypoint-initdb.d/*
db-1      |
db-1      | waiting for server to shut down...2024-05-11 05:26:13.595 UTC [35] LOG:  received fast shutdow
n request
db-1      | .2024-05-11 05:26:13.600 UTC [35] LOG:  aborting any active transactions
db-1      | 2024-05-11 05:26:13.602 UTC [35] LOG:  background worker "logical replication launcher" (PID 4
```

```
db-1      | .2024-05-11 05:26:13.600 UTC [35] LOG:  aborting any active transactions
db-1      | 2024-05-11 05:26:13.602 UTC [35] LOG:  background worker "logical replication launcher" (PID 4
1) exited with exit code 1
db-1      | 2024-05-11 05:26:13.602 UTC [36] LOG:  shutting down
db-1      | 2024-05-11 05:26:13.607 UTC [36] LOG:  checkpoint starting: shutdown immediate
db-1      | 2024-05-11 05:26:14.131 UTC [36] LOG:  checkpoint complete: wrote 922 buffers (5.6%); 0 WAL fi
le(s) added, 0 removed, 0 recycled; write=0.027 s, sync=0.479 s, total=0.529 s; sync files=301, longest=0.
016 s, average=0.002 s; distance=4255 kB, estimate=4255 kB; lsn=0/1912040, redo lsn=0/1912040
db-1      | 2024-05-11 05:26:14.138 UTC [35] LOG:  database system is shut down
db-1      |  done
db-1      | server stopped
db-1      |
db-1      | PostgreSQL init process complete; ready for start up.
db-1      |
db-1      | 2024-05-11 05:26:14.228 UTC [1] LOG:  starting PostgreSQL 16.3 (Debian 16.3-1.pgdg120+1) on x8
6_64-pc-linux-gnu, compiled by gcc (Debian 12.2.0-14) 12.2.0, 64-bit
db-1      | 2024-05-11 05:26:14.228 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5432
db-1      | 2024-05-11 05:26:14.229 UTC [1] LOG:  listening on IPv6 address "::", port 5432
db-1      | 2024-05-11 05:26:14.239 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5
432"
db-1      | 2024-05-11 05:26:14.254 UTC [51] LOG:  database system was shut down at 2024-05-11 05:26:14 UT
C
db-1      | 2024-05-11 05:26:14.263 UTC [1] LOG:  database system is ready to accept connections
server-1  |  * Debug mode: off
server-1  | WARNING: This is a development server. Do not use it in a production deployment. Use a product
ion WSGI server instead.
server-1  |  * Running on all addresses (0.0.0.0)
server-1  |  * Running on http://127.0.0.1:5000
server-1  |  * Running on http://172.18.0.3:5000
server-1  | Press CTRL+C to quit
server-1  | 172.18.0.1 - - [11/May/2024 05:26:29] "GET / HTTP/1.1" 200 -
server-1  | 172.18.0.1 - - [11/May/2024 05:26:30] "GET /favicon.ico HTTP/1.1" 404 -
          ⦿ Rebuilding service "server" after changes were detected...
          ⦿ service "server" successfully built
server-1  | 172.18.0.1 - - [11/May/2024 05:26:58] "GET / HTTP/1.1" 200 -
server-1  | 172.18.0.1 - - [11/May/2024 05:27:01] "GET / HTTP/1.1" 200 -
server-1  | 172.18.0.1 - - [11/May/2024 05:27:01] "GET / HTTP/1.1" 200 -
server-1 exited with code 137
server-1 has been recreated
          ⦿ Rebuilding service "server" after changes were detected...
server-1  |  * Debug mode: off
server-1  | WARNING: This is a development server. Do not use it in a production deployment. Use a product
ion WSGI server instead.
server-1  |  * Running on all addresses (0.0.0.0)
server-1  |  * Running on http://127.0.0.1:5000
server-1  |  * Running on http://172.18.0.3:5000
server-1  | Press CTRL+C to quit
          ⦿ service "server" successfully built
server-1  | 172.18.0.1 - - [11/May/2024 05:27:13] "GET / HTTP/1.1" 200 -
server-1  | 172.18.0.1 - - [11/May/2024 05:27:14] "GET /favicon.ico HTTP/1.1" 404 -
server-1 exited with code 137
server-1 has been recreated
server-1  |  * Debug mode: off
server-1  | WARNING: This is a development server. Do not use it in a production deployment. Use a product
ion WSGI server instead.
server-1  |  * Running on all addresses (0.0.0.0)
server-1  |  * Running on http://127.0.0.1:5000
```

```
ion WSGI server instead.
server-1  |  * Running on all addresses (0.0.0.0)
server-1  |  * Running on http://127.0.0.1:5000
server-1  |  * Running on http://172.18.0.3:5000
server-1  | Press CTRL+C to quit
          ⊙ Rebuilding service "server" after changes were detected...
          ⊙ service "server" successfully built
server-1  | 172.18.0.1 - - [11/May/2024 05:28:33] "GET / HTTP/1.1" 200 -
server-1  | 172.18.0.1 - - [11/May/2024 05:28:35] "GET / HTTP/1.1" 200 -
server-1  | 172.18.0.1 - - [11/May/2024 05:28:35] "GET / HTTP/1.1" 200 -
server-1  | 172.18.0.1 - - [11/May/2024 05:28:36] "GET / HTTP/1.1" 200 -
server-1  | 172.18.0.1 - - [11/May/2024 05:28:37] "GET / HTTP/1.1" 200 -
server-1  | 172.18.0.1 - - [11/May/2024 05:28:38] "GET / HTTP/1.1" 200 -
server-1  | 172.18.0.1 - - [11/May/2024 05:28:38] "GET / HTTP/1.1" 200 -
server-1  | 172.18.0.1 - - [11/May/2024 05:28:39] "GET / HTTP/1.1" 200 -
server-1  | 172.18.0.1 - - [11/May/2024 05:28:40] "GET / HTTP/1.1" 200 -
server-1  | 172.18.0.1 - - [11/May/2024 05:28:40] "GET / HTTP/1.1" 200 -
server-1  | 172.18.0.1 - - [11/May/2024 05:28:41] "GET / HTTP/1.1" 200 -
server-1 exited with code 137
server-1 has been recreated
          ⊙ Rebuilding service "server" after changes were detected...
server-1  |  * Debug mode: off
server-1  | WARNING: This is a development server. Do not use it in a production deployment. Use a product
ion WSGI server instead.
server-1  |  * Running on all addresses (0.0.0.0)
server-1  |  * Running on http://127.0.0.1:5000
server-1  |  * Running on http://172.18.0.3:5000
server-1  | Press CTRL+C to quit
server-1  | 172.18.0.1 - - [11/May/2024 05:28:45] "GET / HTTP/1.1" 200 -
          ⊙ service "server" successfully built
server-1 exited with code 137
server-1 has been recreated
server-1  |  * Debug mode: off
server-1  | WARNING: This is a development server. Do not use it in a production deployment. Use a product
ion WSGI server instead.
server-1  |  * Running on all addresses (0.0.0.0)
server-1  |  * Running on http://127.0.0.1:5000
server-1  |  * Running on http://172.18.0.3:5000
server-1  | Press CTRL+C to quit
db-1      | 2024-05-11 05:31:14.343 UTC [49] LOG:  checkpoint starting: time
db-1      | 2024-05-11 05:31:18.625 UTC [49] LOG:  checkpoint complete: wrote 45 buffers (0.3%); 0 WAL fil
e(s) added, 0 removed, 0 recycled; write=4.228 s, sync=0.027 s, total=4.282 s; sync files=12, longest=0.00
7 s, average=0.003 s; distance=260 kB, estimate=260 kB; lsn=0/1953460, redo lsn=0/1953428
```



After change-



Press ctrl+c in the terminal to stop your application.

Configure CI/CD for your Python application

Prerequisites

Complete all the previous sections of this guide, starting with Containerize a Python application. You must have a GitHub account and a Docker account to complete this section.

Overview

In this section, you'll learn how to set up and use GitHub Actions to build and test your Docker image as well as push it to Docker Hub. You will complete the following steps:

1. Create a new repository on GitHub.
2. Define the GitHub Actions workflow.
3. Run the workflow.

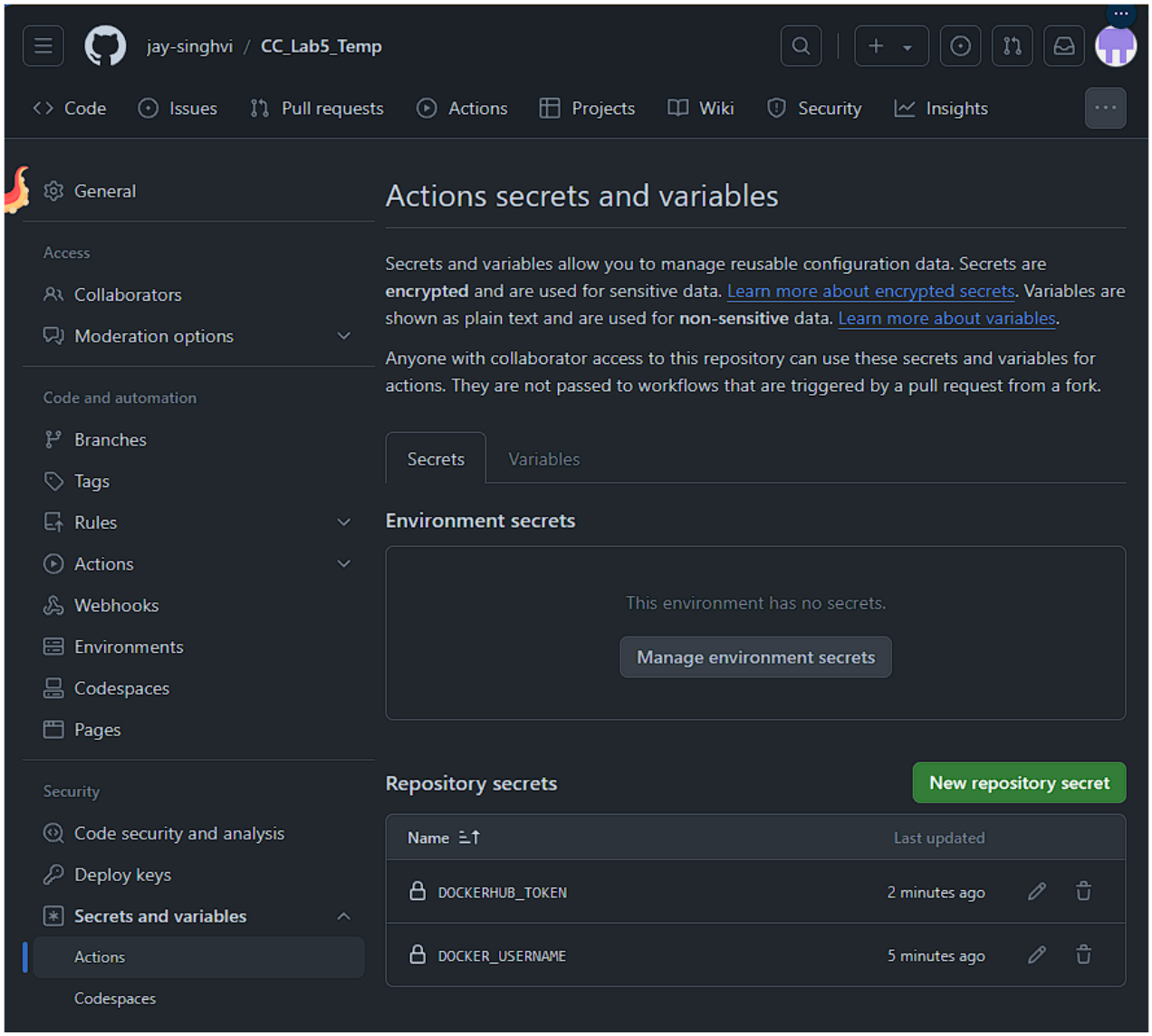
Step one: Create the repository

Create a GitHub repository, configure the Docker Hub secrets, and push your source code.

1. Create a new repository on GitHub.
2. Open the repository Settings, and go to Secrets and variables > Actions.
3. Create a new secret named DOCKER_USERNAME and your Docker ID as value.

4. Create a new Personal Access Token (PAT) for Docker Hub. You can name this token python-docker.

5. Add the PAT as a second secret in your GitHub repository, with the name DOCKERHUB_TOKEN.



6. In your local repository on your machine, run the following command to change the origin to the repository you just created. Make sure you change your-username to your GitHub username and your-repository to the name of the repository you created.

```
git remote set-url origin https://github.com/your-username/your-repository.git
```

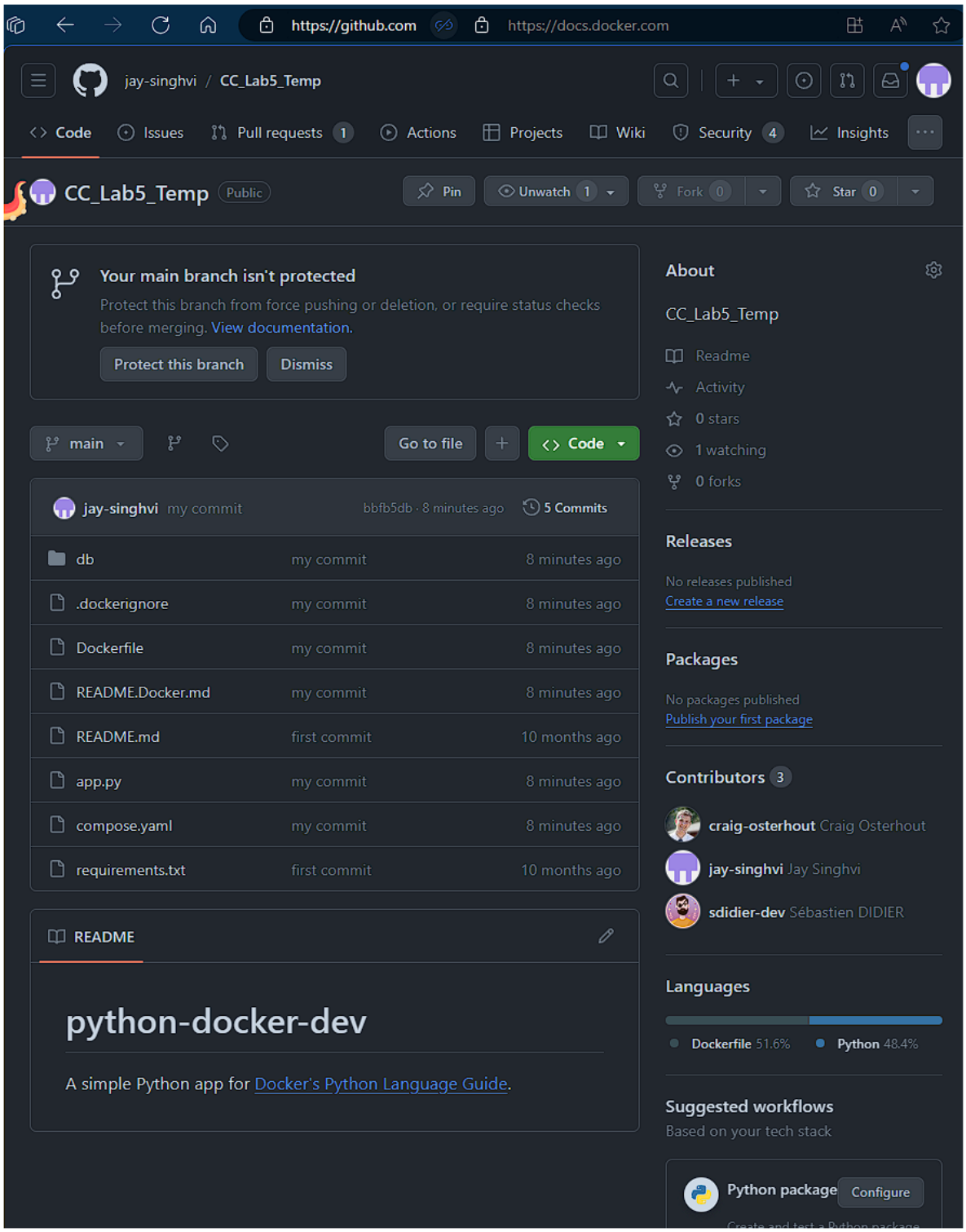
```
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/CC-Lab5/python-docker-dev$ git remote set-url origin https://github.com/jay-singhvi/CC_Lab5_Temp.git
```

7. Run the following commands to stage, commit, and push your local repository to GitHub.

```
git add -A
 git commit -m "my commit"
 git push -u origin main
```

Note: These command does not work and gave below error and later the same was completed using VS Code GUI

```
jay@HP-Jay-Singhvi: /mnt/c/(
^Ccanceled
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/CC-Lab5/python-docker-dev$ git remote set-url origin https://github.com/jay-singhvi/CC_Lab5_Temp.git
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/CC-Lab5/python-docker-dev$ git add -A
git commit -m "my commit"
git push -u origin main
Author identity unknown

*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: empty ident name (for <jay@HP-Jay-Singhvi.>) not allowed
Username for 'https://github.com': git config --global user.email "jsinghvi@seattleu.edu"
Password for 'https://git config --global user.email "jsinghvi@seattleu.edu"@github.com':
remote: Support for password authentication was removed on August 13, 2021.
remote: Please see https://docs.github.com/get-started/getting-started-with-git/about-remote-repositories#cloning-with-https-urls for information on currently recommended
modes of authentication.
fatal: Authentication failed for 'https://github.com/jay-singhvi/CC_Lab5_Temp.git/'
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/CC-Lab5/python-docker-dev$ git config --global user.email "jsinghvi@seattleu.edu"
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/CC-Lab5/python-docker-dev$ git config --global user.name "Jay Singhvi"
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/CC-Lab5/python-docker-dev$ git add -A
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/CC-Lab5/python-docker-dev$ git commit -m "my commit"
[main bbfb5db] my commit
 6 files changed, 170 insertions(+), 13 deletions(-)
 create mode 100644 .dockerignore
 create mode 100644 Dockerfile
 create mode 100644 README.Docker.md
 create mode 100644 compose.yaml
 create mode 100644 db/password.txt
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/CC-Lab5/python-docker-dev$ git push -u origin main
Username for 'https://github.com': jay-singhvi
Password for 'https://jay-singhvi@github.com':
remote: Support for password authentication was removed on August 13, 2021.
remote: Please see https://docs.github.com/get-started/getting-started-with-git/about-remote-repositories#cloning-with-https-urls for information on currently recommended
modes of authentication.
fatal: Authentication failed for 'https://github.com/jay-singhvi/CC_Lab5_Temp.git/'
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/CC-Lab5/python-docker-dev$ git push
Username for 'https://github.com': jay-singhvi
Password for 'https://jay-singhvi@github.com':
remote: Support for password authentication was removed on August 13, 2021.
remote: Please see https://docs.github.com/get-started/getting-started-with-git/about-remote-repositories#cloning-with-https-urls for information on currently recommended
modes of authentication.
fatal: Authentication failed for 'https://github.com/jay-singhvi/CC_Lab5_Temp.git/'
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/CC-Lab5/python-docker-dev$
```



Step two: Set up the workflow

Set up your GitHub Actions workflow for building, testing, and pushing the image to Docker Hub.

1. Go to your repository on GitHub and then select the Actions tab.
2. Select set up a workflow yourself.

   This takes you to a page for creating a new GitHub actions workflow file in your repository, under .github/workflows/main.yml by default.

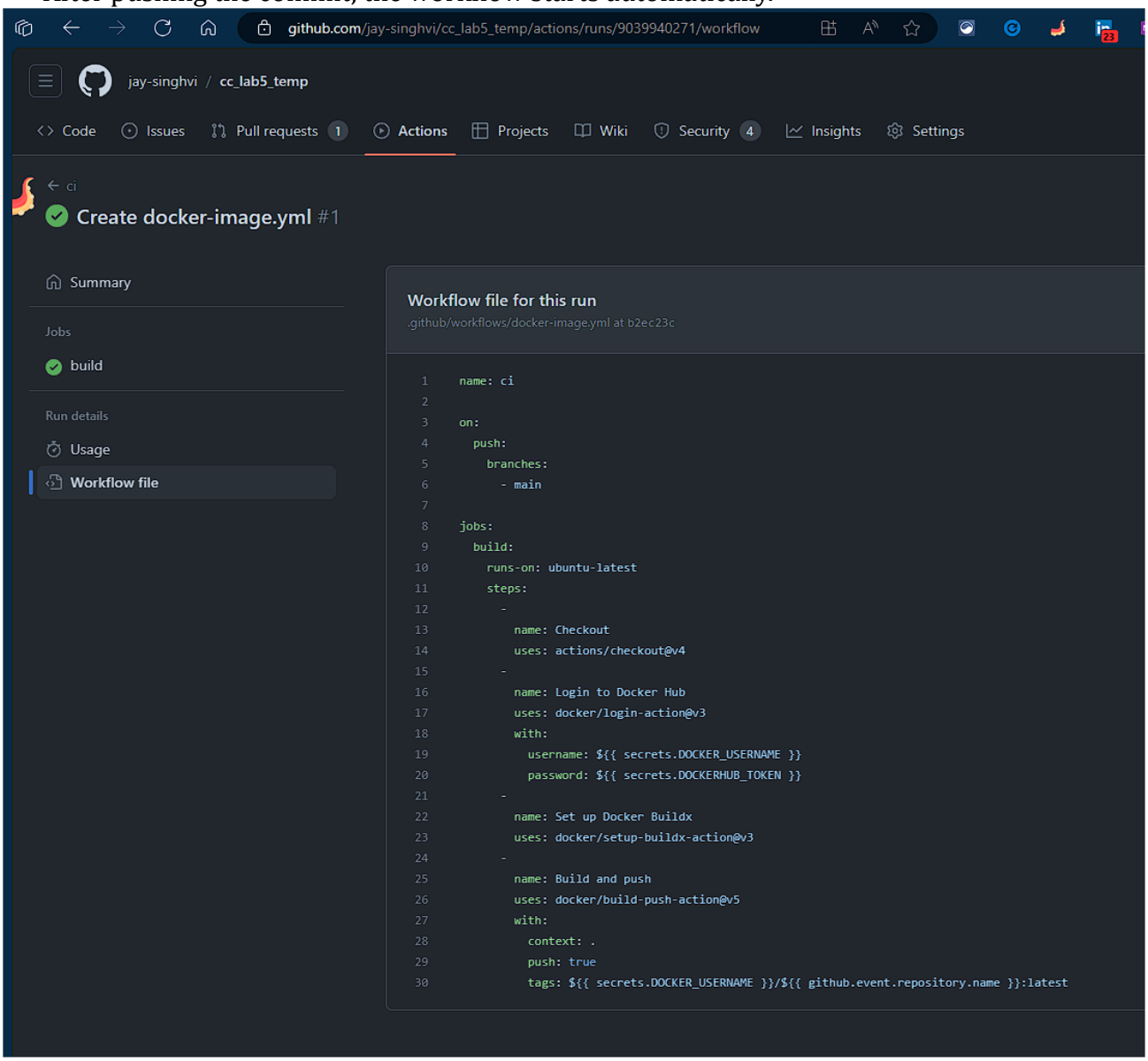
3. In the editor window, copy and paste the following YAML configuration.

```yaml
name: ci

on:
  push:
    branches:
      - main

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      -
        name: Checkout
        uses: actions/checkout@v4
      -
        name: Login to Docker Hub
        uses: docker/login-action@v3
        with:
          username: ${{ secrets.DOCKER_USERNAME }}
          password: ${{ secrets.DOCKERHUB_TOKEN }}
      -
        name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v3
      -
        name: Build and push
        uses: docker/build-push-action@v5
        with:
          context: .
          push: true
          tags: ${{ secrets.DOCKER_USERNAME }}/${{ github.event.repository.name }}:latest
```

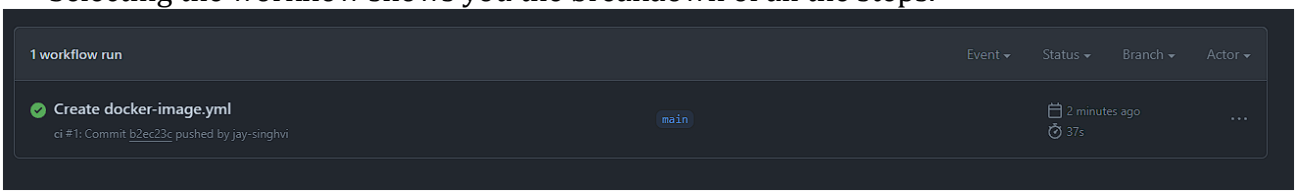
For more information about the YAML syntax used here, see Workflow syntax for GitHub Actions.

Step three: Run the workflow
Save the workflow file and run the job.

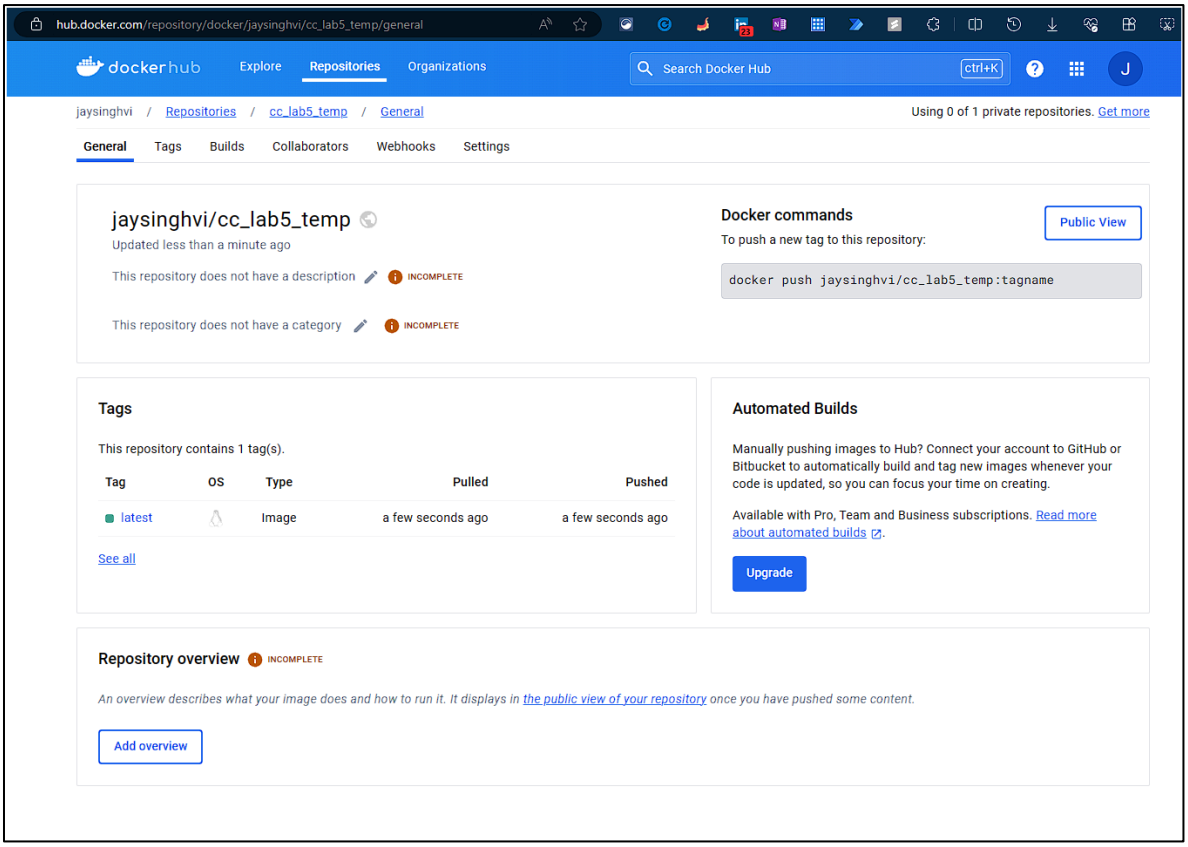
1. Select Commit changes... and push the changes to the main branch.
   After pushing the commit, the workflow starts automatically.



2. Go to the Actions tab. It displays the workflow.
   Selecting the workflow shows you the breakdown of all the steps.



3. When the workflow is complete, go to your repositories on Docker Hub.
   If you see the new repository in that list, it means the GitHub Actions successfully pushed the image to Docker Hub.



Test your Python deployment

Prerequisites

- Complete all the previous sections of this guide, starting with Containerize, a Python application.
- Turn on Kubernetes in Docker Desktop.

Overview

In this section, you'll learn how to use Docker Desktop to deploy your application to a fully featured Kubernetes environment on your development machine. This allows you to test and debug your workloads on Kubernetes locally before deploying.

Create a Kubernetes YAML file

In your python-docker-dev directory, create a file named docker-python-kubernetes.yaml. Open the file in an IDE or text editor and add the following contents. Replace DOCKER_USERNAME/REPO_NAME with your Docker username and the name of the repository that you created in Configure CI/CD for your Python application.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: docker-python-demo
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      service: flask
  template:
    metadata:
      labels:
        service: flask
    spec:
      containers:
       - name: flask-service
         image: DOCKER_USERNAME/REPO_NAME
         imagePullPolicy: Always
         env:
          - name: POSTGRES_PASSWORD
            value: mysecretpassword
---
apiVersion: v1
kind: Service
metadata:
  name: service-entrypoint
  namespace: default
spec:
  type: NodePort
  selector:
    service: flask
```

```
  ports:
  - port: 5000
    targetPort: 5000
    nodePort: 30001
```

```
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/python-docker-and-kubernetes$ cat docker-python-kub
ernetes.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: docker-python-demo
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      service: flask
  template:
    metadata:
      labels:
        service: flask
    spec:
      containers:
       - name: flask-service
         image: jaysinghvi/cc_lab5_temp
         imagePullPolicy: Always
         env:
          - name: POSTGRES_PASSWORD
            value: mysecretpassword
---
apiVersion: v1
kind: Service
metadata:
  name: service-entrypoint
  namespace: default
spec:
  type: NodePort
  selector:
    service: flask
  ports:
  - port: 5000
    targetPort: 5000
    nodePort: 30001
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/python-docker-and-kubernetes$
```

In this Kubernetes YAML file, there are two objects, separated by the ---:

- A Deployment, describing a scalable group of identical pods. In this case, you'll get just one replica, or copy of your pod. That pod, which is described under template, has just one container in it. The container is created from the image built by GitHub Actions in Configure CI/CD for your Python application.
- A NodePort service, which will route traffic from port 30001 on your host to port 5000 inside the pods it routes to, allowing you to reach your app from the network.

To learn more about Kubernetes objects, see the Kubernetes documentation.

Deploy and check your application

1. In a terminal, navigate to python-docker-dev and deploy your application to Kubernetes.

```
kubectl apply -f docker-python-kubernetes.yaml
```

You should see output that looks like the following, indicating your Kubernetes objects were created successfully.

```
deployment.apps/docker-python-demo created
service/service-entrypoint created
```

```
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/python-docker-and-kubernetes$ kubectl apply -f dock
er-python-kubernetes.yaml
deployment.apps/docker-python-demo created
service/service-entrypoint created
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/python-docker-and-kubernetes$
```

2. Make sure everything worked by listing your deployments.

```
kubectl get deployments
```

Your deployment should be listed as follows:

```
NAME              READY   UP-TO-DATE   AVAILABLE   AGE
docker-python-demo   1/1     1            1           15s
```

```
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/python-docker-and-kubernetes$ kubectl get deploymen
ts
NAME                 READY   UP-TO-DATE   AVAILABLE   AGE
docker-python-demo   2/2     2            2           11m
```

This indicates all one of the pods you asked for in your YAML are up and running. Do the same check for your services.

```
kubectl get services
```

You should get output like the following.

```
NAME              TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
kubernetes        ClusterIP   10.96.0.1      <none>        443/TCP        23h
service-entrypoint   NodePort    10.99.128.230   <none>        5000:30001/TCP   75s
```

In addition to the default kubernetes service, you can see your service-entrypoint service, accepting traffic on port 30001/TCP.

```
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/python-docker-and-kubernetes$ kubectl get pods
kubectl get services
NAME                                 READY   STATUS    RESTARTS   AGE
docker-python-demo-5d7896b95d-d5n2n  1/1     Running   0          14m
docker-python-demo-5d7896b95d-tmc2l  1/1     Running   0          14m
NAME                 TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
kubernetes           ClusterIP   10.96.0.1        <none>        443/TCP          49m
service-entrypoint   NodePort    10.108.207.151   <none>        5000:30001/TCP   22m
```

```
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/python-docker-and-kubernetes$ kubectl get nodes -o wide
NAME             STATUS   ROLES           AGE   VERSION   INTERNAL-IP    EXTERNAL-IP   OS-IMAGE         KERNEL-VERSION                     CONTAINER-RUNTIME
docker-desktop   Ready    control-plane   51m   v1.29.2   192.168.65.3   <none>        Docker Desktop   5.15.146.1-microsoft-standard-WSL2   docker://26.1.1
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/python-docker-and-kubernetes$
```

3. In a terminal, curl the service. Note that a database was not deployed in this example.

```
curl http://localhost:30001/
Hello, Docker!!!
```

```
jay@HP-Jay-Singhvi: /mnt/c/(
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/python-docker-and-kubernetes$ curl http://localhost
:30001/
Hello, Docker!!!!jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/python-docker-and-kubernetes$
```

4. Run the following command to tear down your application.

```
kubectl delete -f docker-python-kubernetes.yaml
```

```
jay@HP-Jay-Singhvi:/mnt/c/GitHub_Repos/jay-singhvi/python-docker-and-kubernetes$ kubectl delete -f doc
ker-python-kubernetes.yaml
deployment.apps "docker-python-demo" deleted
service "service-entrypoint" deleted
```