# DevOps challenge

To implement python web application, I have used Flask frame work, which is simple, light weight and provides useful tools and features for creating web application in python.

Python Version: Python 3.11.6

Installed pytest, flask and jinja2 modules

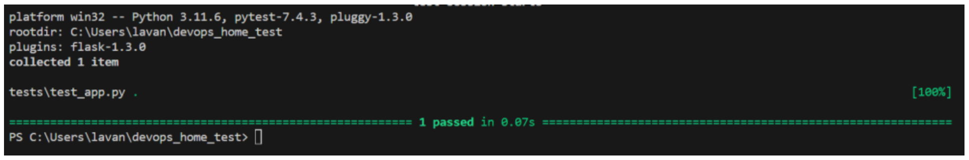
Editor: VS code

Github link: https://github.com/chandrasehkhar/devops_home_test.git

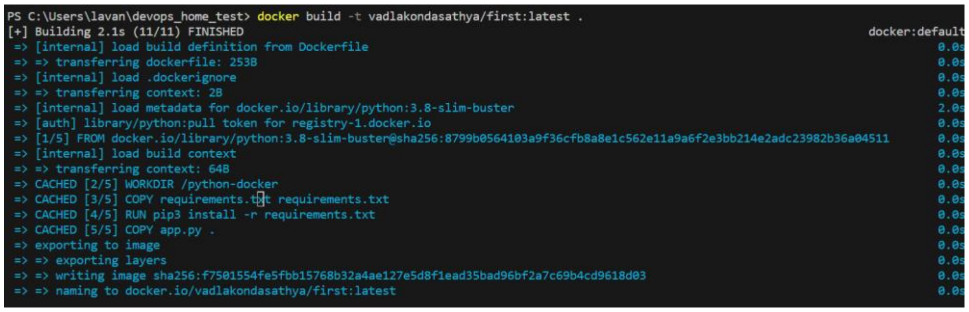
## *Followed below best practices*

- Designed endpoints with proper names
- Used comments in the code when needed
- Code is stable, not an error prone and as I am using flask, performance also good.
- Containerized the application using docker
- Well tested before running the application

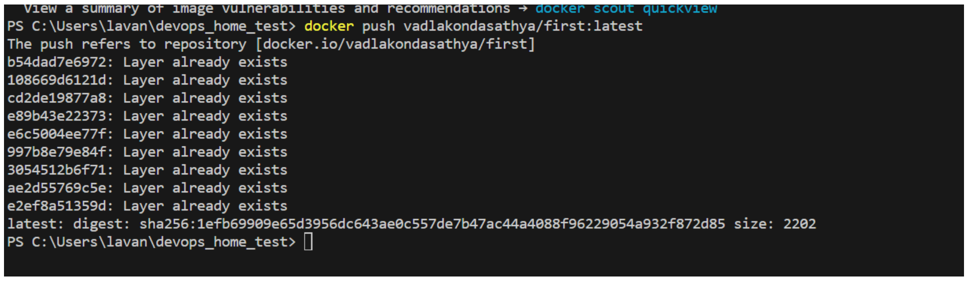
## *Continuous Integration(CI)*

- Continuous Integration is the DevOps best practice. Integrating code changes into main branch, test and build the changes when developer commit the code .
- Wrote the tests for the web application, which ensure quality of the code.
- 

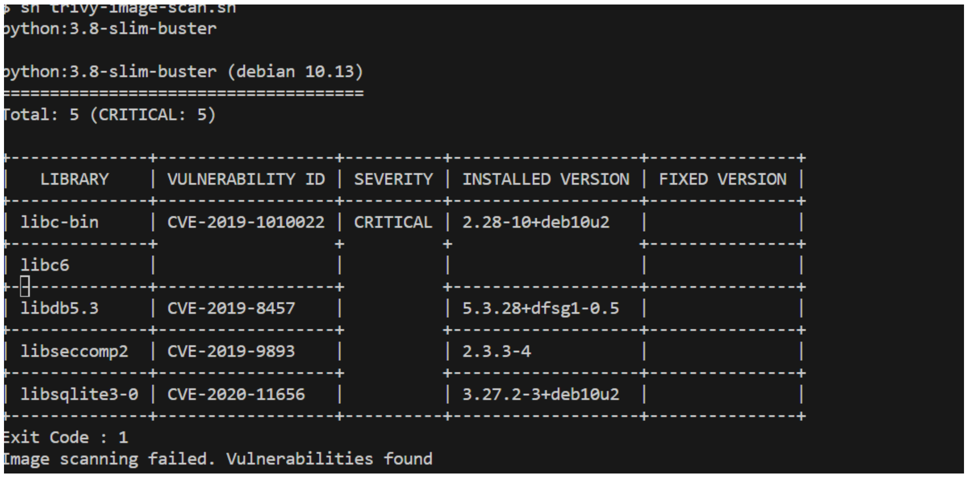
```
platform win32 -- Python 3.11.6, pytest-7.4.3, pluggy-1.3.0
rootdir: C:\Users\lavan\devops_home_test
plugins: flask-1.3.0
collected 1 item

tests\test_app.py .                                                                                    [100%]

=========================================================== 1 passed in 0.07s ===========================================================
PS C:\Users\lavan\devops_home_test> 
```

- **https://github.com/chandrasehkhar/devops_home_test/tree/main/tests**
- Containerized the application using Docker. It has all the dependencies that an application might need to run on any host.  We need to build the image as below then push to repo.
- 

```
PS C:\Users\lavan\devops_home_test> docker build -t vadlakondasathya/first:latest .
[+] Building 2.1s (11/11) FINISHED                                                                  docker:default
 => [internal] load build definition from Dockerfile                                                          0.0s
 => => transferring dockerfile: 253B                                                                          0.0s
 => [internal] load .dockerignore                                                                             0.0s
 => => transferring context: 2B                                                                               0.0s
 => [internal] load metadata for docker.io/library/python:3.8-slim-buster                                     2.0s
 => [auth] library/python:pull token for registry-1.docker.io                                                 0.0s
 => [1/5] FROM docker.io/library/python:3.8-slim-buster@sha256:8799b0564103a9f36cfb8a8e1c562e11a9a6f2e3bb214e2adc23982b36a04511   0.0s
 => [internal] load build context                                                                             0.0s
 => => transferring context: 64B                                                                              0.0s
 => CACHED [2/5] WORKDIR /python-docker                                                                       0.0s
 => CACHED [3/5] COPY requirements.txt requirements.txt                                                       0.0s
 => CACHED [4/5] RUN pip3 install -r requirements.txt                                                         0.0s
 => CACHED [5/5] COPY app.py .                                                                                0.0s
 => exporting to image                                                                                        0.0s
 => => exporting layers                                                                                       0.0s
 => => writing image sha256:f7501554fe5fbb15768b32a4ae127e5d8f1ead35bad96bf2a7c69b4cd9618d03                  0.0s
 => => naming to docker.io/vadlakondasathya/first:latest                                                      0.0s
```

- 

```
  View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\lavan\devops_home_test> docker push vadlakondasathya/first:latest
The push refers to repository [docker.io/vadlakondasathya/first]
b54dad7e6972: Layer already exists
108669d6121d: Layer already exists
cd2de19877a8: Layer already exists
e89b43e22373: Layer already exists
e6c5004ee77f: Layer already exists
997b8e79e84f: Layer already exists
3054512b6f71: Layer already exists
ae2d55769c5e: Layer already exists
e2ef8a51359d: Layer already exists
latest: digest: sha256:1efb69909e65d3956dc643ae0c557de7b47ac44a4088f96229054a932f872d85 size: 2202
PS C:\Users\lavan\devops_home_test>
```

- [https://github.com/chandrasehkhar/devops_home_test/blob/main/Dockerfile](https://github.com/chandrasehkhar/devops_home_test/blob/main/Dockerfile)
- Image scanning is an important security step in containerized applications, we can detect vulnerabilities. For that used **trivy image scanning tool.**
- Trivy is opensource vulnerability scanner with complete database of security vulnerabilities and super easy to use. If we have any vulnerabilities, it will detect them as below to secure the application
- 

```
$ sh trivy-image-scan.sh
python:3.8-slim-buster

python:3.8-slim-buster (debian 10.13)
===================================
Total: 5 (CRITICAL: 5)

+--------------+------------------+----------+-------------------+---------------+
|   LIBRARY    | VULNERABILITY ID | SEVERITY | INSTALLED VERSION | FIXED VERSION |
+--------------+------------------+----------+-------------------+---------------+
| libc-bin     | CVE-2019-1010022 | CRITICAL | 2.28-10+deb10u2   |               |
+--------------+                  +          +                   +---------------+
| libc6        |                  |          |                   |               |
+--------------+------------------+          +-------------------+---------------+
| libdb5.3     | CVE-2019-8457    |          | 5.3.28+dfsg1-0.5  |               |
+--------------+------------------+          +-------------------+---------------+
| libseccomp2  | CVE-2019-9893    |          | 2.3.3-4           |               |
+--------------+------------------+          +-------------------+---------------+
| libsqlite3-0 | CVE-2020-11656   |          | 3.27.2-3+deb10u2  |               |
+--------------+------------------+----------+-------------------+---------------+
Exit Code : 1
Image scanning failed. Vulnerabilities found
```

- https://github.com/chandrasehkhar/devops_home_test/blob/main/trivy-image-scan.sh
- To automate the process of build, test and scanning of the image used jenkins tool. Which is an open source and efficient tool.
- [https://github.com/chandrasehkhar/devops_home_test/blob/main/Jenkinsfile](https://github.com/chandrasehkhar/devops_home_test/blob/main/Jenkinsfile)
- Setup the webhooks to trigger the job automatically , when the code merged into master.

## *Kubernetes Deployment*

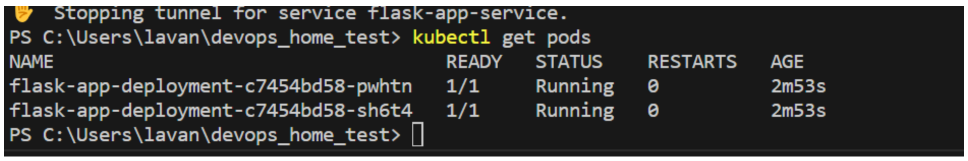
- Kubernetes is an opensource orchestration tool, which is used for automating deployment, scaling and management of containerized applications.
- Pulled the image from the repository
- Our Application(container) is running in the pods, which we exposed using service
- Here, I am using deployment which is useful to scale in and scale out of replica pods. Always runs desired number of pods. We can rollout to previous versions easily.
- I have deployed the application with 2 pods currently, we can increase pods using kubectl command or by modifying in deployment.
- I have created a service, which exposes my application to outside the cluster and listens to user input.

- It is highly available solution, if one node goes down in the cluster, Kubernetes moves our application to other node for availability of application.

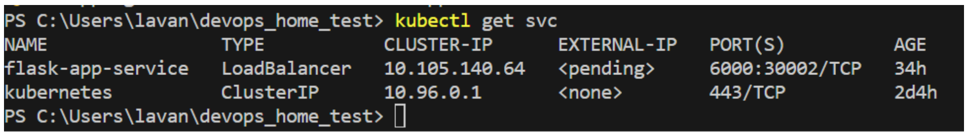
https://github.com/chandrasehkhar/devops_home_test/blob/main/deployment.yaml

We can deploy our application on kubernetes cluster using below command. And check pods are running or not.

kubectl apply -f deployment.yaml

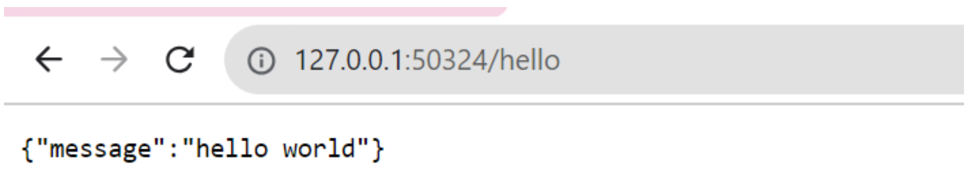


```
   Stopping tunnel for service flask-app-service.
PS C:\Users\lavan\devops_home_test> kubectl get pods
NAME                                   READY   STATUS    RESTARTS   AGE
flask-app-deployment-c7454bd58-pwhtn   1/1     Running   0          2m53s
flask-app-deployment-c7454bd58-sh6t4   1/1     Running   0          2m53s
PS C:\Users\lavan\devops_home_test>
```



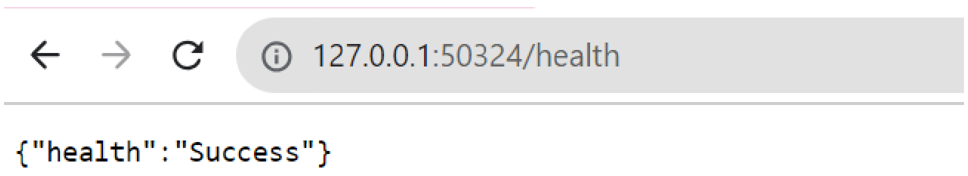
```
PS C:\Users\lavan\devops_home_test> kubectl get svc
NAME                TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
flask-app-service   LoadBalancer   10.105.140.64   <pending>     6000:30002/TCP   34h
kubernetes          ClusterIP      10.96.0.1       <none>        443/TCP          2d4h
PS C:\Users\lavan\devops_home_test>
```

Access the application from the browser with /hello and /health endpoints



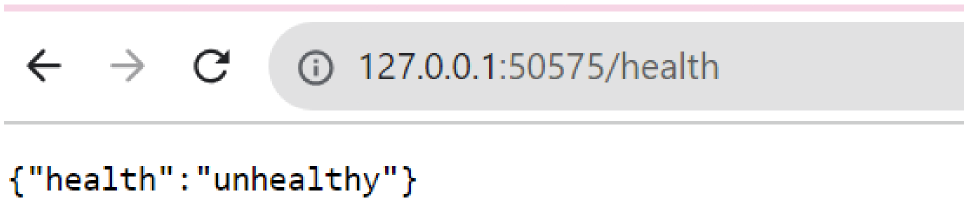
127.0.0.1:50324/hello

```
{"message":"hello world"}
```



127.0.0.1:50324/health

```
{"health":"Success"}
```


127.0.0.1:50575/health

```
{"health":"unhealthy"}
```