In-class Activity 2

Galgamuge Emmanuel Fernando

C0918066

CBD 3324: Containerization and Container Delivery

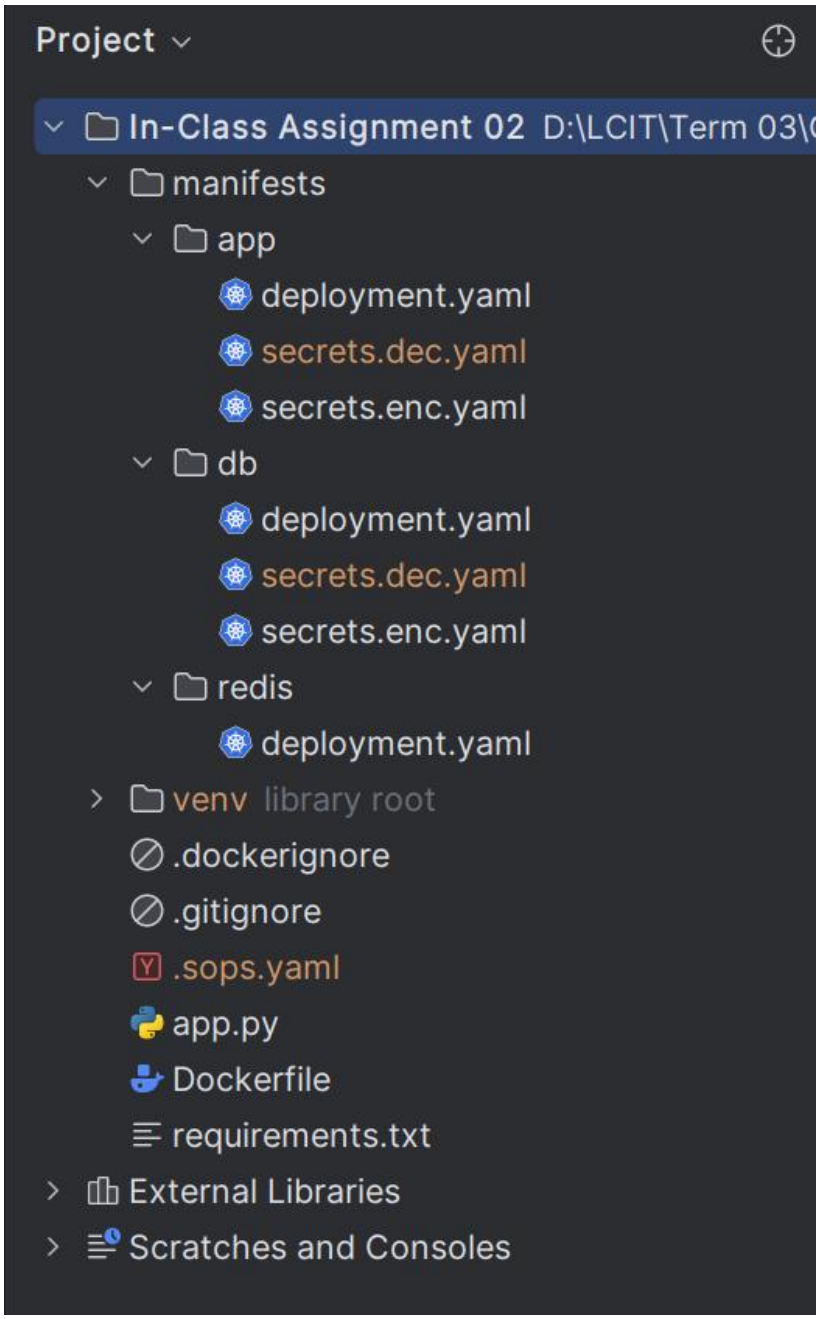
November 19, 2024

Repository URL: https://github.com/RuFerdZ/cbd-3324-in-class-activity-02

1. Project Structure

The below image shows the folder structure I have used for the project and its deployment.

The decrypted secrets and the encryption key is ignored while been pushed to the repository.



2. Deploy the database

The below image shows the deployment of all resources of the database in the ***db-ns*** namespace and also the deployment of the secret after decrypting it.

```
rusir@Rusiru-PC MINGW64 /d/LCIT/Term 03/CBD 3324 - Containerizatio
$ kubectl apply -f manifests/db/deployment.yaml
namespace/db-ns created
configmap/db-configmap created
deployment.apps/database-deployment created
service/db-service created

rusir@Rusiru-PC MINGW64 /d/LCIT/Term 03/CBD 3324 - Containerizatio
$ sops -d manifests/db/secrets.enc.yaml | kubectl apply -f -
secret/db-secret created
```

The below image shows all the resource created in the ***db-ns*** namespace.

```
rusir@Rusiru-PC MINGW64 /d/LCIT/Term 03/CBD 3324 - Containerization and Container Delive
$ kubectl get all -n db-ns
NAME                                    READY   STATUS    RESTARTS   AGE
pod/database-deployment-d466f7c6b-sgbn9   1/1     Running   0          48s

NAME                 TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
service/db-service   ClusterIP   10.43.179.35   <none>        5432/TCP   48s

NAME                                  READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/database-deployment   1/1     1            1           48s

NAME                                             DESIRED   CURRENT   READY   AGE
replicaset.apps/database-deployment-d466f7c6b    1         1         1       48s
```

3. Set up the Database table.

SSH to the database pod and then log in to the database as the *postgres* user, create the *users* table and insert a sample user.

```
rusir@Rusiru-PC MINGW64 /d/LCIT/Term 03/CBD 3324 - Containerization and Container Delivery/I
$ kubectl exec -n db-ns pod/database-deployment-d466f7c6b-tpmd8   -it -- bash
root@database-deployment-d466f7c6b-tpmd8:/# psql -h localhost -U postgres -d flask_app_db
psql (13.17 (Debian 13.17-1.pgdg120+1))
Type "help" for help.

flask_app_db=# CREATE TABLE users (ID INT PRIMARY KEY NOT NULL, NAME TEXT NOT NULL);
CREATE TABLE
flask_app_db=# INSERT INTO users VALUES (1, '3324_1');
INSERT 0 1
flask_app_db=#
```

4. Deploy Redis Cache

The below image shows the deployment of all resources of the Redis in the ***redis-ns*** namespace.

```
rusir@Rusiru-PC MINGW64 /d/LCIT/Term 03/CBD 3324 - Containeriz
$ kubectl apply -f manifests/redis/deployment.yaml
namespace/redis-ns created
deployment.apps/redis-deployment created
service/redis-service created
```

The below image shows all the resource created in the ***redis-ns*** namespace.

```
rusir@Rusiru-PC MINGW64 /d/LCIT/Term 03/CBD 3324 - Containerization and Container
$ kubectl get all -n redis-ns
NAME                                    READY   STATUS    RESTARTS   AGE
pod/redis-deployment-787b8cddcf-d4skq   1/1     Running   0          13s

NAME                    TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
service/redis-service   ClusterIP   10.43.170.63   <none>        6379/TCP   13s

NAME                               READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/redis-deployment   1/1     1            1           13s

NAME                                          DESIRED   CURRENT   READY   AGE
replicaset.apps/redis-deployment-787b8cddcf   1         1         1       13s
```

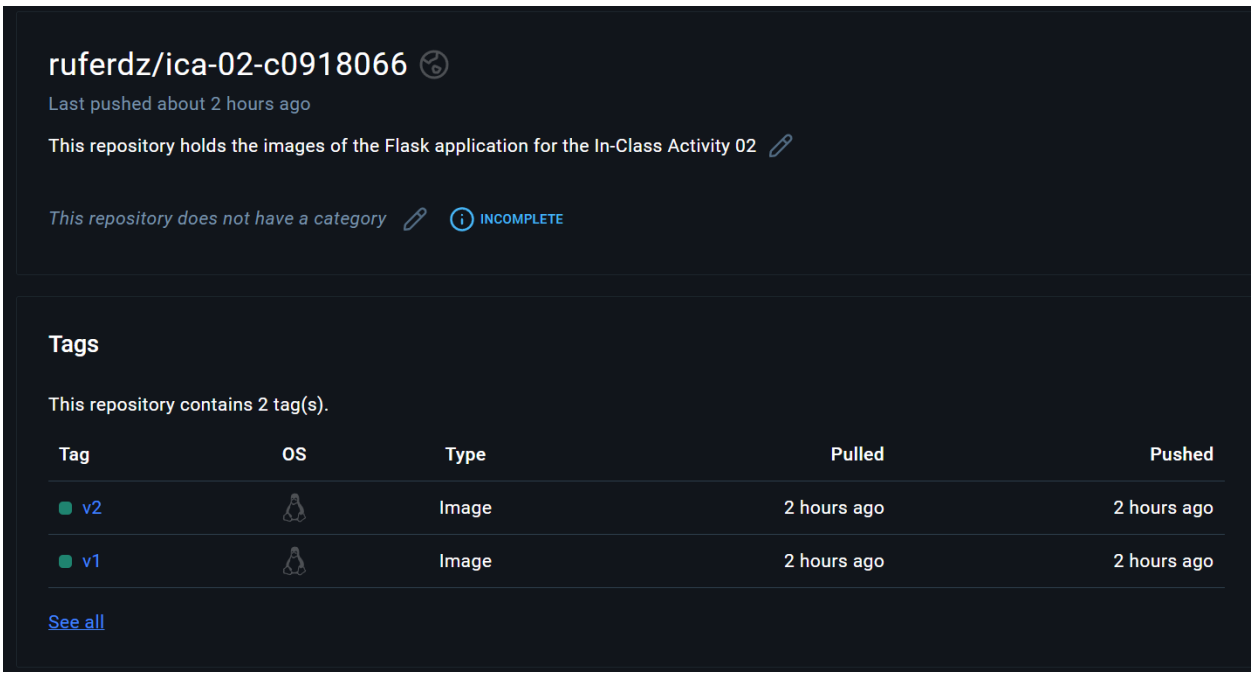
5. Create Docker Image for Flask application

Initially, a Dockerfile was created for the flask application.

```
# Set the base image for the application to run on
FROM python:3.9-slim

# Set the working directory
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install any needed packages specified in requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Expose the port the app runs on
EXPOSE 5000

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

Next, I built and pushed it to DockerHub, (https://hub.docker.com/repository/docker/ruferdz/ica-02-c0918066/general) using the following commands.

- *docker build -t ruferdz/ica-02-c0918066:v2 .*
- *docker push ruferdz/ica-02-c0918066:v2*



6. Deploy flask application

The below image shows the deployment of all resources of the flask application in the ***app-ns*** namespace and also the deployment of the secret after decrypting it.

```
rusir@Rusiru-PC MINGW64 /d/LCIT/Term 03/CBD 3324 - Containerization an
$ kubectl apply -f manifests/app/deployment.yaml
namespace/app-ns created
configmap/app-configmap created
deployment.apps/app-deployment created
service/app-service created

rusir@Rusiru-PC MINGW64 /d/LCIT/Term 03/CBD 3324 - Containerization an
$ sops -d manifests/app/secrets.enc.yaml | kubectl apply -f -
secret/app-secret created
```

The below image shows all the resource created in the ***app-ns*** namespace.

```
rusir@Rusiru-PC MINGW64 /d/LCIT/Term 03/CBD 3324 - Containerization and Container Delivery/In-C
$ kubectl get all -n app-ns
NAME                                  READY   STATUS    RESTARTS   AGE
pod/app-deployment-784d46485b-fgsv9   1/1     Running   0          21s

NAME                  TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)          AGE
service/app-service   LoadBalancer   10.43.244.102   192.168.127.2   5000:32475/TCP   21s

NAME                              READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/app-deployment    1/1     1            1           21s

NAME                                         DESIRED   CURRENT   READY   AGE
replicaset.apps/app-deployment-784d46485b    1         1         1       21s
```

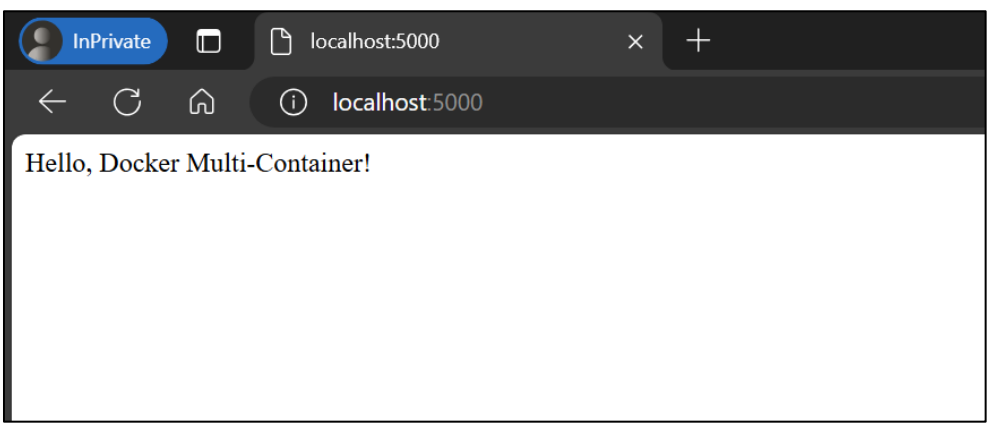
7. Port forward flask application service to access it.

The service serving the flask application is port-forwarded so that we can access it publicly (in this case the localhost) via port 5000 for testing.

```
rusir@Rusiru-PC MINGW64 /d/LCIT/Term 03/CBD 3324 - Containerization and C
$ kubectl port-forward service/app-service 5000:5000 -n app-ns
Forwarding from [::1]:5000 -> 5000
Handling connection for 5000
Handling connection for 5000
Handling connection for 5000
Handling connection for 5000
```

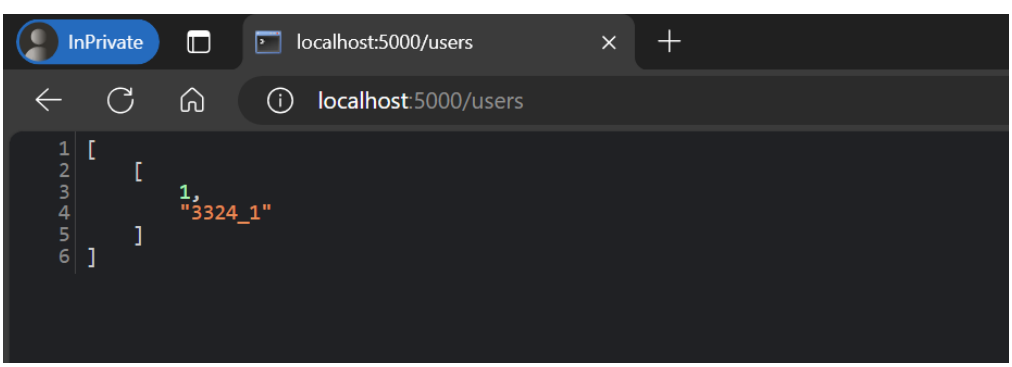
8. Access the endpoints

- First, I accessed the root path- *http:localhost:5000*



- Next accessed the /users path - *http:localhost:5000*/users

This will return the users stored in the Postgres database.



- Next accessed the /cache path - *http:localhost:5000*/cache

If the caching works successfully, it will return the below message.