Assignment   
  
Description:   
Multi-Container Application Deployment with Docker Compose and Kubernetes.  
  
This project involves deploying a multi-container application using Docker Compose and Kubernetes. The application consists of a frontend, backend, and database container. The goal of the project is to design and implement a scalable and efficient containerized application.

**Tools Used:**

Docker, github, docker hub, Kubernetes, docker compose

**Approach:**

The approach used in this project is to design and implement a containerized application using Docker Compose and Kubernetes. The application is divided into three containers: frontend, backend, and database.

The architecture of the application consists of the following components:

* Frontend Container: This container runs the frontend application.
* Backend Container: This container runs the backend application.
* Database Container: This container runs the database.
* Docker-compose: This builds and run images locally.
* Kubernetes Cluster: This is the container orchestration platform used to manage and scale the application. Docker is the provider.

Steps:   
1. Cloned frontend and backend codes.  
2. Created Dockerfiles for each backend, frontend and database.  
3. Firstly created Database container and tried backend locally connecting to the container of database.  
  
output :   
 Express : backend

4. Created Container for backend and ran that, getting same results again. Working properly.  
5. Created docker images of all three database, backend and frontend,  
 Used this image for containerization and deployment.  
6. Created docker-compose.yml file for multi-container management.

7. finally created Kubernetes manifest files i.e. deployment and service file for backend, frontend and database.  
  
8. Deployed all three using Kubernetes  
 kubectl apply -f <deployment/ service>  
  
9. Database and backend are running on Cluster-IP and frontend is running on loadbalancer for outerworld interaction. This is a wauy of exposing your app to outer users.

10. For automation we can use different tools like AWS autoscaling feature same as in Kubernetes deployment but in Kubernetes desired state is already define and in autscaling you set threshold for desired state.  
  
11. We can use terraform also to define our infra and running the app in that.  
  
Hence getting desired results

Error:   
  
Faced multiple errors.  
  
1. Most common and frequently arising error is typo , where there is a little typing error in the code or command due to which it does’t work.  
  
2. Connection error: All the containers were not connecting earlier because of the .env path not set.  
 Solved this by providing .env and also one can do it in Kubernetes using config file.  
  
3. Code Error: There is some error in code due to which it is stuck in some infinite loop and hence loading continuesly, need to check the code again.  
Did some corrections, package update but still problem persists.  
  
4. Image not present: This issue is faced when your image is not present in the respective repository or is not able to detect or find where it is.   
 Solved this by saving docker image in docker hub  
 docker push usedid/imagename:tag

5. multiple error faced that were resolved during running the app and deploying to Kubernetes.