**1. Task Summary and Implementation**

In this task, I extended the previously containerised and Kubernetes-deployed bilingual calculator application by integrating a MongoDB database for persistent storage of operation history. The goal was to allow the backend microservice to perform full CRUD operations against a dedicated MongoDB service running inside the same Kubernetes cluster.  
  
To achieve this, I created several Kubernetes resource configurations:  
- mongo-deployment.yaml: Deploys a single-replica MongoDB instance using the official image.  
- mongo-service.yaml: Exposes MongoDB internally via a ClusterIP service.  
- mongo-secret.yaml: Stores the root username and password for MongoDB.  
- mongo-pv.yaml and mongo-pvc.yaml: Configure persistent volume storage using hostPath.  
  
The calculator application's deployment.yaml was then modified to inject a MONGO\_URL environment variable, pointing to the internal MongoDB service. The backend connects using the Mongoose ODM library. A new /history route was implemented to retrieve stored operations.

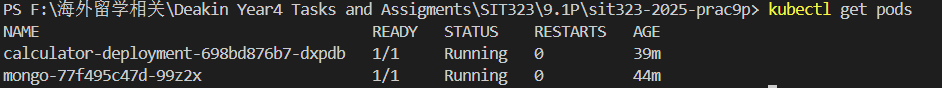
**2. Kubernetes Commands and Interaction Process**

Step 1: Apply Kubernetes Resources

kubectl apply -f mongo-secret.yaml  
kubectl apply -f mongo-pv.yaml  
kubectl apply -f mongo-pvc.yaml  
kubectl apply -f mongo-deployment.yaml  
kubectl apply -f mongo-service.yaml  
kubectl apply -f deployment.yaml  
kubectl apply -f service.yaml

Step 2: Confirm Pod Status

kubectl get pods



Step 3: Access the Application

kubectl port-forward service/calculator-service 3000:80  
Then visit http://localhost:3000 in a browser.

**3. MongoDB Functional Verification**

After performing operations via the calculator UI, I visited the /history route to verify MongoDB recording. This endpoint returns recent arithmetic operation logs.  
  
http://localhost:3000/history



**GitHub Link:** <https://github.com/Lonely-DM/SIT323/tree/main/9.1P/sit323-2025-prac9p>