

Github :

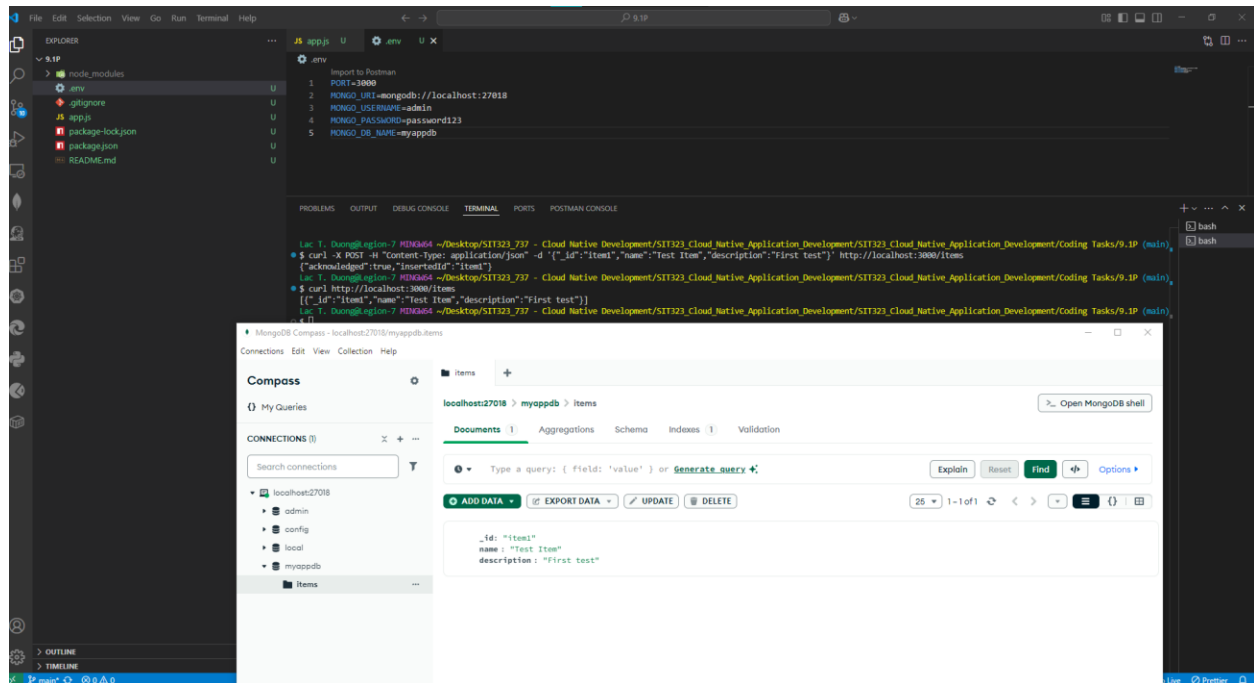
[https://github.com/HaydenDuong/SIT323\\_Cloud\\_Native\\_Application\\_Development/tree/main/Coding%20Tasks/9.1P](https://github.com/HaydenDuong/SIT323_Cloud_Native_Application_Development/tree/main/Coding%20Tasks/9.1P)

Video Demo:

<https://deakin.au.panopto.com/Panopto/Pages/Viewer.aspx?id=b0128c10-18b4-4079-bca5-b2d500ec8c60>

Pictures & Explanation:

Testing the application to determine whether it is connected to mongoDB or not and to test CRUD endpoints.



Picture 1 – App.js is connected to MongoDB, shown in MongoDB Compass.

Because this application will simulate how microservice application look like so the application-side and the database-side will have their own YAML files like Deployment, and Service files. As for the database, it will have additional Secret, and Persistent Volume – Persistent Volume Claim files.

Their contents are shown as follows (or can be accessed publicly through Github link)

```

mongo-pv.yaml U X
mongo-pv.yaml > {} spec > {} resources > {} requests > storage
2   kind: PersistentVolume
3   metadata:
4     name: mongo-pv
5   spec:
6     capacity:
7       storage: 1Gi
8     accessModes:
9       - ReadWriteOnce
10    hostPath:
11      path: /data/mongo-pv
12  ---
13  apiVersion: v1
14  kind: PersistentVolumeClaim
15  metadata:
16    name: mongo-pvc
17  spec:
18    accessModes:
19      - ReadWriteOnce
20    resources:
21      requests:
22        storage: 1Gi

```

Picture 2 - mongo-pv.yaml

```

mongo-secret.yaml X
mongo-secret.yaml > {} data
io.k8s.api.core.v1.Secret (v1@secret.json)
1   apiVersion: v1
2   kind: Secret
3   metadata:
4     name: mongo-secret
5   type: Opaque
6   data:
7     username: YWRtaW4= # base64 encoded value of 'admin'
8     password: cGFzc3dvcmQxMjM= # base64 encoded value of 'password123'

```

Picture 3 – mongo-secret.yaml

```

❏ mongo-deployment.yaml 1 ❏
❏ mongo-deployment.yaml > {} spec > {} template > {} spec > [
1   io.k8s.api.apps.v1.Deployment (v1@deployment.json)
2   apiVersion: apps/v1
3   kind: Deployment
4   metadata:
5     name: mongo
6   spec:
7     selector:
8       matchLabels:
9         app: mongo
10    template:
11      metadata:
12        labels:
13          app: mongo
14      spec:
15        containers:
16          - name: mongo
17            image: mongo:latest
18            ports:
19              - containerPort: 27017
20            volumeMounts:
21              - name: mongo-persistent-storage
22                mountPath: /data/db
23            env:
24              - name: MONGO_INITDB_ROOT_USERNAME
25                valueFrom:
26                  secretKeyRef:
27                    name: mongo-secret
28                    key: username
29              - name: MONGO_INITDB_ROOT_PASSWORD
30                valueFrom:
31                  secretKeyRef:
32                    name: mongo-secret
33                    key: password
34            volumes:
35              - name: mongo-persistent-storage
36                persistentVolumeClaim:
37                  claimName: mongo-pvc

```

Picture 4 – mongo-deployment.yaml

```

❏ mongo-service.yaml ❏
❏ mongo-service.yaml > {} spec > [ ] ports > {} 0 > # targetPort
1   io.k8s.api.core.v1.Service (v1@service.json)
2   apiVersion: v1
3   kind: Service
4   metadata:
5     name: mongo
6   spec:
7     selector:
8       app: mongo
9     ports:
10      - protocol: TCP
11        port: 27017
12        targetPort: 27017

```

Picture 5 – mongo-service.yaml

```

app-deployment.yaml 1 X
app-deployment.yaml > {} spec > {} template > {} spec > [ ] containers >
io.k8s.api.apps.v1.Deployment (v1@deployment.json)
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: my-app
5  spec:
6    replicas: 1
7    selector:
8      matchLabels:
9        app: my-app
10   template:
11     metadata:
12       labels:
13         app: my-app
14     spec:
15       containers:
16       - name: my-app
17         image: hayden2310/sit323-task91p:v1
18         ports:
19         - containerPort: 3000
20         env:
21         - name: PORT
22           value: "3000"
23         - name: MONGO_URI
24           value: "mongodb://mongo:27017/myappdb"
25         - name: MONGO_USERNAME
26           valueFrom:
27             secretKeyRef:
28               name: mongo-secret
29               key: username
30         - name: MONGO_PASSWORD
31           valueFrom:
32             secretKeyRef:
33               name: mongo-secret
34               key: password

```

Picture 6 – app-deployment.yaml

```

app-service.yaml X
app-service.yaml > {} spec > type
io.k8s.api.core.v1.Service (v1@service.json)
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: my-app
5  spec:
6    selector:
7      app: my-app
8    ports:
9      - protocol: TCP
10        port: 80
11        targetPort: 3000
12    type: LoadBalancer

```

## Picture 7 - app-service.yaml

After built an image and pushed it to Docker Hub / localhost repository, run “kubectl apply -f ....yaml” on those YAML files.

The application, by now, can be tested by:

1. Type in command: “kubectl port-forward svc/my-app 8080:80”.
2. Upon received output message like “Forwarding from 127.0.0.1:8080 → 3000”, then open a new terminal and type in command: “kubectl port-forward svc/mongo 27018:27017” – this command will allow user to view the result of CRUD interaction with mongoDB database on mongoDB Compass.
3. Open mongoDB Compass and choose “Add new connection” button and paste this following URI: “mongodb://admin:password123@localhost:27018/”
4. Upon successful connected, open a new terminal in vscode and try the following CRUD command:

- a. For adding a new document item:

```
curl -X POST -H "Content-Type: application/json" -d '{"_id":"item1","name":"Test Item","description":"Just testing"}' http://localhost:8080/items
```

- b. For updating an existing document item:

```
curl -X PUT -H "Content-Type: application/json" -d '{"name":"Updated Item"}' http://localhost:8080/items/item1
```

- c. Get a specific document item:

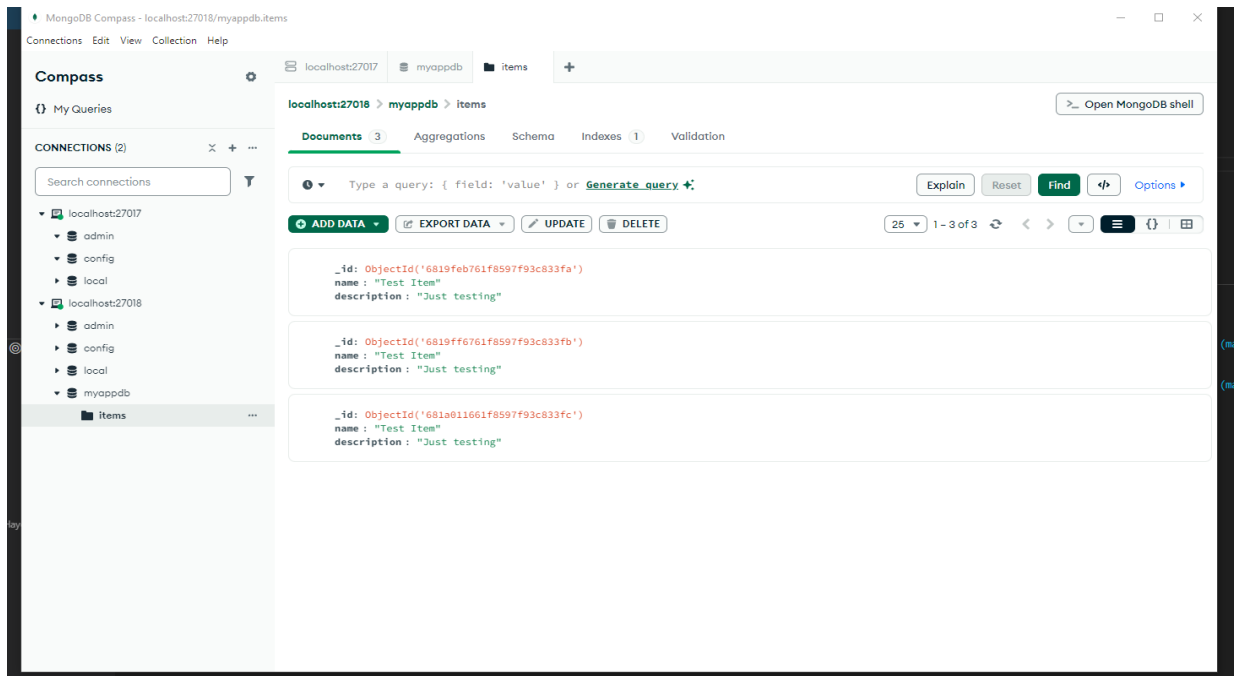
```
curl http://localhost:8080/items/item1
```

- d. Get all document items

```
curl http://localhost:8080/items
```

- e. Delete a document item:

```
curl -X DELETE http://localhost:8080/items/item1
```



Picture 8 – Results after adding three document item.