Task 9.1P

Kubernetes Manifests

1. **MongoDB Deployment**
   - Deploys a single MongoDB pod.
   - Uses secrets for MONGO_INITDB_ROOT_USERNAME and MONGO_INITDB_ROOT_PASSWORD.
   - Persists data with a PersistentVolumeClaim (PVC).

2. **MongoDB Service**
   - Exposes MongoDB internally (headless service).

```yaml
# MongoDB Service (Headless for internal access)
apiVersion: v1
kind: Service
metadata:
  name: mongodb-service
spec:
  selector:
    app: mongodb
  ports:
    - protocol: TCP
      port: 27017
      targetPort: 27017
  clusterIP: None  # Headless service for internal communication
```

3. **MongoDB Secret**
   - Stores credentials securely using base64 encoding.

```yaml
---
# MongoDB Secret for credentials
apiVersion: v1
kind: Secret
metadata:
  name: mongodb-secret
type: Opaque
data:
  username: dXNlcjE=     #user1 (Base64 encoded)
  password: MTIzNDU=     #12345 (Base64 encoded)

---
```

4. **MongoDB PersistentVolumeClaim**
   - Allocates 5Gi of storage for MongoDB data.

```yaml
---
# Persistent Volume Claim for MongoDB
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mongo-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi

---
```

5. **MongoDB Backup CronJob**
   - Backs up MongoDB data every 6 hours using mongodump.
   - Saves backups to a PVC.

```yaml
---
# CronJob for MongoDB Backups
apiVersion: batch/v1
kind: CronJob
metadata:
  name: mongodb-backup-cronjob
spec:
  schedule: "0 */6 * * *"  # Every 6 hours
  jobTemplate:
    spec:
      template:
        spec:
          containers:
            - name: mongodb-backup
              image: mongo:latest
              command:
                - /bin/sh
                - -c
                - |
                  mongodump --uri="mongodb://user1:12345@mongodb-service:27017" --out=/backup/$(date +\%F_\%T)
              volumeMounts:
                - name: backup-storage
                  mountPath: /backup
          restartPolicy: OnFailure
          volumes:
            - name: backup-storage
              persistentVolumeClaim:
                claimName: mongo-backup-pvc

---
```

6. **Backup PVC**
   - Stores MongoDB backups.

```yaml
---
# PVC for storing MongoDB backups
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mongo-backup-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
```

7. **MongoDB Exporter**
   - Deploys Prometheus-compatible exporter for monitoring MongoDB metrics.
   - Exposed via a separate service on port 9216.

```yaml
---
# MongoDB Exporter Deployment
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb-exporter
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mongodb-exporter
  template:
    metadata:
      labels:
        app: mongodb-exporter
    spec:
      containers:
        - name: mongodb-exporter
          image: bitnami/mongodb-exporter:latest
          ports:
            - containerPort: 9216
          env:
            - name: MONGODB_URI
              value: "mongodb://user1:12345@mongodb-service:27017"

---
# MongoDB Exporter Service
apiVersion: v1
kind: Service
metadata:
  name: mongodb-exporter-service
spec:
  selector:
    app: mongodb-exporter
  ports:
    - protocol: TCP
      port: 9216
      targetPort: 9216

---
```

8. **Calculator Microservice Deployment**
   - Node.js microservice for basic arithmetic operations.
   - Connects to MongoDB using environment variable MONGO_URI.
   - Includes a liveness probe on /health.

```yaml
---
# Calculator Deployment
apiVersion: apps/v1
kind: Deployment
metadata:
  name: calculator-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: calculator
  template:
    metadata:
      labels:
        app: calculator
    spec:
      containers:
        - name: calculator
          image: calculator-microservice:latest
          imagePullPolicy: Never
          ports:
            - containerPort: 3000
          env:
            - name: MONGO_URI
              value: "mongodb://user1:12345@mongodb-service:27017"  # MongoDB URI
          livenessProbe:
            httpGet:
              path: /health
              port: 3000
            initialDelaySeconds: 15
            periodSeconds: 10

---
```

9. **Calculator Service**
   - Exposes calculator microservice on port 3000.

```yaml
---
# Calculator Service
apiVersion: v1
kind: Service
metadata:
  name: calculator-service
spec:
  selector:
    app: calculator
  ports:
    - protocol: TCP
      port: 3000
      targetPort: 3000
```

To indicate that **multiple YAML resources (e.g., Deployment, Service, ConfigMap, etc.) are defined in a single deployment.yaml file**, we **separate each resource block with ---**, which is the YAML document separator.

Working of the service

1. We start by creating the docker image of calculator service
   - *docker build -t calculator-microservice:latest .*

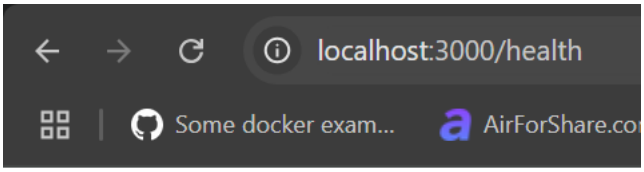
```
(base) PS C:\My Work\data\Deakin\Semester 4\Cloud Native Applications\Task 9.1P\sit737-2025-prac9p> docker build -t calculator-microservice:latest .
 [+] Building 5.4s (10/10) FINISHED
  => [internal] load build definition from Dockerfile
  => => transferring dockerfile: 642B
  => [internal] load metadata for docker.io/library/node:18
  => [internal] load .dockerignore
  => => transferring context: 2B
  => [1/5] FROM docker.io/library/node:18@sha256:867be01f97d45cb7d89a8ef0b328d23e8207412ebec4564441ed8cabc8cc4ecd
  => => resolve docker.io/library/node:18@sha256:867be01f97d45cb7d89a8ef0b328d23e8207412ebec4564441ed8cabc8cc4ecd
  => [internal] load build context
  => => transferring context: 117.02kB
  => CACHED [2/5] WORKDIR /usr/src/app
  => [3/5] COPY package*.json ./
  => [4/5] RUN npm install
  => [internal] load metadata for docker.io/library/node:18
  => [internal] load .dockerignore
  => => transferring context: 2B
  => [1/5] FROM docker.io/library/node:18@sha256:867be01f97d45cb7d89a8ef0b328d23e8207412ebec4564441ed8cabc8cc4ecd
  => => resolve docker.io/library/node:18@sha256:867be01f97d45cb7d89a8ef0b328d23e8207412ebec4564441ed8cabc8cc4ecd
  => [internal] load build context
  => => transferring context: 117.02kB
  => CACHED [2/5] WORKDIR /usr/src/app
  => [3/5] COPY package*.json ./
  => [4/5] RUN npm install
  => [internal] load .dockerignore
  => => transferring context: 2B
  => [1/5] FROM docker.io/library/node:18@sha256:867be01f97d45cb7d89a8ef0b328d23e8207412ebec4564441ed8cabc8cc4ecd
  => => resolve docker.io/library/node:18@sha256:867be01f97d45cb7d89a8ef0b328d23e8207412ebec4564441ed8cabc8cc4ecd
  => [internal] load build context
  => => transferring context: 117.02kB
  => CACHED [2/5] WORKDIR /usr/src/app
  => [3/5] COPY package*.json ./
  => [4/5] RUN npm install
  => [1/5] FROM docker.io/library/node:18@sha256:867be01f97d45cb7d89a8ef0b328d23e8207412ebec4564441ed8cabc8cc4ecd
  => => resolve docker.io/library/node:18@sha256:867be01f97d45cb7d89a8ef0b328d23e8207412ebec4564441ed8cabc8cc4ecd
  => [internal] load build context
  => => transferring context: 117.02kB
  => CACHED [2/5] WORKDIR /usr/src/app
  => [3/5] COPY package*.json ./
  => [4/5] RUN npm install
  => => resolve docker.io/library/node:18@sha256:867be01f97d45cb7d89a8ef0b328d23e8207412ebec4564441ed8cabc8cc4ecd
  => [internal] load build context
  => => transferring context: 117.02kB
  => CACHED [2/5] WORKDIR /usr/src/app
  => [3/5] COPY package*.json ./
```

2. We then apply all of the resources together using **kubectl**
   - *kubectl apply -f deployment.yaml*

```
(base) PS C:\My Work\data\Deakin\Semester 4\Cloud Native Applications\Task 9.1P\sit737-2025-prac9p> kubectl apply -f deployment.yaml
deployment.apps/mongodb-deployment created
service/mongodb-service created
persistentvolumeclaim/mongo-pvc unchanged
secret/mongodb-secret unchanged
cronjob.batch/mongodb-backup-cronjob created
persistentvolumeclaim/mongo-backup-pvc unchanged
deployment.apps/mongodb-exporter created
service/mongodb-exporter-service created
deployment.apps/calculator-deployment created
service/calculator-service created
```

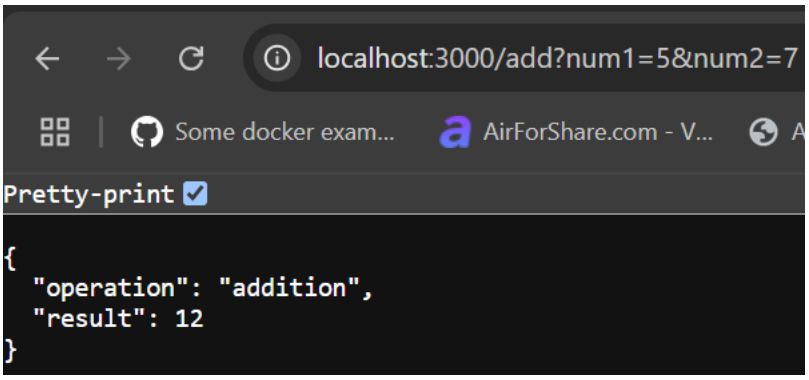
Testing the service and the database

1. Health Endpoint

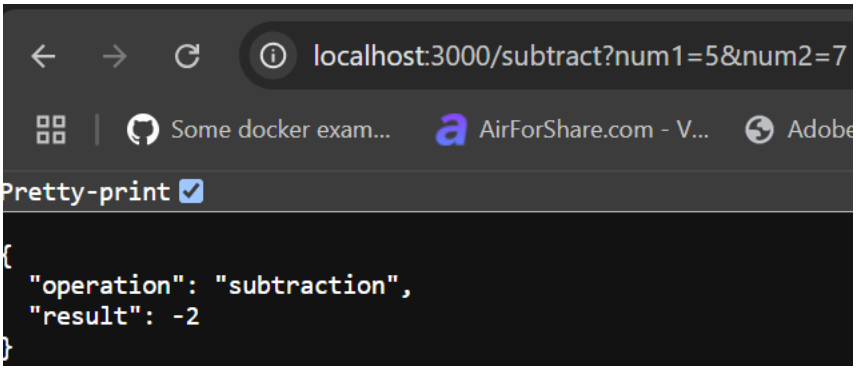


OK

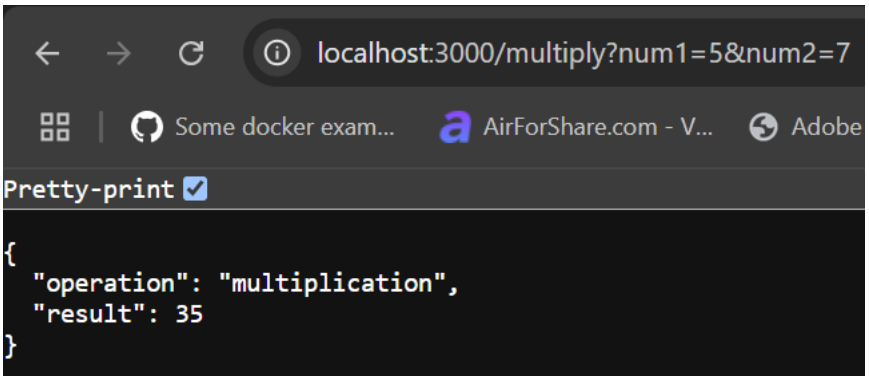
2. Add Endpoint



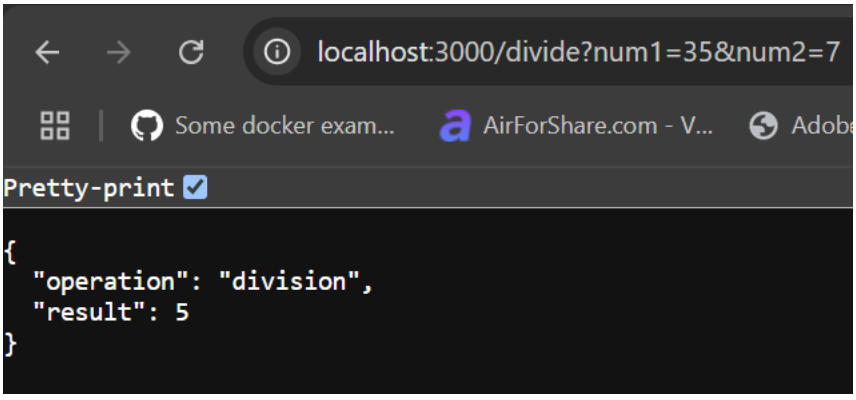
3. Subtract Endpoint



4. Multiply Endpoint



5. Divide Endpoint



6. Endpoints Summary

```
2025-05-07 14:12:57 Calculator microservice running at http://localhost:3000
2025-05-07 14:13:04 Connected to MongoDB
2025-05-07 14:12:57 (node:1) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewU
e next major version
2025-05-07 14:12:57 (Use `node --trace-warnings ...` to show where the warning was created)
2025-05-07 14:12:57 (node:1) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useU
 in the next major version
2025-05-07 14:18:24 info: Addition requested: 5 + 7 = 12 {"service":"calculator-microservice"}
2025-05-07 14:22:55 info: Subtraction requested: 5 - 7 = -2 {"service":"calculator-microservice"}
2025-05-07 14:23:30 info: Multiplication requested: 5 * 7 = 35 {"service":"calculator-microservice"}
2025-05-07 14:24:33 info: Division requested: 35 / 7 = 5 {"service":"calculator-microservice"}
```

7. Database

Whenever we hit these endpoints, their results are stored in the Mongo DB

```
test> use admin
switched to db admin
admin> db.auth('user1', '12345')
{ ok: 1 }
admin> show dbs
admin         100.00 KiB
calculatorDb   72.00 KiB
config         92.00 KiB
local          72.00 KiB
admin> use calculatorDb
switched to db calculatorDb
calculatorDb> show collections
operations
```

```
calculatorDb> db.operations.find().pretty()
[
  {
    _id: ObjectId('681adf10dfbd91b021dd4e58'),
    operation: 'addition',
    num1: 5,
    num2: 7,
    result: 12,
    timestamp: ISODate('2025-05-07T04:18:24.067Z')
  },
  {
    _id: ObjectId('681ae01fdfbd91b021dd4e59'),
    operation: 'subtraction',
    num1: 5,
    num2: 7,
    result: -2,
    timestamp: ISODate('2025-05-07T04:22:55.421Z')
  },
  {
    _id: ObjectId('681ae042dfbd91b021dd4e5a'),
    operation: 'multiplication',
    num1: 5,
    num2: 7,
    result: 35,
    timestamp: ISODate('2025-05-07T04:23:30.748Z')
  },
  {
    _id: ObjectId('681ae081dfbd91b021dd4e5b'),
    operation: 'division',
    num1: 35,
    num2: 7,
    result: 5,
    timestamp: ISODate('2025-05-07T04:24:33.536Z')
  }
]
calculatorDb>
```