Database Systems & Cloud Computing AIN3003
Course Project -Option 1

Malak Alali – 2267959

This project focuses on deploying a Python-based web application that is containerized using Docker, with Flask as the application framework and MongoDB as the database. The system leverages Kubernetes for efficient container management and orchestration. The objective of the project is to develop and deploy a scalable and reliable application that integrates seamlessly with MongoDB, demonstrating the advantages of modern containerization and orchestration technologies.

Table of Contents

1. Installation and System Setup

The tools required:

1. **Docker / docker hub:** For building container images.
2. **Kubernetes Cluster (AKS) on azure:** To run and manage the containers.
3. **PyCharm:** To run the application locally.
4. **kubectl**: Command-line tool to manage Kubernetes.
5. **Azure CLI**: Command-line tool to interact with Azure services.

2. Deploying the System on Kubernetes

2.1 Building and Pushing the Docker Image:

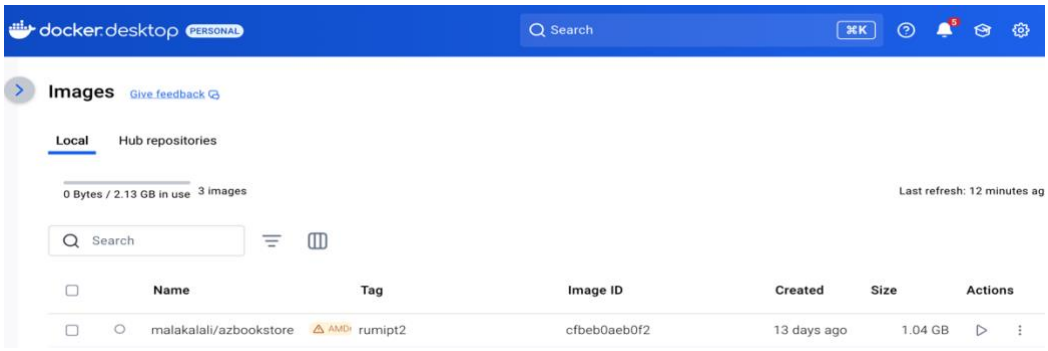
Firstly, create Dockerfile in the project directory

-Build the Docker Image

{docker build -t malakalali/azbookstore:rumipt2 }

-verify docker image by

{docker image}



-Log in to your DockerHub account

{docker login}

-Push the Image to Docker Hub

{ docker push malakalali/azbookstore:rumipt2}

```
The push refers to repository [malakalali/azbookstore:rumipt2]
96df0e5e8179: Pushed
d381bf0bfd6e: Pushed
18a4f13e5f35: Pushed
2e3e235f2dd4: Pushed
75a2bc32319e: Pushed
31d4a6b1f273: Pushed
70014bc81de0: Pushed
fd674058ff8f: Pushed
```

2.2: launching Kubernetes on azure

1- Create a cluster on azure

2- Install azure cli and kubectl

3-run the following commands

Login to your azure account

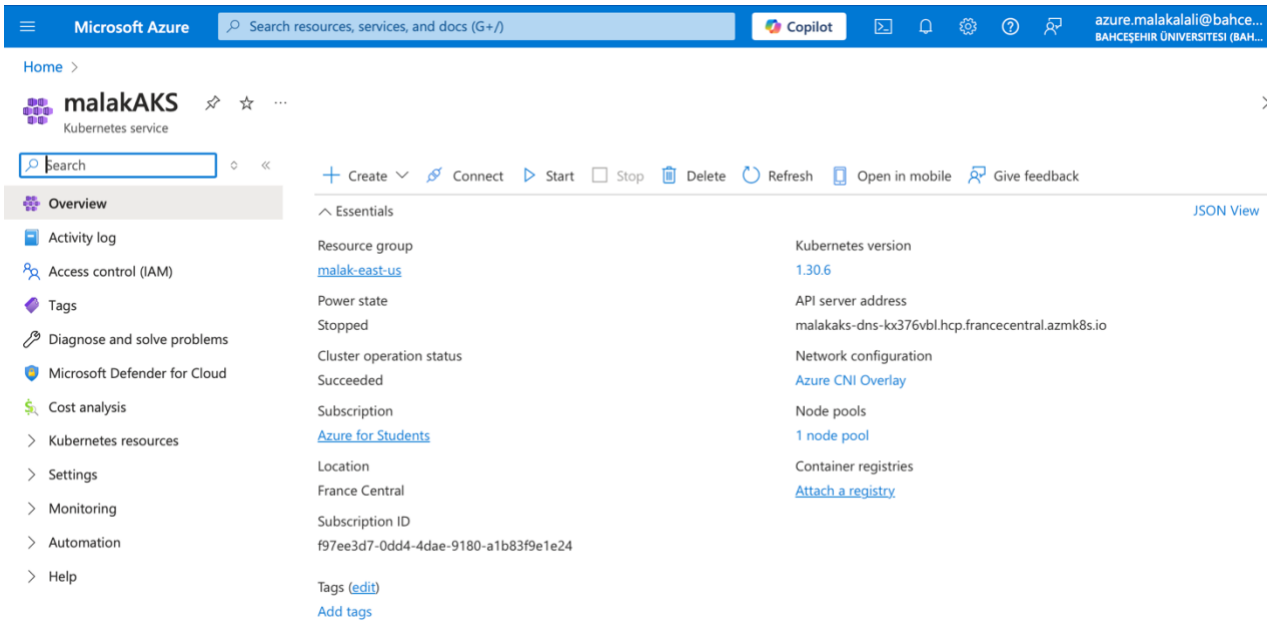
```
az login
```

Set the cluster subscription

```
az account set --subscription f97ee3d7-0dd4-4dae-9180-a1b83f9e1e24
```

Download cluster credentials

```
az aks get-credentials --resource-group malak-east-us --name malakAKS --overw...
```



2.3 Launching the Python Application

- **Apply the ConfigMap:**
- **Deploy the Flask application:**

{kubectl apply -f deployment.yaml}

- **Expose the Flask application as a service:**

{kubectl apply -f service.yaml}

- **Deploy the MongoDB StatefulSet and Service:**

{ kubectl apply -f mongodb-statefulset.yaml }

2.4 Launching the MongoDB Instance on Kubernetes

a. ensure Kubernetes cluster is running
b. kubectl is installed and configured to access the cluster.
c. The mongodb-statefulset.yaml file is present in the project directory.

3. YAML Files:

The YAML files are critical for managing MongoDB, Flask app, and Kubernetes services. for the Kubernetes resources required for deploying the system, including **StatefulSet**, **Deployment**, and other resource types.

3.1 MongoDB StatefulSet (mongodb-statefulset. yaml)

This file deploys MongoDB as a StatefulSet to ensure stable storage and unique network identity for each pod.

Service:

- Exposes MongoDB on port 27017 using a LoadBalancer for external access.
- selector.app: Selects pods with the label app: mongodb.

```yaml
apiVersion: v1
kind: Service
metadata:
  name: mongodb-service
  labels:
    app: mongodb
spec:
  ports:
  - port: 27017
    targetPort: 27017
  type: LoadBalancer
  selector:
    app: mongodb
```

3.2 Stateful Set:

- metadata.name: Name of the StatefulSet is mongodb.
- spec.serviceName: Refers to the MongoDB service name.
- spec.replicas: Number of replicas (1 for MongoDB).
- containers.image: Specifies the MongoDB Docker image.
- containers.ports.containerPort: Exposes port 27017 for MongoDB.

```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: mongodb
spec:
  serviceName: "mongodb-service"
  replicas: 1
  selector:
    matchLabels:
      app: mongodb
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      containers:
      - name: mongodb
        image: mongo
        ports:
        - containerPort: 27017
          name: mongo
```

3.4 ConfigMap.yaml

This ConfigMap stores the MongoDB connection URI and passes it as an environment variable to the Flask application.

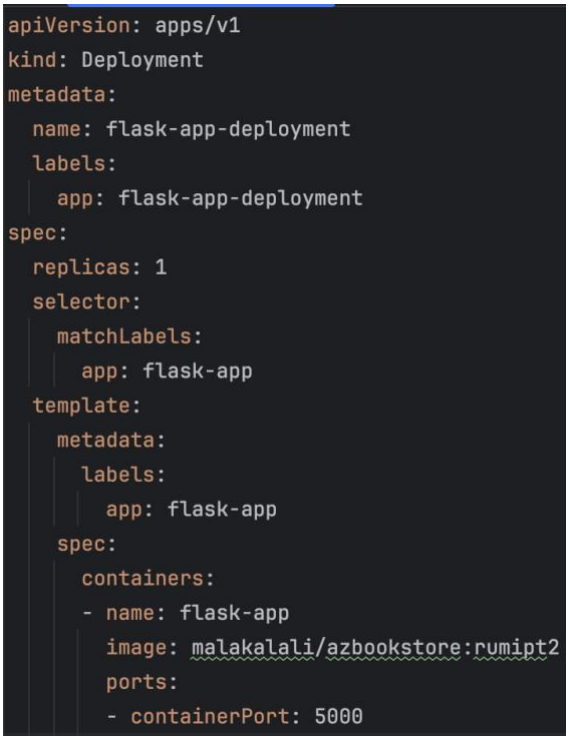
- apiVersion: Specifies the Kubernetes API version for ConfigMaps.
- kind: Indicates this resource is a ConfigMap.
- metadata.name: The name of the ConfigMap is flask-app-config.
- data: Contains key-value pairs. Here, MONGO_URI stores the MongoDB connection URI.

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: flask-app-config
data:
  MONGO_URI: "mongodb://admin:admin@mongodb-service:27017/"
```

3.5 Flask application deployment (Deployment.yaml)

This file defines the Flask application Deployment, specifying the Docker image and its configuration.

- apiVersion: Specifies the Kubernetes API version for Deployments.
- kind: Indicates this resource is a Deployment.
- metadata.name: The name of the Deployment is flask-app-deployment.
- spec. replicas: Specifies the number of replicas (pods). Here, it's set to 1.
- spec. selector.matchLabels: Ensures the pod matches the labels defined in the template.
- spec. template:
  - ¬ metadata.labels: Labels for the pod.
  - ¬ containers:
  - ¬ image: Specifies the Docker image to use
  - ¬ ports.containerPort: Exposes port 5000

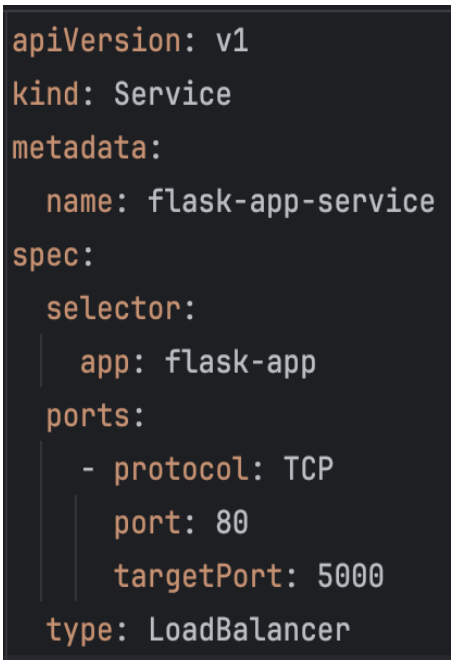


```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flask-app-deployment
  labels:
    app: flask-app-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: flask-app
  template:
    metadata:
      labels:
        app: flask-app
    spec:
      containers:
      - name: flask-app
        image: malakalali/azbookstore:rumipt2
        ports:
        - containerPort: 5000
```

3.6 Flask Application Service (service. yaml)

This Service exposes the Flask application to external traffic using a LoadBalancer. Creates an internal service named mongodb-service to allow other pods (like the Flask app) to connect to MongoDB. These YAML files can be applied to the Kubernetes cluster using kubectl apply -f <filename> and ensure a seamless deployment of the system.

- **apiVersion**: Specifies the Kubernetes API version for Services.
- **kind:** Indicates this resource is a Service.
- **metadata.name**: The name of the Service is flask-app-service.
- **spec.selector**: Selects pods with the label app: flask-app.
- **port**: The port exposed by the Service (80).
- **targetPort**: The port on the container (5000).
- **type**: LoadBalancer type to provide an external IP address.



```yaml
apiVersion: v1
kind: Service
metadata:
  name: flask-app-service
spec:
  selector:
    app: flask-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 5000
  type: LoadBalancer
```