

SIT323/SIT737- Cloud Native Application Development

10.1P: Monitoring and Visibility

Overview

The objective of this task is to demonstrate your understanding and practical skills in implementing monitoring and visibility for a cloud-native application using Google Cloud Platform (GCP) tools. You will containerize a simple application using Node.js, Docker, and Kubernetes, and then deploy it to a GCP Kubernetes cluster. You will then apply monitoring and visibility

Step 1. Install MongoDB into the Kubernetes Cluster

Command:

```
`kubectl create -f <mongo-deployment.yaml>`
```

File Name: mongo-deployment.yaml

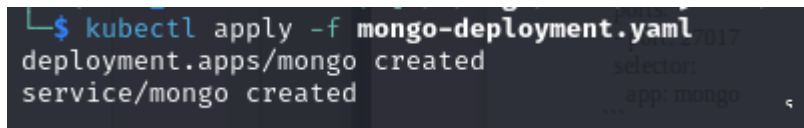
File Extension: .yaml

File Content:

```

The apply the configuration:

```
kubectl apply -f mongo-deployment.yaml
```



```
$ kubectl apply -f mongo-deployment.yaml
deployment.apps/mongo created
service/mongo created
```

### 2. Configure Persistent Storage

#### Command:

```
`kubectl create -f <mongo-pv-claim.yaml>`
```

File Name: mongo-pv-claim.yaml

File Extension: .yaml

```

Apply the configuration:

```
kubectl apply -f mongo-pv-claim.yaml
```

3. Create a Kubernetes Secret

Command:

```
`kubectl create secret generic mongo-credentials \  
  --from-literal=username=<username> \  
  --from-literal=password=<password>`
```

```
└─$ kubectl create secret generic mongo-credentials \  
  --from-literal=username=root \  
  --from-literal=password=admin  
secret/mongo-credentials created
```

4. Modify the Kubernetes Deployment Manifest

Command:

```
`kubectl apply -f <my-app-deployment.yaml>`
```

File Name: my-app-deployment.yaml

File Extension: .yaml

...

Apply the configuration:

```
kubectl apply -f my-app-deployment.yaml
```

```
└─$ kubectl apply -f my-app-deployment.yaml  
deployment.apps/my-mongo-app created
```

5. Setup Backups / Disaster Recovery

Command:

```
`kubectl create -f <mongo-backup.yaml>`
```

File Name: mongo-backup.yaml

File Extension: .yaml

File Content:

...

apiVersion: batch/v1

kind: Job

metadata:

name: mongo-backup

spec:

backoffLimit: 0

template:

```
spec:
  restartPolicy: OnFailure
  containers:
  - name: backup
    image: mongo-backup
    volumeMounts:
    - name: backup-vol
      mountPath: /data
  volumes:
  - name: backup-vol
    persistentVolumeClaim:
      claimName: pvc-mongo
...
```

Apply the configuration:
kubectl apply -f mongo-backup.yaml

```
$ kubectl apply -f mongo-backup.yaml
job.batch/mongo-backup created
name: backup-vol
mountPath: /data
```

6. Monitoring

Command:
`kubectl create -f <mongo-prometheus.yaml>`

File Name: mongo-prometheus.yaml

File Extension: .yaml

...

Apply the configuration:
kubectl apply -f mongo-prometheus.yaml

```
$ kubectl apply -f mongo-prometheus.yaml
servicemonitor.monitoring.coreos.com/mongo-monitor created
```

7. Check whether the pod is ready.

Command: kubectl get pod

NAME	READY	STATUS	RESTARTS	AGE
hello-minikube-77b6f68484-d4shz	1/1	Running	2 (6h46m ago)	2d21h
mongo-6f64b4fb45-j65df	1/1	Running	0	3h14m
mongo-backup-l5h5d	0/1	ImagePullBackOff	0	7m55s
my-mongo-app-59b64f7d9d-rz2vq	0/1	ImagePullBackOff	0	14m
prometheus-operator-6b8d85bc4c-l84b6	1/1	Running	0	4m9s

8. connect to the created application

command:

```
kubectl exec -it mongo-6f64b4fb45-j65df -- sh
```

```
$ kubectl exec -it mongo-6f64b4fb45-j65df -- sh
#
```