Lab 10 – Monitoring and Logging

Monitoring and Health Checks

Kubernetes supports monitoring applications in the form of readiness and liveness probes. Health checks can be performed on each container in a Pod. Readiness probes indicate when a Pod is “ready” to serve traffic. Liveness probes indicate a container is “alive”. If a liveness probe fails multiple times the container will be restarted. Liveness probes that continue to fail will cause a Pod to enter a crash loop. If a readiness check fails the container will be marked as not ready and will be removed from any load balancers.

In this lab you will deploy a new Pod named healthy-monolith, which is largely based on the monolith Pod with the addition of readiness and liveness probes.

In this lab you will learn how to:

- Create Pods with readiness and liveness probes
- Troubleshoot failing readiness and liveness probes

Creating Pods with Liveness and Readiness Probes

Explore the **healthy-monolith** pod configuration file:

Copy

```
cat > healthy-monolith.yaml <<EOF

apiVersion: v1

kind: Pod

metadata:

  name: "healthy-monolith"

  labels:

    app: monolith

spec:

  containers:
```

```yaml
    - name: monolith

      image: kelseyhightower/monolith:1.0.0

      ports:

        - name: http

          containerPort: 80

        - name: health

          containerPort: 81

      resources:

        limits:

          cpu: 0.2

          memory: "10Mi"

      livenessProbe:

        httpGet:

          path: /healthz

          port: 81

          scheme: HTTP

        initialDelaySeconds: 5

        periodSeconds: 15

        timeoutSeconds: 5

      readinessProbe:

        httpGet:
          path: /readiness

          port: 81

          scheme: HTTP

        initialDelaySeconds: 5

        timeoutSeconds: 1

EOF
```

Create the healthy-monolith pod using kubectl:

Copy

```
kubectl create -f healthy-monolith.yaml

projectone/musa-cluster/admin
```

Copy

```
kubectl create -f quota.yaml
```

View Pod details

Pods will not be marked ready until the readiness probe returns an HTTP 200 response. Use the kubectl describe to view details for the healthy-monolith Pod.

Copy

```
kubectl describe pods healthy-monolith
```

Experiment with Readiness Probes

you will observe how Kubernetes responds to failed readiness probes. The monolith container supports the ability to force failures of it’s readiness and liveness probes. This will enable us to simulate failures for the healthy-monolith Pod.

Copy

```
kubectl port-forward healthy-monolith 10081:81
```

Note : You know have access to the /healthz and /readiness HTTP endpoints exposed by the monolith container.

Force the monolith container readiness probe to fail. Use the curl command to toggle the readiness probe status:

Copy

```
curl http://127.0.0.1:10081/readiness/status
```

Wait about 45 seconds and get the status of the healthy-monolith Pod using the kubectl get pods command:

Copy

```
kubectl get pods healthy-monolith
```

Use the kubectl describe command to get more details about the failing readiness probe:

Copy

```
kubectl describe pods healthy-monolith
```

Notice the events for the healthy-monolith Pod report details about failing readiness probe.

Force the monolith container readiness probe to pass. Use the curl command to toggle the readiness probe status:

Copy

```
curl http://127.0.0.1:10081/readiness/status
```

Wait about 15 seconds and get the status of the healthy-monolith Pod using the kubectl get pods command:

Copy

```
kubectl get pods healthy-monolith
```

Experiment with Liveness Probes

Building on what you learned in the previous tutorial use the kubectl port-forward and curl commands to force the monolith container liveness probe to fail. Observe how Kubernetes responds to failing liveness probes.

Copy

```
kubectl port-forward healthy-monolith 10081:81
```

Copy

```
curl http://127.0.0.1:10081/healthz/status
```