Lab 3: Deploying a Circuit Breaking ambassador and a Function-as-a-Service (FaaS)

Course: SOFE 4790U
Date: 19/10/2022
Lab #: 3 (Individual Report)
Submitted by: Matheeshan Sivalingam(100703887)

Objective:

1. Learn how to create a service using NodeJS.
2. Learn how to create a Docker image.
3. Learn how to configure and use a circuit breaker using Nginx.
4. Get Familiar with FaaS.
5. Learn how to configure and use OpenFaas on GCP
6. Get Familiar with the Decorator Pattern.

Discussion:

Summarize the problem, the solution, and the requirements for the pattern given in part 1. Which of these requirements can be achieved by the procedures shown in parts 2 and 3?

The problem that the Health Endpoint Monitoring Pattern aims to solve is the inherent difficulty in monitoring and identifying factors that can affect the performance and availability of cloud-hosted services. The Health Endpoint Monitoring Pattern solves this by issuing a request to the endpoint of the application to perform the necessary checks. The application would then send a response back indicating the status of the application. The requirements necessary to implement this pattern would be a monitoring agent to send a request to the application, at least one exposed endpoint in the application and a CDN to send requests and receive the results of the check to the end user. The procedures shown in Part 2 of the lab can be implemented to achieve the necessary requirements to employ the Health Endpoint Monitoring Pattern.

Design:

Kubernetes provides persistent volumes. Why such a feature can be important? How to implement it? Provide an example in which persistent volumes are needed. Configure a YAML file to implement the example. Run it and test the creation of persistent volume and its ability to provide the required functionality within the example.

Persistent volumes are important as they allow one to retain data for long periods of time. They also allow Kubernetes applications a way to request and consume storage resources. To implement a persistent volume, one would need to configure and apply a persistent volume YAML file to the deployment. A Persistent Volume Claim would need to also be configured in order for a pod to request a volume of data.

pv.yaml

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: persistent-volume
  labels:
    type: local
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/mnt/nginx"

```

Figure 1 - Persistent Volume configuration

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pv-claim
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 3Gi
```

Figure 2 - Persistent Volume Claim configuration

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: task-pv-pod
spec:
  volumes:
    - name: task-pv-storage
      persistentVolumeClaim:
        claimName: pv-claim
  containers:
    - name: task-pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: task-pv-storage
```

Figure 3 - Pod configuration.