Faculty of Engineering and Applied Science

SOFE 4790U Distributed Systems

Lab 2: Deploying a request splitting ambassador and a load balancer with Kubernetes

Group 19 - Individual Report

William Robinson

100751756

Group GitHub Link: https://github.com/sunilt4/Distributed-Systems/tree/main/Lab%203

Procedure/Video Links:

William Robinson Video 1:
https://drive.google.com/file/d/1ZiuoMjM5MHGFoKxyObn7wKTL0Crc3Er3/view?usp=sharing

William Robinson Video 2:
https://drive.google.com/file/d/11zXKGUSViiUXZAsP5I_Nno6IgVAyVKNJ/view?usp=sharing

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 Decorator Pattern

Part 2.

Videos of the procedure can be found above. ***Video 1 is the procedure of part 2-3***

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?

Part 1.

Health Endpoint Monitoring pattern is a function as a service(Faas) system that solves the issue of when an application or a service experiences an error it will be recognized by the system and proper steps can be taken to resolve the problem. It helps keep the system on going while making sure everything works as it should. It is also capable of checking the transfer and response delays to make sure that everything is running smoothly and efficiently. One of the main requirements of this Faas is to be able to log and audit within the application. Verifying availability by monitoring the web application will guarantee the correct operation with no failures or disruptions are present. It must also be able to isolate any detected issues so that services can be returned to their original working state.

Part 2-3.

Requirements that are achieved by processes in part 2-3 involve isolating detected failures or issues. It will hold a locked state allowing the service to return to normal operation without being interrupted by user/client requests. After verifying that the correct operation is functioning the service is opened to user/client requests.

Design:

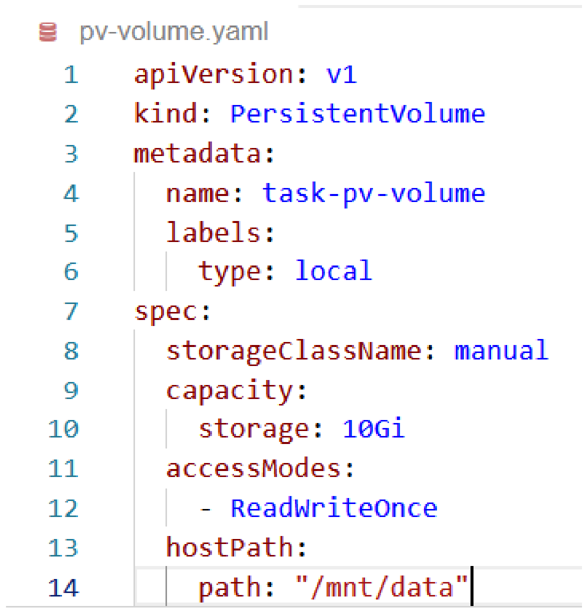
Kubernetes provides persistent volumes. Why can such a feature be important? How to implement it? Provide an example in which persistent volumes are needed. Configure a YAML file to implement the example. Run it and test the creation of persistent volume and its ability to provide the required functionality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

Why can such a feature be important?

Persistent volumes are an abstract implementation of a physical storage that is deployed within the kubernetes pods. This storage volume can remain beyond the lifetime of a pod allowing continual use. This feature can be important as if a deployment goes down or gets restarted this volume of storage remains accessible and contains all previous stored data within.

How to implement it?

Persistent volumes can be deployed by a yaml file into the deployment. An example of such yaml file can be seen below:



pv-volume.yaml

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: task-pv-volume
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/mnt/data"
```

Provide an example in which persistent volumes are needed:
Since persistent volumes remain after the lifetime of a pod they are very useful for storing any data the user wants to access and modify long into the deployment cycle. Persistent volumes are also very useful when handling a list for an order so it can be brought back to the user when they access their account.

Video example of running the persistent volume yaml can be seen in video 2