Faculty of Engineering and Applied Science

SOFE 4790U Distributed Systems

CRN 44425

Lab #3

Alexander Campbell

100703650

Introduction:

This lab gave us an introduction to deploying a circuit breaker ambassador and utilizing the Function as a Service. Below is the detailed description of what happened as I was unable to record due to completing this lab in a busy environment.

Part 2 Procedure:

First I did the initial setup of the clusters and then cloned in the required Github repository

```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ gcloud config set compute/zone northamerica-northeast1-b
Updated property [compute/zone].
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ gcloud container clusters create openfaas --num-nodes=3
Default change: VPC-native is the default mode during cluster creation for versions greater than 1.21.0-gke.1500. To create advanced routes based clusters, please pass the `--no-enable-ip-alias` flag
Default change: During creation of nodepools or autoscaling configuration changes for cluster versions greater than 1.24.1-gke.800 a default location policy is applied. For Spot and PVM it defaults to ANY, and for all other VM kinds a BALANCED policy is used. To change the default values use the `--location-policy` flag.
Note: Your Pod address range (`--cluster-ipv4-cidr`) can accommodate at most 1008 node(s).
Creating cluster openfaas in northamerica-northeast1-b... Cluster is being health-checked (master is healthy)...done.
Created [https://container.googleapis.com/v1/projects/dev-solstice-362019/zones/northamerica-northeast1-b/clusters/openfaas].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload_/gcloud/northamerica-northeast1-b/openfaas?project=dev-solstice-362019
kubeconfig entry generated for openfaas.
NAME: openfaas
LOCATION: northamerica-northeast1-b
MASTER_VERSION: 1.22.12-gke.2300
MASTER_IP: 35.203.10.22
MACHINE_TYPE: e2-medium
NODE_VERSION: 1.22.12-gke.2300
NUM_NODES: 3
STATUS: RUNNING
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ cd ~
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ git clone https://github.com/GeorgeDaoud3/SOFE4790U-lab3.git
Cloning into 'SOFE4790U-lab3'...
remote: Enumerating objects: 27, done.
remote: Counting objects: 100% (27/27), done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 27 (delta 4), reused 24 (delta 4), pack-reused 0
Receiving objects: 100% (27/27), 5.77 KiB | 2.88 MiB/s, done.
Resolving deltas: 100% (4/4), done.
```

After reading each of the specified files, server.js and Dockerfile, I ran the docker build and docker push commands

```
a_bainhewcampbell@cloudshell:~/SOFE4790U-lab3/part2/DummyServiceContainer (dev-solstice-362019)$ docker build . -t us.gcr.io/dev-solstice-362019/dummyservice
Sending build context to Docker daemon  6.656kB
Step 1/7 : FROM node:carbon
 ---> 8eeadf3757f4
Step 2/7 : WORKDIR /usr/src/app
 ---> Using cache
 ---> c0e6f06ee4e7
Step 3/7 : COPY package*.json ./
 ---> Using cache
 ---> a2a4c11682a0
Step 4/7 : RUN npm install
 ---> Using cache
 ---> 52325dc938b3
Step 5/7 : COPY . .
 ---> Using cache
 ---> 78fa91288b49
Step 6/7 : EXPOSE 80
 ---> Using cache
 ---> 8e469583b32d
Step 7/7 : CMD [ "npm", "start" ]
 ---> Using cache
 ---> 6a5134535472
Successfully built 6a5134535472
Successfully tagged us.gcr.io/dev-solstice-362019/dummyservice:latest
```

```
a_bainhewcampbell@cloudshell:~/SOFE4790U-lab3/part2/DummyServiceContainer (dev-solstice-362019)$ docker push us.gcr.io/dev-solstice-362019/dummyservice
Using default tag: latest
The push refers to repository [us.gcr.io/dev-solstice-362019/dummyservice]
6be78af59d60: Pushed
f4429c54b846: Pushed
73541d4c04cf: Pushed
7fd54dd9ee94: Pushed
423451ed44f2: Layer already exists
b2aaf85d6633: Layer already exists
88601a85ce11: Layer already exists
42f9c2f9c08e: Layer already exists
99e8bd3efaaf: Layer already exists
bee1e39d7c3a: Layer already exists
1f59a4b2e206: Layer already exists
0ca7f54856c0: Layer already exists
ebb9ae013834: Layer already exists
latest: digest: sha256:8f740fda721f7a9188bbcec513fca4d3e8808bdd754cc909f3ad7dc4db2b2289 size: 3048
```

I then edited, deployed and exposed both the dummy-deployment and the backup-deployment and confirmed that they were both running successfully

Dockerfile | dummy-deployment.yaml | backu

U-lab3 > part2 > dummy-deployment.yaml > {} spec > {}

```
spec:
  containers:
  - name: dummy-depoyment
    image: us.gcr.io/dev-solstice-362019/dummyservice
    ports:
    - containerPort: 80
    livenessProbe:
      httpGet:
        # The /alive endpoint is the one we will not to
        path: /alive
        port: 80
        scheme: HTTP
      initialDelaySeconds: 5
      periodSeconds: 10
```

Dockerfile | dummy-deployment.yaml | backup-deployment.yaml

b3 > part2 > backup-deployment.yaml > ...

```
  run: backup-deployment
ec:
 containers:
 - name: backup-depoyment
   image: us.gcr.io/dev-solstice-362019/dummyservice ### ToDo: fill the valu
   ports:
   - containerPort: 80
   livenessProbe:
     httpGet:
       # The /alive endpoint is the one we will not touch in our test case,
       path: /alive
       port: 80
       scheme: HTTP
     initialDelaySeconds: 5
     periodSeconds: 10
```

```
a_bainhewcampbell@cloudshell:~/SOFE4790U-lab3/part2 (dev-solstice-362019)$ kubectl create -f dummy-deployment.yaml
deployment.apps/dummy-deployment created
a_bainhewcampbell@cloudshell:~/SOFE4790U-lab3/part2 (dev-solstice-362019)$ kubectl expose deployment dummy-deployment --port=80 --type=LoadBalancer --name dummy-deployment
service/dummy-deployment exposed
```

```
a_bainhewcampbell@cloudshell:~/SOFE4790U-lab3/part2 (dev-solstice-362019)$ kubectl create -f backup-deployment.yaml
deployment.apps/backup-deployment created
a_bainhewcampbell@cloudshell:~/SOFE4790U-lab3/part2 (dev-solstice-362019)$ kubectl expose deployment backup-deployment --port=80 --type=LoadBalancer --name backup-deployment
service/backup-deployment exposed
```

```
a_bainhewcampbell@cloudshell:~/SOFE4790U-lab3/part2 (dev-solstice-362019)$ kubectl get pods
NAME                                 READY   STATUS    RESTARTS   AGE
backup-deployment-7c8b946bdf-m54r2   1/1     Running   0          61s
dummy-deployment-687f88ddb5-vql47    1/1     Running   0          3m30s
a_bainhewcampbell@cloudshell:~/SOFE4790U-lab3/part2 (dev-solstice-362019)$ kubectl get deployments
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
backup-deployment   1/1     1            1           67s
dummy-deployment    1/1     1            1           3m36s
a_bainhewcampbell@cloudshell:~/SOFE4790U-lab3/part2 (dev-solstice-362019)$ kubectl get services
NAME                TYPE           CLUSTER-IP     EXTERNAL-IP    PORT(S)        AGE
backup-deployment   LoadBalancer   10.80.7.2      35.203.34.60   80:30105/TCP   57s
dummy-deployment    LoadBalancer   10.80.0.161    34.95.41.181   80:30437/TCP   3m17s
kubernetes          ClusterIP      10.80.0.1      <none>         443/TCP        16m
```

The circuit breaker was then configured and deployed

```
a_bainhewcampbell@cloudshell:~/SOFE4790U-lab3/part2 (dev-solstice-362019)$ kubectl create -f nginx-configmap.yaml
configmap/nginx-configuration created
a_bainhewcampbell@cloudshell:~/SOFE4790U-lab3/part2 (dev-solstice-362019)$ kubectl create -f circuitbreaker.yaml
deployment.apps/circuitbreaker created
```

```
a_bainhewcampbell@cloudshell:~/SOFE4790U-lab3/part2 (dev-solstice-362019)$ kubectl get services
NAME                TYPE           CLUSTER-IP     EXTERNAL-IP    PORT(S)        AGE
backup-deployment   LoadBalancer   10.80.7.2      35.203.34.60   80:30105/TCP   3m2s
circuitbreaker      LoadBalancer   10.80.4.213    35.203.65.90   80:30385/TCP   74s
dummy-deployment    LoadBalancer   10.80.0.161    34.95.41.181   80:30437/TCP   5m22s
kubernetes          ClusterIP      10.80.0.1      <none>         443/TCP        18m
```

The circuit breaker was then tested

```
a_bainhewcampbell@cloudshell:~/SOFE4790U-lab3/part2 (dev-solstice-362019)$ curl -v http://35.203.65.90
*   Trying 35.203.65.90:80...
* Connected to 35.203.65.90 (35.203.65.90) port 80 (#0)
> GET / HTTP/1.1
> Host: 35.203.65.90
> User-Agent: curl/7.74.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Server: nginx/1.13.7
< Date: Tue, 18 Oct 2022 17:54:14 GMT
< Content-Type: text/html; charset=utf-8
< Content-Length: 27
< Connection: keep-alive
< X-Powered-By: Express
< ETag: W/"1b-y2OQ2D360g9elBAWWrTIhINssik"
<
* Connection #0 to host 35.203.65.90 left intact
SOMERESPONSE FROM 10.76.1.7a_bainhewcampbell@cloudshell:~/SOFE4790U-lab3/part2 (dev-solstice-362019)$ curl -v http://35.203.65.90
*   Trying 35.203.65.90:80...
* Connected to 35.203.65.90 (35.203.65.90) port 80 (#0)
> GET / HTTP/1.1
> Host: 35.203.65.90
> User-Agent: curl/7.74.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Server: nginx/1.13.7
< Date: Tue, 18 Oct 2022 17:54:16 GMT
< Content-Type: text/html; charset=utf-8
< Content-Length: 27
< Connection: keep-alive
< X-Powered-By: Express
< ETag: W/"1b-y2OQ2D360g9elBAWWrTIhINssik"
<
* Connection #0 to host 35.203.65.90 left intact
SOMERESPONSE FROM 10.76.1.7a_bainhewcampbell@cloudshell:~/SOFE4790U-lab3/part2 (dev-solstice-362019)$ curl -v http://35.203.65.90
*   Trying 35.203.65.90:80...
* Connected to 35.203.65.90 (35.203.65.90) port 80 (#0)
> GET / HTTP/1.1
```

```
<
* Connection #0 to host 35.203.65.90 left intact
```

```
SOMERESPONSE FROM 10.76.1.7a_bainhewcampbell@cloudshell:~/SOFE4790U-lab3/part2 (dev-solstice-362019)$ curl -d "" -s -D - http://34.95.41.181/fakeerrormodern
HTTP/1.1 404 Not Found
X-Powered-By: Express
Content-Security-Policy: default-src 'none'
X-Content-Type-Options: nosniff
Content-Type: text/html; charset=utf-8
Content-Length: 155
Date: Tue, 18 Oct 2022 17:55:06 GMT
Connection: keep-alive

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Error</title>
</head>
<body>
<pre>Cannot POST /fakeerrormodern</pre>
</body>
</html>
a_bainhewcampbell@cloudshell:~/SOFE4790U-lab3/part2 (dev-solstice-362019)$ curl -v http://35.203.65.90
*   Trying 35.203.65.90:80...
* Connected to 35.203.65.90 (35.203.65.90) port 80 (#0)
> GET / HTTP/1.1
> Host: 35.203.65.90
> User-Agent: curl/7.74.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Server: nginx/1.13.7
< Date: Tue, 18 Oct 2022 17:55:19 GMT
< Content-Type: text/html; charset=utf-8
< Content-Length: 27
```

The error was reset and curl was run one last time

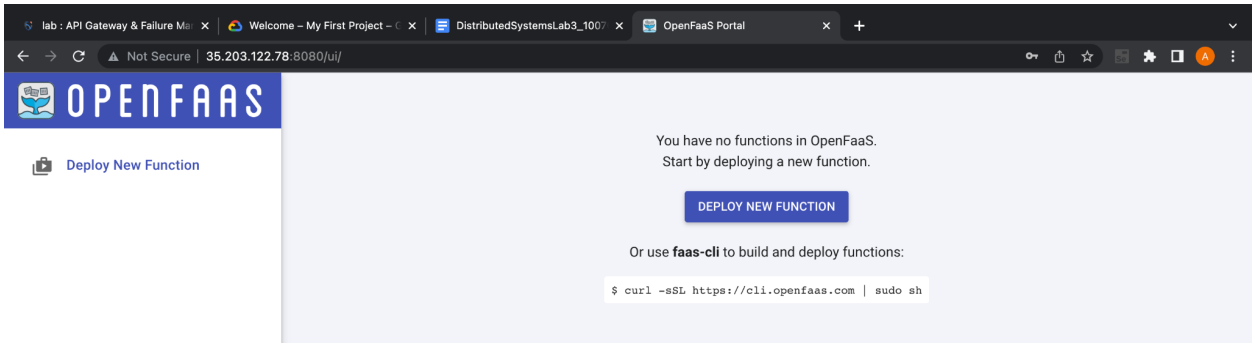
```
SOMERESPONSE FROM 10.76.1.7a_bainhewcampbell@cloudshell:~/SOFE4790U-lab3/part2 (dev-solstice-362019)$ curl -d "" -s -D - http://34.95.41.181/fakeerrormodern
HTTP/1.1 404 Not Found
X-Powered-By: Express
Content-Security-Policy: default-src 'none'
X-Content-Type-Options: nosniff
Content-Type: text/html; charset=utf-8
Content-Length: 155
Date: Tue, 18 Oct 2022 17:55:30 GMT
Connection: keep-alive

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Error</title>
</head>
<body>
<pre>Cannot POST /fakeerrormodern</pre>
</body>
</html>
a_bainhewcampbell@cloudshell:~/SOFE4790U-lab3/part2 (dev-solstice-362019)$ curl -v http://35.203.65.90
*   Trying 35.203.65.90:80...
* Connected to 35.203.65.90 (35.203.65.90) port 80 (#0)
> GET / HTTP/1.1
> Host: 35.203.65.90
> User-Agent: curl/7.74.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Server: nginx/1.13.7
< Date: Tue, 18 Oct 2022 17:55:33 GMT
< Content-Type: text/html; charset=utf-8
< Content-Length: 27
< Connection: keep-alive
< X-Powered-By: Express
< ETag: W/"1b-y2OQ2D360g9elBAWWrTIhINssik"
<
* Connection #0 to host 35.203.65.90 left intact
```

Part 3 Procedure:

First I created the cluster admin role binding, deployed OpenFass to GKE, installed, the cli and verified that OpenFaas has started

```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl -n openfaas get deployments -l "release=openfaas, app=openfaas"
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
alertmanager       1/1     1            1           34s
basic-auth-plugin  1/1     1            1           34s
gateway            1/1     1            1           34s
nats               1/1     1            1           34s
prometheus         1/1     1            1           34s
queue-worker       1/1     1            1           34s
```

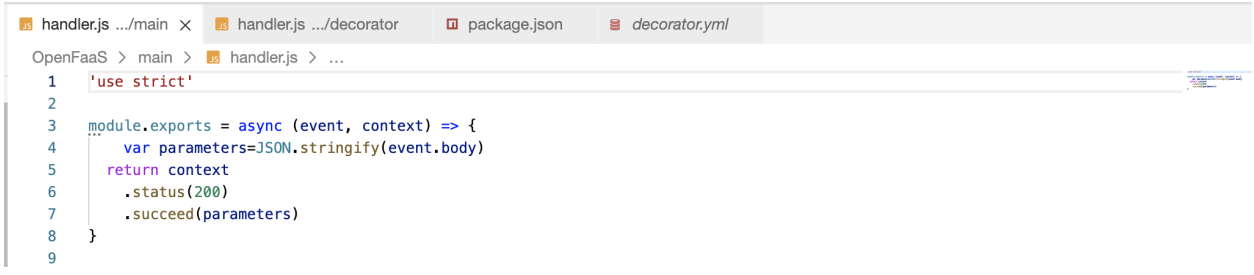
I then logged into OpenFaas and deployed the first function after updating main/handler.js with the specified code




```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl rollout status -n openfaas deploy/gateway
deployment "gateway" successfully rolled out
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl get svc -o wide gateway-external -n openfaas
NAME               TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)          AGE   SELECTOR
gateway-external   LoadBalancer   10.80.11.194   35.203.122.78   8080:32308/TCP   69s   app=gateway
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ export OPENFAAS_URL="35.203.122.78:8080"
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ PASSWORD=$(kubectl get secret -n openfaas basic-auth -o jsonpath="{.data.basic-auth-password}" | base64 --decode; echo)
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ echo $PASSWORD
p4mQ6wxF31r5GI8frM7Z36yM7
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ echo -n $PASSWORD | faas-cli login --username admin --password-stdin
Calling the OpenFaaS server to validate the credentials...
WARNING! You are not using an encrypted connection to the gateway, consider using HTTPS.
credentials saved for admin http://35.203.122.78:8080
```

```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ mkdir ~/OpenFaaS
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ cd ~/OpenFaaS
a_bainhewcampbell@cloudshell:~/OpenFaaS (dev-solstice-362019)$ faas-cli template pull
Fetch templates from repository: https://github.com/openfaas/templates.git at
2022/10/18 18:27:01 Attempting to expand templates from https://github.com/openfaas/templates.git
2022/10/18 18:27:02 Fetched 17 template(s) : [csharp dockerfile go java11 java11-vert-x node node12 node12-debian node14 node16 node17 php7 php8 python python3 python3-debian rub
 from https://github.com/openfaas/templates.git
a_bainhewcampbell@cloudshell:~/OpenFaaS (dev-solstice-362019)$ faas-cli new --list
Languages available as templates:
- csharp
- dockerfile
- go
```

```
a_bainhewcampbell@cloudshell:~/OpenFaaS (dev-solstice-362019)$ faas-cli new --lang node12 --prefix us.gcr.io/dev-solstice-362019 main
Folder: main created.
```



```
a_bainhewcampbell@cloudshell:~/OpenFaaS (dev-solstice-362019)$ faas-cli build -f main.yml
[0] > Building main.
Clearing temporary build folder: ./build/main/
Preparing: ./main/ build/main/function
Building: us.gcr.io/dev-solstice-362019/main:latest with node12 template. Please wait..
Sending build context to Docker daemon   12.8kB
Step 1/31 : FROM --platform=${TARGETPLATFORM:-linux/amd64} ghcr.io/openfaas/of-watchdog:0.9.8 as watchdog
0.9.8: Pulling from openfaas/of-watchdog
ecf1c50101a3: Pulling fs layer
ecf1c50101a3: Verifying Checksum
ecf1c50101a3: Download complete
ecf1c50101a3: Pull complete
```

```
a_bainhewcampbell@cloudshell:~/OpenFaaS (dev-solstice-362019)$ faas-cli deploy -f main.yml
Deploying: main.
WARNING! You are not using an encrypted connection to the gateway, consider using HTTPS.

Deployed. 202 Accepted.
URL: http://35.203.122.78:8080/function/main
```

```
a_bainhewcampbell@cloudshell:~/OpenFaaS (dev-solstice-362019)$ curl http://35.203.122.78:8080/function/main -H 'Content-Type: application/json' -d '{ "Name": "Square", "Color": "Red", "Dimensions": 2 }'
{"Name":"Square","Color":"Red","Dimensions":2}a_bainhewcampbell@cloudshell:~/OpenFaaS (dev-solstice-362019)$
```

I then defined the decorator logic as specified in the instructions. It took a few tries, which explains why the decorator exists and many of the layers have already been built. I started out by modifying the decorator file and building it as specified.

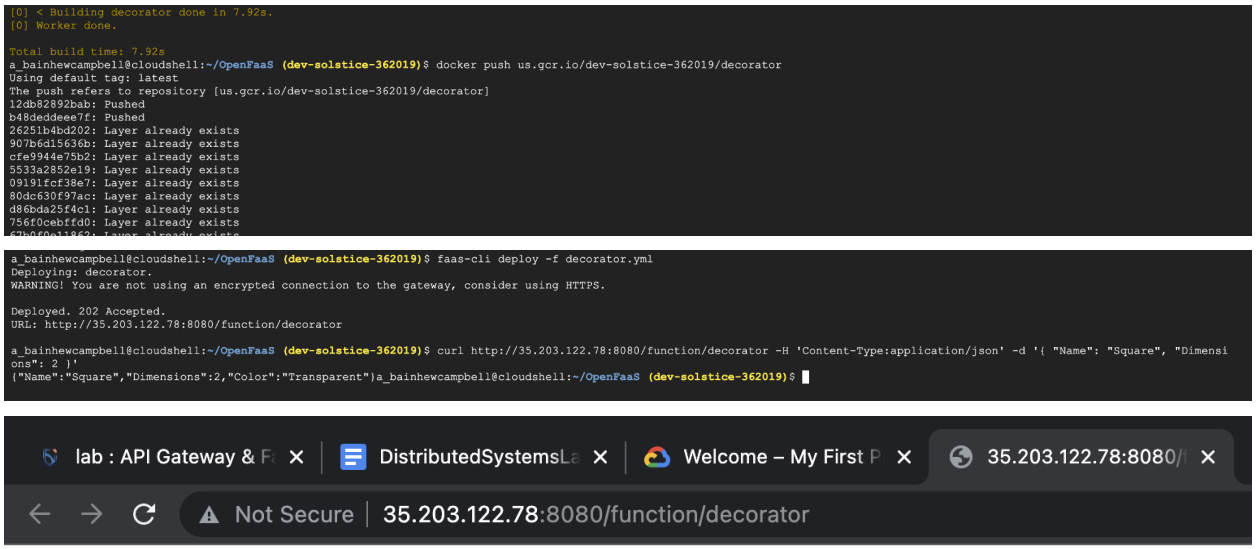
```
a_bainhewcampbell@cloudshell:~/OpenFaaS (dev-solstice-362019)$ faas-cli new --lang node12 --prefix us.gcr.io/dev-solstice-362019 decorator
Folder: decorator already exists
a_bainhewcampbell@cloudshell:~/OpenFaaS (dev-solstice-362019)$ faas-cli build -f decorator.yml
[0] > Building decorator.
Clearing temporary build folder: ./build/decorator/
Preparing: ./decorator/ build/decorator/function
Building: us.gcr.io/dev-solstice-362019/decorator:latest with node12 template. Please wait..
Sending build context to Docker daemon  13.31kB
Step 1/31 : FROM --platform=${TARGETPLATFORM:-linux/amd64} ghcr.io/openfaas/of-watchdog:0.9.8 as watchdog
 ---> 486f87f58f79
Step 2/31 : FROM --platform=${TARGETPLATFORM:-linux/amd64} node:12-alpine as ship
 ---> bb6d28039b8c
Step 3/31 : ARG TARGETPLATFORM
 ---> Using cache
 ---> 67f8744e453e
Step 4/31 : ARG BUILDPLATFORM
 ---> Using cache
 ---> aafa8fbe521a
Step 5/31 : COPY --from=watchdog /fwatchdog /usr/bin/fwatchdog
 ---> Using cache
 ---> b9aebb71772e
Step 6/31 : RUN chmod +x /usr/bin/fwatchdog
 ---> Using cache
 ---> bb714f6e4193
Step 7/31 : RUN apk --no-cache add curl ca-certificates     && addgroup -S app && adduser -S -g app app
```

handler.js .../main | handler.js .../decorator | package.json | decorator.yml

OpenFaaS > decorator > handler.js > <unknown> > ...

```
1   'use strict'
2   const request = require('sync-request');
3
4   module.exports = async (event, context) => {
5       var obj = event.body;
6       if (obj['Name'] === undefined) {
7           obj['Name'] = 'Nameless';
8       }
9       if (obj['Color'] === undefined) {
10          obj['Color'] = 'Transparent';
11      }
12      var res = request('POST', 'http://35.203.122.78:8080/function/main', {
13          body: JSON.stringify(obj)
14      });
15      console.log(res["body"].toString())
16      return context.status(200).succeed(res["body"].toString('utf8', 1, res["body"].length-
17  1).replace(/\\\"/g,'\"'));
18  }
```

I then deployed the YAML file using the faas-cli and verified it was running by using the curl command and visiting the URL



{"Name":"Nameless","Color":"Transparent"}

Discussion

In any system it's extremely important to monitor its health and statistics, such as latency, to ensure that it runs correctly and maintains high availability. In distributed systems, it can be hard to do so as the physical hardware associated with the system may not be available to be monitored. To monitor the health of a distributed system we can implement the Health Endpoint Monitoring Pattern. This pattern involves polling the system of interest with requests to gather information about the current state of the application. After receiving a response from the system in question, the receiver will validate the response code to ensure regular functionality. To implement this pattern there are several required components: An application that can perform health checks when polled, an agent that will perform the poll and validate the response code from the polled application, measure its response time, or check for any certificate expirations.

In Part 2, we implemented a service, a backup service, and a circuit breaker for the two services. The circuit breaker acts as a health check endpoint that will check every 3 seconds, and reroute the request to the backup if it's not considered healthy. In Part 3, we implemented a function using the Decorator pattern that added a function that transforms the values sent to this. This is part of the Health Endpoint Monitoring as this /decorator endpoint will verify the content of a response to detect any errors and fix them if it's missing any parameters.

Design

In Kubernetes, there are storage structures known as ‘Persistent Volumes’ (PVs). These volumes act as a method of storage that abstracts away the storage details of data from how the data is consumed and read. These PVs can be provisioned either by an administrator of the system or automatically using Storage classes. These are extremely important for storing information and increasing its availability. We can combine these PVs with some data management system such as MySQL in order to create a full and effective system.

To better understand Persistent Volumes, I followed a Kubernetes guide to creating and accessing one. It first started with creating a single node with minikube and entering the node.

```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ minikube ssh
Last login: Tue Oct 18 19:51:51 2022 from 192.168.49.1
```

After successfully entering the node, I created an html file that contains a string “Hello from Kubernetes Storage”. After exiting, I created a persistent volume, a persistent volume claim, and a pod that can utilize the persistent volume.

```
root@minikube:~# sudo mkdir /mnt/data
root@minikube:~# sudo sh -c "echo 'Hello from Kubernetes storage' > /mnt/data/index.html"
root@minikube:~# cat /mnt/data/index.html
Hello from Kubernetes storage
root@minikube:~# exit
logout
docker@minikube:~$ exit
logout
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl apply -f https://k8s.io/examples/pods/storage/pv-volume.yaml
persistentvolume/task-pv-volume created
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl apply -f https://k8s.io/examples/pods/storage/pv-claim.yaml
persistentvolumeclaim/task-pv-claim created
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl get pvc task-pv-claim
NAME            STATUS   VOLUME           CAPACITY   ACCESS MODES   STORAGECLASS   AGE
task-pv-claim   Bound    task-pv-volume   10Gi       RWO            manual         11s
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl apply -f https://k8s.io/examples/pods/storage/pv-pod.yaml
pod/task-pv-pod created
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl get pod task-pv-pod
NAME          READY   STATUS    RESTARTS   AGE
task-pv-pod   1/1     Running   0          7s
```

```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"
```

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

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: task-pv-pod
spec:
  volumes:
    - name: task-pv-storage
      persistentVolumeClaim:
        claimName: task-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
```

From here, we could reenter the node, install curl and other necessary software, and check if the persistent volume is being utilized and serving the index.html file that was created earlier in the node.

```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl exec -it task-pv-pod -- /bin/bash
root@task-pv-pod:/# apt update
Get:1 http://deb.debian.org/debian bullseye InRelease [116 kB]
Get:2 http://deb.debian.org/debian-security bullseye-security InRelease [48.4 kB]
Get:3 http://deb.debian.org/debian bullseye-updates InRelease [44.1 kB]
Get:4 http://deb.debian.org/debian bullseye/main amd64 Packages [8184 kB]
Get:5 http://deb.debian.org/debian-security bullseye-security/main amd64 Packages [190 kB]
Get:6 http://deb.debian.org/debian bullseye-updates/main amd64 Packages [6340 B]
Fetched 8588 kB in 2s (4941 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
1 package can be upgraded. Run 'apt list --upgradable' to see it.
root@task-pv-pod:/# apt install curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
curl is already the newest version (7.74.0-1.3+deb11u3).
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
root@task-pv-pod:/# curl http://localhost/
Hello from Kubernetes storage
root@task-pv-pod:/#
```