Faculty of Engineering & Applied Science

# SOFE4790U – Distributed Systems

## Group Work

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

## Due Date: 10/30/2022

| First Name | Last Name | Student ID |
|---|---|---|
| Abdul | Bhutta | 100785884 |
| Alexander | Campbell | 100703650 |
| Mamun | Hossain | 100553073 |
| Owen | Musselman | 100657709 |

## Part 1 - Health Endpoint Monitoring Pattern

### Problem

One of the business requirements is to periodically check for status or monitor the applications and backend services to ensure they are functioning without any issues. It is much easier to monitor with a centralized physical system than a cloud system because of having complete control of the physical system.

### Solution

The problem can be solved through a health monitoring system which checks the status of the components of the system, which can be implemented at the endpoint of the application and returns the status through a response code. Below is a list of requirements to implement the pattern.

| Requirements | Description |
|---|---|
| **Agent** | Analyzes the outcome from the application which performed the health check |
| **Application** | Check the health status of the request at the endpoint |

Below is a list of checks that these services/tools perform:

- **Measuring response times:** The sum of the network latency, and the time that the service or application performs the request. If the sum is high, then there could be a potential problem on the network side (load balancing, traffic), or unoptimized usage of message transfer protocols.

- **Validating the response code:** Ensure that the correct response code is returning like 200 for OK.

- **Checking the content of the response:** Checking if the returned value of the response code 200 (OK) or something else. Also if the value is 20 it will still check other aspects of the page that will verify if the returned webpage is correct like a test phrase on the page.

- **Checking resources/services:** Checks the content delivery NW that is utilized to send content from global caches.

- **Checking expiration of SSL certs:** Checks if the SSL certificate is expired or not.

- **Measuring response time of DNS lookup:** Looks up the URL for the service/application being used to see if there is DNS failure, or check the response time.

- **Validating URL from the DNS lookup response:** To stop malicious redirections of requests if there is an attack aimed at the DNS server.

## Part 2 – Circuit Breaker Ambassador

First, I did the initial setup of the clusters and then cloned them 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-en
able-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 an
d 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 | backup

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

The circuit breaker was then tested

```
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
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 – Decorator Pattern

*Create a cluster admin role binding*

```
bhutta_abdul@cloudshell:~/SOFE4790U-lab3/part3 (sofe4790u-lab3)$ kubectl create clusterrolebinding "cluster-admin-$(whoami)" --clusterrole=cluster-admin --user
="$(gcloud config get-value core/account)"
Your active configuration is: [cloudshell-25206]
clusterrolebinding.rbac.authorization.k8s.io/cluster-admin-bhutta_abdul created
```

*Deploy OpenFaas to GKE*

```
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ curl -SLsf https://dl.get-arkade.dev/ | sudo sh
arkade install openfaas --load-balancer
x86_64
Downloading package https://github.com/alexellis/arkade/releases/download/0.8.50/arkade as /tmp/arkade
Download complete.

Running with sufficient permissions to attempt to move arkade to /usr/local/bin
New version of arkade installed to /usr/local/bin
Creating alias 'ark' for 'arkade'.

Open Source Marketplace For Developer Tools
```

*Install Client*

```
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ curl -SLsf https://cli.openfaas.com | sudo sh
Finding latest version from GitHub
0.14.11
Downloading package https://github.com/openfaas/faas-cli/releases/download/0.14.11/faas-cli as /tmp/faas-cli
Download complete.

Running with sufficient permissions to attempt to move faas-cli to /usr/local/bin
New version of faas-cli installed to /usr/local/bin
Creating alias 'faas' for 'faas-cli'.

CLI:
 commit:  8820d8e4a15dab900d8a7e8fc271851ccb94012e
 version: 0.14.11
```

*Verify OpenFaas is working*

```
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ kubectl -n openfaas get deployments -l "release=openfaas, app=openfaas"
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
alertmanager        1/1     1            1           6m45s
basic-auth-plugin   1/1     1            1           6m45s
gateway             1/1     1            1           6m45s
nats                1/1     1            1           6m45s
prometheus          1/1     1            1           6m45s
queue-worker        1/1     1            1           6m45s
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$
```

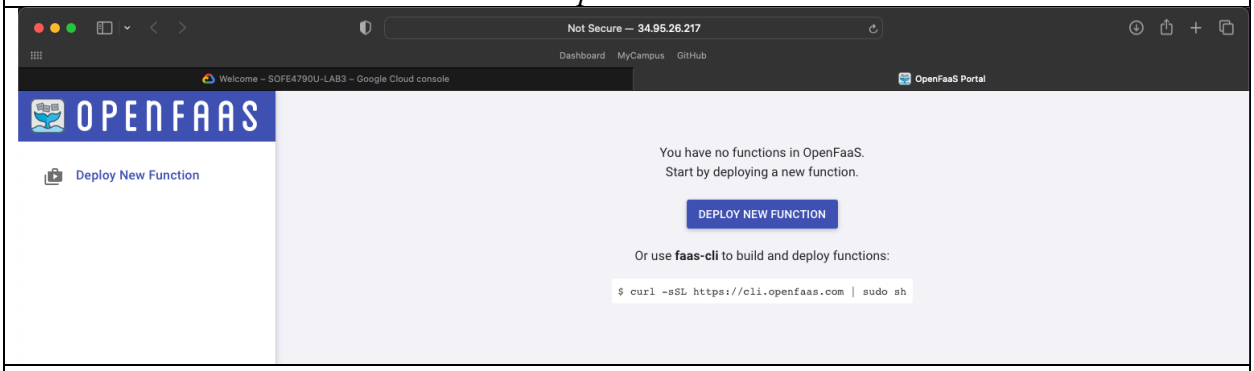
*Check OpenFaas is ready and get IP*

```
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ kubectl rollout status -n openfaas deploy/gateway
deployment "gateway" successfully rolled out
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ kubectl get svc -o wide gateway-external -n openfaas
NAME               TYPE           CLUSTER-IP    EXTERNAL-IP     PORT(S)          AGE    SELECTOR
gateway-external   LoadBalancer   10.80.5.210   34.95.26.217    8080:32273/TCP   7m33s  app=gateway
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$
```

*Save Login and password*

```
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ export OPENFAAS_URL="34.95.26.217:8080"
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ PASSWORD=$(kubectl get secret -n openfaas basic-auth -o jsonpath="{.data.basic-auth-password}" | base64 --decode; echo
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ echo $PASSWORD
9u248zD4208eW7dnHakhDkOf5
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ echo -n $PASSWORD | faas-cli login --username admin --password-stdin
Calling the OpenFaaS server to validate the credentials...
WARNING! You are not using an encrypted connection to the gateway, consider using HTTPS.
credentials saved for admin http://34.95.26.217:8080
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$
```

*Open Faas*




*Create new directory and pull all available templates*

```
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ mkdir ~/OpenFaaS
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ cd ~/OpenFaaS
bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$ faas-cli template pull
Fetch templates from repository: https://github.com/openfaas/templates.git at
2022/10/30 00:40:07 Attempting to expand templates from https://github.com/openfaas/templates.git
2022/10/30 00:40:08 Fetched 17 template(s) : [csharp dockerfile go java11 java11-vert-x node node12 node12-debian node14 node16 node17 php7 php8 python pyth
on3 python3-debian ruby] from https://github.com/openfaas/templates.git
bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$ faas-cli new --list
Languages available as templates:
- csharp
- dockerfile
- go
- java11
- java11-vert-x
- node
- node12
- node12-debian
- node14
- node16
- node17
- php7
- php8
- python
- python3
- python3-debian
- ruby

bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$
```

*Empty NodeJS function*

```
bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$ faas-cli new --lang node12 --prefix us.gcr.io/sofe4790u-lab3 main
Folder: main created.
 ___                   _____           ____
/ _ \ _ __   ___ _ __ |  ___|_ _  __ _/ ___|
| | | | '_ \ / _ \ '_ \| |_ / _` |/ _` \___ \
| |_| | |_) |  __/ | | |  _| (_| | (_| |___) |
 \___/| .__/ \___|_| |_|_|  \__,_|\__,_|____/
      |_|


Function created in folder: main
Stack file written: main.yml

Notes:
You have created a new function which uses Node.js 12.

npm i --save can be used to add third-party packages like request or cheerio
npm documentation: https://docs.npmjs.com/

Unit tests are run at build time via "npm run", edit package.json to specify
how you want to execute them.

bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$
```

*Update handler.js file*

OpenFaaS > main > handler.js > ...

```
1  'use strict'
2
3  module.exports = async (event, context) => {
4      var parameters=JSON.stringify(event.body)
5      return context
6      .status(200)
7      .succeed(parameters)
8  }
```

*Build Docker image*

```
bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$ cd ~/OpenFaaS
bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$ faas-cli build -f main.yml
[0] > Building main.
Clearing temporary build folder: ./build/main/
Preparing: ./main/ build/main/function
Building: us.gcr.io/sofe4790u-lab3/main:latest with node12 template. Please wait..
Sending build context to Docker daemon   12.8kB
Step 1/31 : FROM --platform=${TARGETPLATFORM:-linux/amd64} ghcr.io/openfaas/of-watchdog:0.9.10 as watchdog
0.9.10: Pulling from openfaas/of-watchdog
c4fc21d17d12: Pulling fs layer
c4fc21d17d12: Download complete
c4fc21d17d12: Pull complete
Digest: sha256:5d1ed766546ff5510614c695d48e3e0bafefee01ec8c04dc3a51297cb75bb57a
Status: Downloaded newer image for ghcr.io/openfaas/of-watchdog:0.9.10
 ---> 9f97468a4531
Step 2/31 : FROM --platform=${TARGETPLATFORM:-linux/amd64} node:12-alpine as ship
12-alpine: Pulling from library/node
```

*Push to container*

```
bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$ docker push us.gcr.io/sofe4790u-lab3/main
Using default tag: latest
The push refers to repository [us.gcr.io/sofe4790u-lab3/main]
c4cc512dfbdb: Pushed
f3e52fa4f0ee: Pushed
c17d15683f0c: Pushed
4ce34a584422: Pushed
9a941d3d77c7: Pushed
6229463858ce: Pushed
57021fbca4bc: Pushed
2900e05f5c62: Pushed
d86bda25f4c1: Pushed
5ee04ae9a725: Pushed
4c5614929cb1: Pushed
7f30cde3f699: Layer already exists
fe810f5902cc: Layer already exists
dfd8c046c602: Layer already exists
4fc242d58285: Layer already exists
latest: digest: sha256:28f6b6df3fcb179a238b0be33a1b5c8479b09efbcdf5a1ed8a136e106d3ac253 size: 3659
```

*Deploy to OpenFaaS*

```
bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$ 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://34.95.26.217:8080/function/main
```

*Send JSON object*

```
bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$ curl http://34.95.26.217:8080/function/main -H 'Content-Type: application/json' -d '{ "Name": "Square", "Color": "Red",
 "Dimensions": 2 }'
{"Name":"Square","Color":"Red","Dimensions":2}bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$
```

*Create new empty NodeJS function*

```
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ cd ~/OpenFaaS
bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$ faas-cli new --lang node12 --prefix us.gcr.io/sofe4790u-lab3 decorator
Folder: decorator created.

Function created in folder: decorator
Stack file written: decorator.yml

Notes:
You have created a new function which uses Node.js 12.

npm i --save can be used to add third-party packages like request or cheerio
npm documentation: https://docs.npmjs.com/

Unit tests are run at build time via "npm run", edit package.json to specify
how you want to execute them.

bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$ cd ~/OpenFaaS
bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$ faas-cli build -f decorator.yml
[0] > Building decorator.
Clearing temporary build folder: ./build/decorator/
Preparing: ./decorator/ build/decorator/function
Building: us.gcr.io/sofe4790u-lab3/decorator:latest with node12 template. Please wait..
Sending build context to Docker daemon   12.8kB
Step 1/31 : FROM --platform=${TARGETPLATFORM:-linux/amd64} ghcr.io/openfaas/of-watchdog:0.9.10 as watchdog
0.9.10: Pulling from openfaas/of-watchdog
c4fc21d17d12: Pulling fs layer
c4fc21d17d12: Verifying Checksum
c4fc21d17d12: Download complete
c4fc21d17d12: Pull complete
Digest: sha256:5d1ed766546ff5510614c695d48e3e0bafefee01ec8c04dc3a51297cb75bb57a
Status: Downloaded newer image for ghcr.io/openfaas/of-watchdog:0.9.10
 ---> 9f97468a4531
Step 2/31 : FROM --platform=${TARGETPLATFORM:-linux/amd64} node:12-alpine as ship
```

*Update handler.js*

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      var res = request('POST', 'http://34.95.26.217:8080/function/main', {
12          body: JSON.stringify(obj)
13  });
14  console.log(res["body"].toString())
15  return context.status(200).succeed(res["body"].toString('utf8', 1, res["body"].length-1).replace(/\\\"/g,'\"'
16  }
```

*Update package.json*

OpenFaaS > decorator > package.json > ...

```
1   {
2       "name": "openfaas-function", "version": "1.0.0",
3       "description": "OpenFaaS Function", "main": "handler.js",
            ▷ Debug
4       "scripts": {
5       "test": "echo \"Error: no test specified\" && exit 0" },
6       "keywords": [],
7       "author": "OpenFaaS Ltd", "license": "MIT", "dependencies": {
8       "sync-request": "^6.1.0" }
9   }
```

*Build docker image*

```
bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$ cd ~/OpenFaaS
bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$ faas-cli build -f decorator.yml
[0] > Building decorator.
Clearing temporary build folder: ./build/decorator/
Preparing: ./decorator/ build/decorator/function
Building: us.gcr.io/sofe4790u-lab3/decorator:latest with node12 template. Please wait..
Sending build context to Docker daemon   12.8kB
Step 1/31 : FROM --platform=${TARGETPLATFORM:-linux/amd64} ghcr.io/openfaas/of-watchdog:0.9.10 as watchdog
0.9.10: Pulling from openfaas/of-watchdog
c4fc21d17d12: Pulling fs layer
c4fc21d17d12: Verifying Checksum
c4fc21d17d12: Download complete
c4fc21d17d12: Pull complete
Digest: sha256:5d1ed766546ff5510614c695d48e3e0bafefee01ec8c04dc3a51297cb75bb57a
Status: Downloaded newer image for ghcr.io/openfaas/of-watchdog:0.9.10
 ---> 9f97468a4531
Step 2/31 : FROM --platform=${TARGETPLATFORM:-linux/amd64} node:12-alpine as ship
12-alpine: Pulling from library/node
df9b9388f04a: Pulling fs layer
3bf6d7380205: Pulling fs layer
7939e601ee5e: Pulling fs layer
31f0fb9de071: Pulling fs layer
31f0fb9de071: Waiting
```

*Push to container*

```
bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$ docker push us.gcr.io/sofe4790u-lab3/decorator
Using default tag: latest
The push refers to repository [us.gcr.io/sofe4790u-lab3/decorator]
de2351a82d23: Pushed
8b1412e6966b: Pushed
b6d4c4d8c4a1: Pushed
a63e9eed078c: Pushed
7c23f000b1cc: Pushed
293ddda09737: Pushed
583972de9644: Pushed
b6e9fd53cb5f: Pushed
d86bda25f4c1: Pushed
4d5ce352abac: Pushed
aded21115157: Pushed
7f30cde3f699: Layer already exists
fe810f5902cc: Layer already exists
dfd8c046c602: Layer already exists
4fc242d58285: Layer already exists
latest: digest: sha256:ac1a5c5d7cc28a30b27002e4781012c39647ace7f7db4d4bad0d3df259035396 size: 3663
```

*Deploy to OpenFaaS*

```
bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$ faas-cli deploy -f decorator.yml
Deploying: decorator.
WARNING! You are not using an encrypted connection to the gateway, consider using HTTPS.

Deployed. 202 Accepted.
URL: http://34.95.26.217:8080/function/decorator
```

*Send a JSON object and will return transparent*

```
bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$ curl http://34.95.26.217:8080/function/decorator -H 'Content-Type: application/json' -d '{ "Name": "Square", "Dim
ensions": 2 }'
{"Name":"Square","Dimensions":2,"Color":"Transparent"}bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$
```

**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 is that distributed applications need to be examined to see if they are working properly. This is known as health monitoring in this pattern. Through the use of health monitoring the checks are done with amalgamating two attributes. The first being the health checks that are done when requested by the health verification endpoint. The second being the analysis of what is returned by the check. Also assisting in the solution of this problem is there are tools and services that assist in auditing these web applications, and it is simple to make these tools/services endpoints that have the job to check the functionality of the system. Below is a list of checks that these services/tools perform:

- **Measuring response times:** The sum of the network latency, and the time that the service or application performs the request. If the sum is high, then there could be a potential problem on the network side (load balancing, traffic), or unoptimized usage of message transfer protocols.

- **Validating the response code:** Ensure that the correct response code is returning like 200 for OK.

- **Checking the content of the response:** Checking if the returned value of the response code 200 (OK) or something else. Also if the value is 20 it will still check other aspects of the page that will verify if the returned webpage is correct like a test phrase on the page.

- **Checking resources/services:** Checks the content delivery NW that is utilized to send content from global caches.

- **Checking expiration of SSL certs:** Checks if the SSL certificate is expired or not.

- **Measuring response time of DNS lookup:** Looks up the URL for the service/application being used to see if there is DNS failure, or check the response time.

- **Validating URL from the DNS lookup response:** To stop malicious redirections of requests if there is an attack aimed at the DNS server.

The Above checks also are requirements for the system to ensure that it is working as intended. The problem is that distributed applications and services need to be analyzed to see if they are working properly. This is done through health monitoring. The tools/services assist in the auditing process of these applications, and it is simple to create these tools/services. But to create these tools there are some requirements that need to be followed like Measuring response times, validating the response codes, checking the content of the response, checking the resources/services, checking the expiration of SSL certificates, measuring the response time of DNS lookup, and validating URL from the DNS lookup.

In part 2 the requirements are followed as well through the use of checking if there are errors in the system using endpoints created in the server.js. The errors in the system are detected using these endpoints and the requests are sent elsewhere, where they can be dealt with instead of bombarding the erroneous part indefinitely. Some of the requirements are used in part 3 where the handler for both the decorator and the main, are sending a successful response code when the operation is done.

## Design – Persistent Volumes

Persistent volumes allow the administrator to save data and use it as a plugin which is independent from the container and has its own lifecycle. The data can be shared between pods even after the pod has been restarted while keeping the data from the previous session can still be accessed. An example of persistent volume is used with applications that require a database, and if the application is closed, we still require the data to be stored. An example of persistent volume implementation is provided by Kubernetes official website (https://kubernetes.io/docs/tasks/configure-pod-container/configure-persistent-volume-storage/).

*Using Minikube to create a single node cluster*

```
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ minikube start
* minikube v1.27.0 on Debian 11.5 (amd64)
  - MINIKUBE_FORCE_SYSTEMD=true
  - MINIKUBE_HOME=/google/minikube
  - MINIKUBE_WANTUPDATENOTIFICATION=false
! Kubernetes 1.25.0 has a known issue with resolv.conf. minikube is using a workaround that should work for most use cases.
! For more information, see: https://github.com/kubernetes/kubernetes/issues/112135
* Automatically selected the docker driver. Other choices: none, ssh
* Using Docker driver with root privileges
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
* Downloading Kubernetes v1.25.0 preload ...
    > preloaded-images-k8s-v18-v1...:  385.37 MiB / 385.37 MiB  100.00% 204.83
    > gcr.io/k8s-minikube/kicbase:  0 B [_____________________] ?% ? p/s 8.9s
* Creating docker container (CPUs=2, Memory=4000MB) ...
* Preparing Kubernetes v1.25.0 on Docker 20.10.17 ...
  - kubelet.cgroups-per-qos=false
  - kubelet.enforce-node-allocatable=""
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: default-storageclass, storage-provisioner
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ minikube ssh
docker@minikube:~$
```

*Create index file and test output*

```
docker@minikube:~$ sudo mkdir /mnt/data
docker@minikube:~$ sudo sh -c "echo 'Hello from Kubernetes storage' > /mnt/data/index.html"
docker@minikube:~$ cat /mnt/data/index.html
Hello from Kubernetes storage
docker@minikube:~$
```

*Persistent Volume Yaml file*

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"
```

*Create and get the information for the persistent volume*

```
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ kubectl apply -f https://k8s.io/examples/pods/storage/pv-volume.yaml
persistentvolume/task-pv-volume created
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ kubectl get pv task-pv-volume
NAME             CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE
task-pv-volume   10Gi       RWO            Retain           Available           manual                  7s
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$
```

*Persistent Volume Claim Yaml file*

pv-claim.yaml

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

*Create and bound the persistent volume claim to persisten volume*

```
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ kubectl apply -f https://k8s.io/examples/pods/storage/pv-claim.yaml
persistentvolumeclaim/task-pv-claim created
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ kubectl get pv task-pv-volume
NAME             CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                   STORAGECLASS   REASON   AGE
task-pv-volume   10Gi       RWO            Retain           Bound    default/task-pv-claim   manual                  2m23s
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ kubectl get pvc task-pv-claim
NAME            STATUS   VOLUME           CAPACITY   ACCESS MODES   STORAGECLASS   AGE
task-pv-claim   Bound    task-pv-volume   10Gi       RWO            manual         19s
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$
```

*Creating a POD Yaml file*

pv-pod.yaml > {} spec > containers > ...

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: task-pv-pod
spec:
  volumes:
    - name: task-pv-storage
      persistentVolumeClaim:
        Created a minute ago
        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
```

*Create Pod and verify its running*

```
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ kubectl get pod task-pv-poc
NAME          READY   STATUS              RESTARTS   AGE
task-pv-pod   0/1     ContainerCreating   0          3s
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$
```

*Create the pod and verify its running within the container*

```
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ kubectl apply -f https://k8s.io/examples/pods/storage/pv-pod.yam
pod/task-pv-pod created
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ kubectl get pod task-pv-pod
NAME          READY   STATUS              RESTARTS   AGE
task-pv-pod   0/1     ContainerCreating   0          3s
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ 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 [193 kB]
Get:6 http://deb.debian.org/debian bullseye-updates/main amd64 Packages [14.6 kB]
Fetched 8600 kB in 2s (4874 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.
```

*Verify output*

```
root@task-pv-pod:/# curl http://localhost/
Hello from Kubernetes storage
root@task-pv-pod:/#
```

**========== END OF GROUP WORK ==========**

**Lab 3: Deploying a Circuit Breaking ambassador and FaaS**
**Member: Abdul Bhutta**
**Changes: RED**

## Part 1 - Health Endpoint Monitoring Pattern

### Problem

One of the business requirements is to periodically check for status or monitor the applications and backend services to ensure they are functioning without any issues. It is much easier to monitor with a centralized physical system than a cloud system because of having complete control of the physical system.

### Solution

The problem can be solved through a health monitoring system which checks the status of the components of the system, which can be implemented at the endpoint of the application and returns the status through a response code. Below is a list of requirements to implement the pattern.

| Requirements | Description |
|---|---|
| **Agent** | Analyzes the outcome from the application which performed the health check |
| **Application** | Check the health status of the request at the endpoint |

Below is a list of checks that these services/tools perform:

- **Measuring response times:** The sum of the network latency, and the time that the service or application performs the request. If the sum is high, then there could be a potential problem on the network side (load balancing, traffic), or unoptimized usage of message transfer protocols.
- **Validating the response code:** Ensure that the correct response code is returning like 200 for OK.
- **Checking the content of the response:** Checking if the returned value of the response code 200 (OK) or something else. Also if the value is 20 it will still check other aspects of the page that will verify if the returned webpage is correct like a test phrase on the page.
- **Checking resources/services:** Checks the content delivery NW that is utilized to send content from global caches.
- **Checking expiration of SSL certs:** Checks if the SSL certificate is expired or not.

- **Measuring response time of DNS lookup:** Looks up the URL for the service/application being used to see if there is DNS failure, or check the response time.

- **Validating URL from the DNS lookup response:** To stop malicious redirections of requests if there is an attack aimed at the DNS server.

**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?**

A Health Endpoint monitoring pattern is used for monitoring through periodic intervals and by sending requests to an endpoint of the cloud applications and services. The monitoring pattern will send back the request code of each service and application to determine the availability and response time (latency). To implement this pattern, you will require two key components an agent on the cloud service which will analyze and perform the health checks. The other component will be an application to check the health status of the request at the endpoint.

The requirements are achieved in part 2 since it behaves as a health endpoint monitor. It checks for errors at the endpoint while integrating the circuit breaker pattern, which is used as a contingency as failures are bound to happen. The circuit breakers allow the services not to be overloaded with too many requests and redirect the traffic to allow the service to come back online. The circuit breaker is configured to check every 3 seconds at the main endpoint, which acts as a health checkpoint. Part 3 also follows the requirements as well since the decorator pattern was applied to a function which transforms the input value sent by the user and checks whether any of the fields are missing. It then changes those values to a predefined default value.

- **Measuring response times:** The sum of the network latency, and the time that the service or application performs the request. If the sum is high, then there could be a potential problem on the network side (load balancing, traffic), or unoptimized usage of message transfer protocols.

- **Validating the response code:** Ensure that the correct response code is returning like 200 for OK.

- **Checking the content of the response:** Checking if the returned value of the response code 200 (OK) or something else. Also if the value is 20 it will still check other aspects of the page that will verify if the returned webpage is correct like a test phrase on the page.

- **Checking resources/services:** Checks the content delivery NW that is utilized to send content from global caches.

- **Checking expiration of SSL certs:** Checks if the SSL certificate is expired or not.

- **Measuring response time of DNS lookup:** Looks up the URL for the service/application being used to see if there is DNS failure, or check the response time.

- **Validating URL from the DNS lookup response:** To stop malicious redirections of requests if there is an attack aimed at the DNS server.

## Part 2

*Build Docker Image*

```
bhutta_abdul@cloudshell:~/SOFE4790U-lab3/part2/DummyServiceContainer (sofe4790u-lab3)$ docker build . -t us.gcr.io/sofe4790u-lab3/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
 ---> ca2eab3d6b36
Step 3/7 : COPY package*.json ./
 ---> Using cache
 ---> 244a016b5a88
Step 4/7 : RUN npm install
 ---> Using cache
 ---> 383cce04775a
Step 5/7 : COPY . .
 ---> Using cache
 ---> e097f6d23148
Step 6/7 : EXPOSE 80
 ---> Using cache
 ---> 4c35150b8d2c
Step 7/7 : CMD [ "npm", "start" ]
 ---> Using cache
 ---> 1579b6abb06d
Successfully built 1579b6abb06d
Successfully tagged us.gcr.io/sofe4790u-lab3/dummyservice:latest
bhutta_abdul@cloudshell:~/SOFE4790U-lab3/part2/DummyServiceContainer (sofe4790u-lab3)$
```

*Push the docker image to the container*

```
bhutta_abdul@cloudshell:~/SOFE4790U-lab3/part2/DummyServiceContainer (sofe4790u-lab3)$ docker push us.gcr.io/sofe4790u-lab3/dummyservice
Using default tag: latest
The push refers to repository [us.gcr.io/sofe4790u-lab3/dummyservice]
994112752128: Pushed
1c6407d87b9b: Pushed
ac4e20b24eca: Pushed
d9143b82daf0: 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: sha256:58f87b8f57c6112c38e3db063910cfb37b2f70240dd7f92f0fb9017645ce65b2 size: 3048
bhutta_abdul@cloudshell:~/SOFE4790U-lab3/part2/DummyServiceContainer (sofe4790u-lab3)$
```

*Create Image for Dummy Service*

```
bhutta_abdul@cloudshell:~/SOFE4790U-lab3/part2/DummyServiceContainer (sofe4790u-lab3)$ docker build . -t us.gcr.io/sofe4790u-lab3/dummyservice

Sending build context to Docker daemon  6.656kB
Step 1/7 : FROM node:carbon
carbon: Pulling from library/node
146bd6a88618: Pull complete
9935d0c62ace: Pull complete
db0efb86e806: Pull complete
e705a4c4fd31: Pull complete
c877b722db6f: Pull complete
645c20ec8214: Pull complete
db8fbd9db2fe: Pull complete
1c151cd1b3ea: Pull complete
fbd993995f40: Pull complete
Digest: sha256:a681bf74805b80d03eb21a6c0ef168a976108a287a74167ab593fc953aac34df
Status: Downloaded newer image for node:carbon
 ---> 8eeadf3757f4
Step 2/7 : WORKDIR /usr/src/app
 ---> Running in a0338cff83ed
Removing intermediate container a0338cff83ed
 ---> ca2eab3d6b36
Step 3/7 : COPY package*.json ./
 ---> 244a016b5a88
Step 4/7 : RUN npm install
 ---> Running in ae670c7bca78
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN dummyservice@1.0.1 No repository field.
npm WARN dummyservice@1.0.1 No license field.

added 57 packages from 42 contributors and audited 57 packages in 2.161s

6 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

Removing intermediate container ae670c7bca78
 ---> 383cce04775a
Step 5/7 : COPY . .
 ---> e097f6d23148
Step 6/7 : EXPOSE 80
 ---> Running in 0972982a8aca
Removing intermediate container 0972982a8aca
 ---> 4c35150b8d2c
Step 7/7 : CMD [ "npm", "start" ]
 ---> Running in f4b842fb0d2a
Removing intermediate container f4b842fb0d2a
 ---> 1579b6abb06d
Successfully built 1579b6abb06d
Successfully tagged us.gcr.io/sofe4790u-lab3/dummyservice:latest
bhutta_abdul@cloudshell:~/SOFE4790U-lab3/part2/DummyServiceContainer (sofe4790u-lab3)$
```

*Deploy the service and expose it through a load balancer*

```
bhutta_abdul@cloudshell:~/SOFE4790U-lab3/part2 (sofe4790u-lab3)$ kubectl delete services dummy-deployment
service "dummy-deployment" deleted
bhutta_abdul@cloudshell:~/SOFE4790U-lab3/part2 (sofe4790u-lab3)$ kubectl create -f dummy-deployment.yaml
deployment.apps/dummy-deployment created
bhutta_abdul@cloudshell:~/SOFE4790U-lab3/part2 (sofe4790u-lab3)$ kubectl expose deployment dummy-deployment --port=80 --type=LoadBalancer --name
dummy-deployment
service/dummy-deployment exposed
bhutta_abdul@cloudshell:~/SOFE4790U-lab3/part2 (sofe4790u-lab3)$ kubectl create -f backup-deployment.yaml
deployment.apps/backup-deployment created
bhutta_abdul@cloudshell:~/SOFE4790U-lab3/part2 (sofe4790u-lab3)$ kubectl expose deployment backup-deployment --port=80 --type=LoadBalancer --name
 backup-deployment
service/backup-deployment exposed
bhutta_abdul@cloudshell:~/SOFE4790U-lab3/part2 (sofe4790u-lab3)$
```

*Check deployment and services status*

```
bhutta_abdul@cloudshell:~/SOFE4790U-lab3/part2 (sofe4790u-lab3)$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
backup-deployment-79cf469564-xvlx6    1/1     Running   0          53s
dummy-deployment-7bd5bc5dd-tqzrf      1/1     Running   0          109s
bhutta_abdul@cloudshell:~/SOFE4790U-lab3/part2 (sofe4790u-lab3)$ kubectl get services
^[[ANAME                TYPE           CLUSTER-IP     EXTERNAL-IP    PORT(S)         AGE
backup-deployment    LoadBalancer   10.80.14.197   34.95.18.116   80:31897/TCP    49s
dummy-deployment     LoadBalancer   10.80.10.189   34.95.52.218   80:32651/TCP    106s
kubernetes           ClusterIP      10.80.0.1      <none>         443/TCP         97m
bhutta_abdul@cloudshell:~/SOFE4790U-lab3/part2 (sofe4790u-lab3)$ kubectl get deployments
NAME                 READY   UP-TO-DATE   AVAILABLE   AGE
backup-deployment    1/1     1            1           61s
dummy-deployment     1/1     1            1           117s
bhutta_abdul@cloudshell:~/SOFE4790U-lab3/part2 (sofe4790u-lab3)$
```

*Create Configmap*

```
bhutta_abdul@cloudshell:~/SOFE4790U-lab3/part2 (sofe4790u-lab3)$ kubectl create -f nginx-configmap.yaml
configmap/nginx-configuration created
bhutta_abdul@cloudshell:~/SOFE4790U-lab3/part2 (sofe4790u-lab3)$ kubectl create -f circuitbreaker.yaml
deployment.apps/circuitbreaker created
service/circuitbreaker created
bhutta_abdul@cloudshell:~/SOFE4790U-lab3/part2 (sofe4790u-lab3)$
```

*Get external IP Address*

```
bhutta_abdul@cloudshell:~/SOFE4790U-lab3/part2 (sofe4790u-lab3)$ kubectl get services
NAME                 TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)          AGE
backup-deployment    LoadBalancer   10.80.14.197   34.95.18.116    80:31897/TCP     3m6s
circuitbreaker       LoadBalancer   10.80.7.220    35.203.92.185   80:32080/TCP     93s
dummy-deployment     LoadBalancer   10.80.10.189   34.95.52.218    80:32651/TCP     4m3s
kubernetes           ClusterIP      10.80.0.1      <none>          443/TCP          100m
```

*Testing the Circuit Breaker, a) Run the command three times*

```
bhutta_abdul@cloudshell:~/SOFE4790U-lab3/part2 (sofe4790u-lab3)$ curl -v http://35.203.92.185
*   Trying 35.203.92.185:80...
* Connected to 35.203.92.185 (35.203.92.185) port 80 (#0)
> GET / HTTP/1.1
> Host: 35.203.92.185
> User-Agent: curl/7.74.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Server: nginx/1.13.7
< Date: Sat, 29 Oct 2022 20:37:59 GMT
< Content-Type: text/html; charset=utf-8
< Content-Length: 28
< Connection: keep-alive
< X-Powered-By: Express
< ETag: W/"1c-j4qqy4RWDpImPl1DpT1dstkgMcg"
<
* Connection #0 to host 35.203.92.185 left intact
SOMERESPONSE FROM 10.76.0.12bhutta_abdul@cloudshell:~/SOFE4790U-lab3/part2 (sofe4790u-lab3)$ curl -v http://35.203.92.185
*   Trying 35.203.92.185:80...
* Connected to 35.203.92.185 (35.203.92.185) port 80 (#0)
> GET / HTTP/1.1
> Host: 35.203.92.185
> User-Agent: curl/7.74.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Server: nginx/1.13.7
< Date: Sat, 29 Oct 2022 20:38:40 GMT
< Content-Type: text/html; charset=utf-8
< Content-Length: 28
< Connection: keep-alive
< X-Powered-By: Express
< ETag: W/"1c-j4qqy4RWDpImPl1DpT1dstkgMcg"
<
* Connection #0 to host 35.203.92.185 left intact
SOMERESPONSE FROM 10.76.0.12bhutta_abdul@cloudshell:~/SOFE4790U-lab3/part2 (sofe4790u-lab3)$ curl -v http://35.203.92.185
*   Trying 35.203.92.185:80...
* Connected to 35.203.92.185 (35.203.92.185) port 80 (#0)
> GET / HTTP/1.1
> Host: 35.203.92.185
> User-Agent: curl/7.74.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Server: nginx/1.13.7
< Date: Sat, 29 Oct 2022 20:38:41 GMT
< Content-Type: text/html; charset=utf-8
< Content-Length: 28
< Connection: keep-alive
< X-Powered-By: Express
< ETag: W/"1c-j4qqy4RWDpImPl1DpT1dstkgMcg"
<
* Connection #0 to host 35.203.92.185 left intact
```

*b) Mimic an Error*

```
bhutta_abdul@cloudshell:~/SOFE4790U-lab3/part2 (sofe4790u-lab3)$ curl -d "" -s -D - http://34.95.52.218/fakeerrormodeon
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: text/html; charset=utf-8
Content-Length: 18
ETag: W/"12-N3xK5AJr1Lw2pCRqwRZhBbPf84g"
Date: Sat, 29 Oct 2022 20:39:14 GMT
Connection: keep-alive

OK FROM 10.76.0.12bhutta_abdul@cloudshell:~/SOFE4790U-lab3/part2 (sofe4790u-lab3)$
```

*c) Reset the error*

```
bhutta_abdul@cloudshell:~/SOFE4790U-lab3/part2 (sofe4790u-lab3)$ curl -d "" -s -D - http://34.95.52.218/fakeerrormodeoff
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: text/html; charset=utf-8
Content-Length: 18
ETag: W/"12-N3xK5AJr1Lw2pCRqwRZhBbPf84g"
Date: Sat, 29 Oct 2022 20:40:04 GMT
Connection: keep-alive
```

*d) Run command and check IP*

```
bhutta_abdul@cloudshell:~/SOFE4790U-lab3/part2 (sofe4790u-lab3)$ curl -v http://35.203.92.185
*   Trying 35.203.92.185:80...
* Connected to 35.203.92.185 (35.203.92.185) port 80 (#0)
> GET / HTTP/1.1
> Host: 35.203.92.185
> User-Agent: curl/7.74.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Server: nginx/1.13.7
< Date: Sat, 29 Oct 2022 20:40:38 GMT
< Content-Type: text/html; charset=utf-8
< Content-Length: 28
< Connection: keep-alive
< X-Powered-By: Express
< ETag: W/"1c-j4qqy4RWDpImPl1DpT1dstkgMcg"
<
* Connection #0 to host 35.203.92.185 left intact
```

## Part 3

*Create a cluster admin role binding*

```
bhutta_abdul@cloudshell:~/SOFE4790U-lab3/part3 (sofe4790u-lab3)$ kubectl create clusterrolebinding "cluster-admin-$(whoami)" --clusterrole=cluster-admin --user
="$(gcloud config get-value core/account)"
Your active configuration is: [cloudshell-25206]
clusterrolebinding.rbac.authorization.k8s.io/cluster-admin-bhutta_abdul created
```

*Deploy OpenFaas to GKE*

```
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ curl -SLsf https://dl.get-arkade.dev/ | sudo sh
arkade install openfaas --load-balancer
x86_64
Downloading package https://github.com/alexellis/arkade/releases/download/0.8.50/arkade as /tmp/arkade
Download complete.

Running with sufficient permissions to attempt to move arkade to /usr/local/bin
New version of arkade installed to /usr/local/bin
Creating alias 'ark' for 'arkade'.
```

```
Open Source Marketplace For Developer Tools
```

*Install Client*

```
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ curl -SLsf https://cli.openfaas.com | sudo sh
Finding latest version from GitHub
0.14.11
Downloading package https://github.com/openfaas/faas-cli/releases/download/0.14.11/faas-cli as /tmp/faas-cli
Download complete.

Running with sufficient permissions to attempt to move faas-cli to /usr/local/bin
New version of faas-cli installed to /usr/local/bin
Creating alias 'faas' for 'faas-cli'.

CLI:
 commit:  8820d8e4a15dab900d8a7e8fc271851ccb94012e
 version: 0.14.11
```

*Verify OpenFaas is working*

```
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ kubectl -n openfaas get deployments -l "release=openfaas, app=openfaas"
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
alertmanager        1/1     1            1           6m45s
basic-auth-plugin   1/1     1            1           6m45s
gateway             1/1     1            1           6m45s
nats                1/1     1            1           6m45s
prometheus          1/1     1            1           6m45s
queue-worker        1/1     1            1           6m45s
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$
```

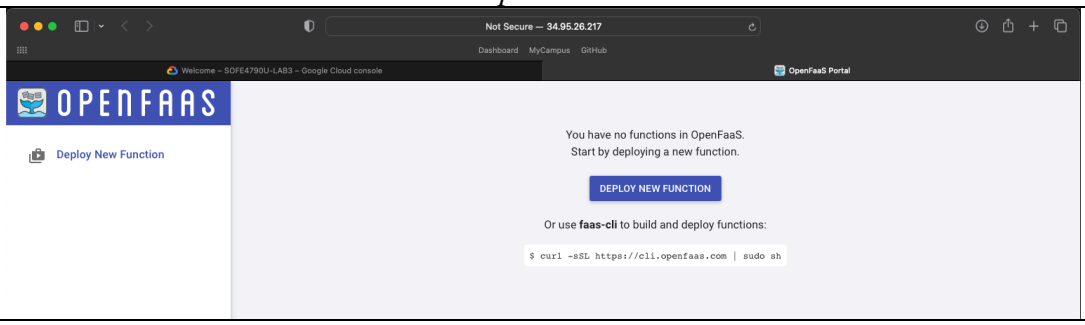
*Check OpenFaas is ready and get IP*

```
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ kubectl rollout status -n openfaas deploy/gateway
deployment "gateway" successfully rolled out
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ kubectl get svc -o wide gateway-external -n openfaas
NAME               TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)          AGE     SELECTOR
gateway-external   LoadBalancer   10.80.5.210    34.95.26.217    8080:32273/TCP   7m33s   app=gateway
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$
```

*Save Login and password*

```
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ export OPENFAAS_URL="34.95.26.217:8080"
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ PASSWORD=$(kubectl get secret -n openfaas basic-auth -o jsonpath="{.data.basic-auth-password}" | base64 --decode; echo)
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ echo $PASSWORD
9u248zD4208eW7dnHakhDkOf5
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ echo -n $PASSWORD | faas-cli login --username admin --password-stdin
Calling the OpenFaaS server to validate the credentials...
WARNING! You are not using an encrypted connection to the gateway, consider using HTTPS.
credentials saved for admin http://34.95.26.217:8080
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$
```

*OpenFaas*




*Create new directory and pull all available templates*

```
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ mkdir ~/OpenFaaS
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ cd ~/OpenFaaS
bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$ faas-cli template pull
Fetch templates from repository: https://github.com/openfaas/templates.git at
2022/10/30 00:40:07 Attempting to expand templates from https://github.com/openfaas/templates.git
2022/10/30 00:40:08 Fetched 17 template(s) : [csharp dockerfile go java11 java11-vert-x node node12 node12-debian node14 node16 node17 php7 php8 python pyth
on3 python3-debian ruby] from https://github.com/openfaas/templates.git
bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$ faas-cli new --list
Languages available as templates:
- csharp
- dockerfile
- go
- java11
- java11-vert-x
- node
- node12
- node12-debian
- node14
- node16
- node17
- php7
- php8
- python
- python3
- python3-debian
- ruby

bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$
```

*Empty NodeJS function*

```
bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$ faas-cli new --lang node12 --prefix us.gcr.io/sofe4790u-lab3 main
Folder: main created.
  ___                   _____           ____
 / _ \ _ __   ___ _ __ |  ___|_ _  __ _/ ___|
| | | | '_ \ / _ \ '_ \| |_ / _` |/ _` \___ \
| |_| | |_) |  __/ | | |  _| (_| | (_| |___) |
 \___/| .__/ \___|_| |_|_|  \__,_|\__,_|____/
      |_|


Function created in folder: main
Stack file written: main.yml

Notes:
You have created a new function which uses Node.js 12.

npm i --save can be used to add third-party packages like request or cheerio
npm documentation: https://docs.npmjs.com/

Unit tests are run at build time via "npm run", edit package.json to specify
how you want to execute them.

bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$
```

*Update handler.js file*

OpenFaaS > main > handler.js > ...

```
1  'use strict'
2
3  module.exports = async (event, context) => {
4      var parameters=JSON.stringify(event.body)
5      return context
6      .status(200)
7      .succeed(parameters)
8  }
```

*Build Docker image*

```
bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$ cd ~/OpenFaaS
bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$ faas-cli build -f main.yml
[0] > Building main.
Clearing temporary build folder: ./build/main/
Preparing: ./main/ build/main/function
Building: us.gcr.io/sofe4790u-lab3/main:latest with node12 template. Please wait..
Sending build context to Docker daemon   12.8kB
Step 1/31 : FROM --platform=${TARGETPLATFORM:-linux/amd64} ghcr.io/openfaas/of-watchdog:0.9.10 as watchdog
0.9.10: Pulling from openfaas/of-watchdog
c4fc21d17d12: Pulling fs layer
c4fc21d17d12: Download complete
c4fc21d17d12: Pull complete
Digest: sha256:5d1ed766546ff5510614c695d48e3e0bafefee01ec8c04dc3a51297cb75bb57a
Status: Downloaded newer image for ghcr.io/openfaas/of-watchdog:0.9.10
 ---> 9f97468a4531
Step 2/31 : FROM --platform=${TARGETPLATFORM:-linux/amd64} node:12-alpine as ship
12-alpine: Pulling from library/node
```

*Push to container*

```
bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$ docker push us.gcr.io/sofe4790u-lab3/main
Using default tag: latest
The push refers to repository [us.gcr.io/sofe4790u-lab3/main]
c4cc512dfbdb: Pushed
f3e52fa4f0ee: Pushed
c17d15683f0c: Pushed
4ce34a584422: Pushed
9a941d3d77c7: Pushed
6229463858ce: Pushed
57021fbca4bc: Pushed
2900e05f5c62: Pushed
d86bda25f4c1: Pushed
5ee04ae9a725: Pushed
4c5614929cb1: Pushed
7f30cde3f699: Layer already exists
fe810f5902cc: Layer already exists
dfd8c046c602: Layer already exists
4fc242d58285: Layer already exists
latest: digest: sha256:28f6b6df3fcb179a238b0be33a1b5c8479b09efbcdf5a1ed8a136e106d3ac253 size: 3659
```

*Deploy to OpenFaaS*

```
bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$ 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://34.95.26.217:8080/function/main
```

*Send JSON object*

```
bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$ curl http://34.95.26.217:8080/function/main -H 'Content-Type: application/json' -d '{ "Name": "Square", "Color": "Red",
 "Dimensions": 2 }'
{"Name":"Square","Color":"Red","Dimensions":2}bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$
```

*Create new empty NodeJS function*

```
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ cd ~/OpenFaaS
bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$ faas-cli new --lang node12 --prefix us.gcr.io/sofe4790u-lab3 decorator
Folder: decorator created.
  ___                   _____           ____
 / _ \ _ __   ___ _ __ |  ___|_ _  __ _/ ___|
| | | | '_ \ / _ \ '_ \| |_ / _` |/ _` \___ \
| |_| | |_) |  __/ | | |  _| (_| | (_| |___) |
 \___/| .__/ \___|_| |_|_|  \__,_|\__,_|____/
      |_|


Function created in folder: decorator
Stack file written: decorator.yml

Notes:
You have created a new function which uses Node.js 12.

npm i --save can be used to add third-party packages like request or cheerio
npm documentation: https://docs.npmjs.com/

Unit tests are run at build time via "npm run", edit package.json to specify
how you want to execute them.

bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$ cd ~/OpenFaaS
bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$ faas-cli build -f decorator.yml
[0] > Building decorator.
Clearing temporary build folder: ./build/decorator/
Preparing: ./decorator/ build/decorator/function
Building: us.gcr.io/sofe4790u-lab3/decorator:latest with node12 template. Please wait..
Sending build context to Docker daemon   12.8kB
Step 1/31 : FROM --platform=${TARGETPLATFORM:-linux/amd64} ghcr.io/openfaas/of-watchdog:0.9.10 as watchdog
0.9.10: Pulling from openfaas/of-watchdog
c4fc21d17d12: Pulling fs layer
c4fc21d17d12: Verifying Checksum
c4fc21d17d12: Download complete
c4fc21d17d12: Pull complete
Digest: sha256:5d1ed766546ff5510614c695d48e3e0bafefee01ec8c04dc3a51297cb75bb57a
Status: Downloaded newer image for ghcr.io/openfaas/of-watchdog:0.9.10
 ---> 9f97468a4531
Step 2/31 : FROM --platform=${TARGETPLATFORM:-linux/amd64} node:12-alpine as ship
```

*Update handler.js*

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      var res = request('POST', 'http://34.95.26.217:8080/function/main', {
12          body: JSON.stringify(obj)
13  });
14  console.log(res["body"].toString())
15  return context.status(200).succeed(res["body"].toString('utf8', 1, res["body"].length-1).replace(/\\\"/g,'\"'
16  }
```

*Update package.json*

OpenFaaS > decorator > package.json > ...

```
{
    "name": "openfaas-function", "version": "1.0.0",
    "description": "OpenFaaS Function", "main": "handler.js",
    "scripts": {
    "test": "echo \"Error: no test specified\" && exit 0" },
    "keywords": [],
    "author": "OpenFaaS Ltd", "license": "MIT", "dependencies": {
    "sync-request": "^6.1.0" }
}
```

*Build docker image*

```
bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$ cd ~/OpenFaaS
bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$ faas-cli build -f decorator.yml
[0] > Building decorator.
Clearing temporary build folder: ./build/decorator/
Preparing: ./decorator/ build/decorator/function
Building: us.gcr.io/sofe4790u-lab3/decorator:latest with node12 template. Please wait..
Sending build context to Docker daemon   12.8kB
Step 1/31 : FROM --platform=${TARGETPLATFORM:-linux/amd64} ghcr.io/openfaas/of-watchdog:0.9.10 as watchdog
0.9.10: Pulling from openfaas/of-watchdog
c4fc21d17d12: Pulling fs layer
c4fc21d17d12: Verifying Checksum
c4fc21d17d12: Download complete
c4fc21d17d12: Pull complete
Digest: sha256:5d1ed766546ff5510614c695d48e3e0bafefee01ec8c04dc3a51297cb75bb57a
Status: Downloaded newer image for ghcr.io/openfaas/of-watchdog:0.9.10
 ---> 9f97468a4531
Step 2/31 : FROM --platform=${TARGETPLATFORM:-linux/amd64} node:12-alpine as ship
12-alpine: Pulling from library/node
df9b9388f04a: Pulling fs layer
3bf6d7380205: Pulling fs layer
7939e601ee5e: Pulling fs layer
31f0fb9de071: Pulling fs layer
31f0fb9de071: Waiting
```

*Push to container*

```
bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$ docker push us.gcr.io/sofe4790u-lab3/decorator
Using default tag: latest
The push refers to repository [us.gcr.io/sofe4790u-lab3/decorator]
de2351a82d23: Pushed
8b1412e6966b: Pushed
b6d4c4d8c4a1: Pushed
a63e9eed078c: Pushed
7c23f000b1cc: Pushed
293ddda09737: Pushed
583972de9644: Pushed
b6e9fd53cb5f: Pushed
d86bda25f4c1: Pushed
4d5ce352abac: Pushed
aded21115157: Pushed
7f30cde3f699: Layer already exists
fe810f5902cc: Layer already exists
dfd8c046c602: Layer already exists
4fc242d58285: Layer already exists
latest: digest: sha256:ac1a5c5d7cc28a30b27002e4781012c39647ace7f7db4d4bad0d3df259035396 size: 3663
```

*Deploy to OpenFaaS*

```
bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$ faas-cli deploy -f decorator.yml
Deploying: decorator.
WARNING! You are not using an encrypted connection to the gateway, consider using HTTPS.

Deployed. 202 Accepted.
URL: http://34.95.26.217:8080/function/decorator
```

*Send a JSON object and will return transparent*

```
bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$ curl http://34.95.26.217:8080/function/decorator -H 'Content-Type: application/json' -d '{ "Name": "Square", "Dim
ensions": 2 }'
{"Name":"Square","Dimensions":2,"Color":"Transparent"}bhutta_abdul@cloudshell:~/OpenFaaS (sofe4790u-lab3)$
```

**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 allow the administrator to save data and use it as a plugin which is independent from the container and has its own lifecycle. The data can be shared between pods even after the pod has been restarted while keeping the data from the previous session can still be accessed. An example of persistent volume is used with applications that require a database, and if the application is closed, we still require the data to be stored. An example of persistent volume implementation is provided by Kubernetes official website (https://kubernetes.io/docs/tasks/configure-pod-container/configure-persistent-volume-storage/).

*Using Minikube to create a single node cluster*

```
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ minikube start
* minikube v1.27.0 on Debian 11.5 (amd64)
  - MINIKUBE_FORCE_SYSTEMD=true
  - MINIKUBE_HOME=/google/minikube
  - MINIKUBE_WANTUPDATENOTIFICATION=false
! Kubernetes 1.25.0 has a known issue with resolv.conf. minikube is using a workaround that should work for most use cases.
! For more information, see: https://github.com/kubernetes/kubernetes/issues/112135
* Automatically selected the docker driver. Other choices: none, ssh
* Using Docker driver with root privileges
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
* Downloading Kubernetes v1.25.0 preload ...
    > preloaded-images-k8s-v18-v1...:  385.37 MiB / 385.37 MiB  100.00% 204.83
    > gcr.io/k8s-minikube/kicbase:  0 B [_____________________] ?% ? p/s 8.9s
* Creating docker container (CPUs=2, Memory=4000MB) ...
* Preparing Kubernetes v1.25.0 on Docker 20.10.17 ...
  - kubelet.cgroups-per-qos=false
  - kubelet.enforce-node-allocatable=""
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: default-storageclass, storage-provisioner
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ minikube ssh
docker@minikube:~$
```

*Create index file and test output*

```
docker@minikube:~$ sudo mkdir /mnt/data
docker@minikube:~$ sudo sh -c "echo 'Hello from Kubernetes storage' > /mnt/data/index.html"
docker@minikube:~$ cat /mnt/data/index.html
Hello from Kubernetes storage
docker@minikube:~$
```

*Persistent Volume Yaml file*

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"
```

*Create and get the information for the persistent volume*

```
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ kubectl apply -f https://k8s.io/examples/pods/storage/pv-volume.yaml
persistentvolume/task-pv-volume created
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ kubectl get pv task-pv-volume
NAME             CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE
task-pv-volume   10Gi       RWO            Retain           Available           manual                  7s
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$
```

*Persistent Volume Claim Yaml file*

pv-claim.yaml

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

*Create and bound the persistent volume claim to persisten volume*

```
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ kubectl apply -f https://k8s.io/examples/pods/storage/pv-claim.yaml
persistentvolumeclaim/task-pv-claim created
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ kubectl get pv task-pv-volume
NAME             CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                   STORAGECLASS   REASON   AGE
task-pv-volume   10Gi       RWO            Retain           Bound    default/task-pv-claim   manual                  2m23s
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ kubectl get pvc task-pv-claim
NAME            STATUS   VOLUME           CAPACITY   ACCESS MODES   STORAGECLASS   AGE
task-pv-claim   Bound    task-pv-volume   10Gi       RWO            manual         19s
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$
```

*Creating a POD Yaml file*

pv-pod.yaml > {} spec > containers > ...

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: task-pv-pod
spec:
  volumes:
    - name: task-pv-storage
      persistentVolumeClaim:
        Created a minute ago
        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
```

*Create Pod and verify its running*

```
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ kubectl get pod task-pv-pod
NAME          READY   STATUS              RESTARTS   AGE
task-pv-pod   0/1     ContainerCreating   0          3s
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$
```

*Create the pod and verify its running within the container*

```
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ kubectl apply -f https://k8s.io/examples/pods/storage/pv-pod.yam
pod/task-pv-pod created
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ kubectl get pod task-pv-pod
NAME          READY   STATUS              RESTARTS   AGE
task-pv-pod   0/1     ContainerCreating   0          3s
bhutta_abdul@cloudshell:~ (sofe4790u-lab3)$ 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 [193 kB]
Get:6 http://deb.debian.org/debian bullseye-updates/main amd64 Packages [14.6 kB]
Fetched 8600 kB in 2s (4874 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.
```

*Verify output*

```
root@task-pv-pod:/# curl http://localhost/
Hello from Kubernetes storage
root@task-pv-pod:/#
```

**Lab 3: Deploying a Circuit Breaking ambassador and FaaS**
**Member: Owen Musselman**

**Note: Changes are in Red**

**Objective:**

- Learn how to create a service using NodeJS
- Learn how to create a Docker image
- Learn how to configure and use a circuit breaker using Nginx
- Get familiar with FaaS
- Learn how to configure and use OpenFaaS on GCP
- Get familiar with Decorator Pattern

**Part 1:**

The problem is that distributed applications need to be examined to see if they are working properly. This is known as health monitoring in this pattern. Through the use of health monitoring the checks are done with amalgamating two attributes. The first being the health checks that are done when requested by the health verification endpoint. The second being the analysis of what is returned by the check. Also assisting in the solution of this problem is there are tools and services that assist in auditing these web applications, and it is simple to make these tools/services endpoints that have the job to check the functionality of the system. Below is a list of checks that these services/tools perform:

- **Measuring response times:** The sum of the network latency, and the time that the service or application performs the request. If the sum is high, then there could be a potential problem on the network side (load balancing, traffic), or unoptimized usage of message transfer protocols.
- **Validating the response code:** Ensure that the correct response code is returning like 200 for OK.
- **Checking the content of the response:** Checking if the returned value of the response code 200 (OK) or something else. Also if the value is 20 it will still check other aspects of the page that will verify if the returned webpage is correct like a test phrase on the page.
- **Checking resources/services:** Checks the content delivery NW that is utilized to send content from global caches.
- **Checking expiration of SSL certs:** Checks if the SSL certificate is expired or not.
- **Measuring response time of DNS lookup:** Looks up the URL for the service/application being used to see if there is DNS failure, or check the response time.
- **Validating URL from the DNS lookup response:** To stop malicious redirections of requests if there is an attack aimed at the DNS server.

The Above checks also are requirements for the system to assure that it is working as intended.

Other Requirements:

- Agent: Analyze outcomes that are sent from the application that performed the health check.
- Application: Check the status of the request at the designated endpoint.

Should periodically check the status of the application and services to make sure that they are functioning as intended. It is noticeably easier to monitor audit a more centralized system than a cloud system, as with the cloud system you do not have complete control over the physical systems components, only virtual ones.

This problem could be solved utilizing a health monitor system that would check the status of the system through the use of endpoints of the application. The endpoints would return status codes indicating the current status.

**Part 2:**

```
slikcaustic1@cloudshell:~ (bubbly-granite-362015)$ 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 | 1.92 MiB/s, done.
Resolving deltas: 100% (4/4), done.
```

Cloning the lab repository.

```
slikcaustic1@cloudshell:~/SOFE4790U-lab3/part2 (bubbly-granite-362015)$ cd DummyServiceContainer/
slikcaustic1@cloudshell:~/SOFE4790U-lab3/part2/DummyServiceContainer (bubbly-granite-362015)$
```

Changing to the DummyServiceContainer directory from the cloned repository.

```
slikcaustic1@cloudshell:~/SOFE4790U-lab3/part2/DummyServiceContainer (bubbly-granite-362015)$ docker build . -t us.gcr.io/bubbly-granite-362015/dummyservice
Sending build context to Docker daemon  6.656kB
Step 1/7 : FROM node:carbon
carbon: Pulling from library/node
146bd6a88618: Pull complete
9935d0c62ace: Pull complete
db0efb86e806: Pull complete
e705a4c4fd31: Pull complete
c877b722db6f: Pull complete
645c20ec8214: Pull complete
db8fbd9db2fe: Pull complete
1c151cd1b3ea: Pull complete
fbd993995f40: Pull complete
Digest: sha256:a681bf74805b80d03eb21a6c0ef168a976108a287a74167ab593fc953aac34df
Status: Downloaded newer image for node:carbon
 ---> 8eeadf3757f4
Step 2/7 : WORKDIR /usr/src/app
 ---> Running in 09f1ae185248
Removing intermediate container 09f1ae185248
 ---> 27784e84298b
Step 3/7 : COPY package*.json ./
 ---> 5a9a6e5a3cc5
Step 4/7 : RUN npm install
 ---> Running in 64231dbe6e5f
```

Creating docker image.

```
slikcaustic1@cloudshell:~/SOFE4790U-lab3/part2/DummyServiceContainer (bubbly-granite-362015)$ docker push us.gcr.io/bubbly-granite-362015/dummyservice
Using default tag: latest
The push refers to repository [us.gcr.io/bubbly-granite-362015/dummyservice]
c54aabad4fd7: Pushed
b589f5049c60: Pushed
20c342325374: Pushed
4fdcb28002fd: 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: sha256:0391b559fe8363eb9f941f987b1c07641877e5d076fa667ff1956115db0376a0 size: 3048
```

Pushing docker image to docker hub.

```
20        image: us.gcr.io/bubbly-granite-362015/dummyservice
```

Setting the image in dummy-deployment.yaml

```
slikcaustic1@cloudshell:~/SOFE4790U-lab3/part2 (bubbly-granite-362015)$ kubectl create -f dummy-deployment.yaml
deployment.apps/dummy-deployment created
```

Creating the deployment that contains the image that was previously created.

```
slikcaustic1@cloudshell:~/SOFE4790U-lab3/part2 (bubbly-granite-362015)$ kubectl expose deployment dummy-deployment --port=80 --type=LoadBalancer --name dummy-deployment
service/dummy-deployment exposed
```

Exposing dummy-deployment using loadbalancer.

```
slikcaustic1@cloudshell:~/SOFE4790U-lab3/part2 (bubbly-granite-362015)$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
dummy-deployment-695cbd746d-zlkx2   1/1     Running   0          4m12s
mongodb-deployment-7945646c67-r94d6   1/1     Running   0          19d
mysql-deployment-5496fdc956-8g9rw   1/1     Running   0          19d
```

Showing dummy pods are created.

```
slikcaustic1@cloudshell:~/SOFE4790U-lab3/part2 (bubbly-granite-362015)$ kubectl get deployments
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
dummy-deployment   1/1     1            1           4m19s
mongodb-deployment   1/1     1            1           34d
mysql-deployment   1/1     1            1           39d
```

Showing dummy deployments are created.

```
slikcaustic1@cloudshell:~/SOFE4790U-lab3/part2 (bubbly-granite-362015)$ kubectl get services
NAME               TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)          AGE
dummy-deployment   LoadBalancer   10.108.5.125    35.234.244.62   80:32704/TCP     6m30s
kubernetes         ClusterIP      10.108.0.1      <none>          443/TCP          41d
mongodb-service    LoadBalancer   10.108.13.162   35.234.248.86   3306:30766/TCP   34d
mysql-service      LoadBalancer   10.108.8.33     34.152.15.114   3306:31301/TCP   39d
```

Showing dummy services were created.

```
20        image: us.gcr.io/bubbly-granite-362015/dummyservice
```

Editing the image to be used in backup-deployment.yaml

```
slikcaustic1@cloudshell:~/SOFE4790U-lab3/part2 (bubbly-granite-362015)$ kubectl create -f backup-deployment.yaml
deployment.apps/backup-deployment created
```

Create backup-deployment.

```
slikcaustic1@cloudshell:~/SOFE4790U-lab3/part2 (bubbly-granite-362015)$ kubectl expose deployment backup-deployment --port=80 --type=LoadBalancer --name backup-deployment
service/backup-deployment exposed
```

Expose the backup-deployment with loadbalancer.

```
slikcaustic1@cloudshell:~/SOFE4790U-lab3/part2 (bubbly-granite-362015)$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
backup-deployment-56ffcb8fd8-cz7vs    1/1     Running   0          4m55s
dummy-deployment-695cbd746d-zlkx2     1/1     Running   0          18m
mongodb-deployment-7945646c67-r94d6   1/1     Running   0          19d
mysql-deployment-5496fdc956-8g9rw     1/1     Running   0          19d
```

Showing the backup pods was created.

```
slikcaustic1@cloudshell:~/SOFE4790U-lab3/part2 (bubbly-granite-362015)$ kubectl get deployments
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
backup-deployment   1/1     1            1           5m6s
dummy-deployment    1/1     1            1           18m
mongodb-deployment  1/1     1            1           34d
mysql-deployment    1/1     1            1           39d
```

Showing the backup deployment was created.

```
slikcaustic1@cloudshell:~/SOFE4790U-lab3/part2 (bubbly-granite-362015)$ kubectl get services
NAME                TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)          AGE
backup-deployment   LoadBalancer   10.108.6.34     34.152.63.218   80:30637/TCP     4m16s
dummy-deployment    LoadBalancer   10.108.5.125    35.234.244.62   80:32704/TCP     15m
kubernetes          ClusterIP      10.108.0.1      <none>          443/TCP          41d
mongodb-service     LoadBalancer   10.108.13.162   35.234.248.86   3306:30766/TCP   34d
mysql-service       LoadBalancer   10.108.8.33     34.152.15.114   3306:31301/TCP   39d
```

Showing the backup service was created.

```
slikcaustic1@cloudshell:~/SOFE4790U-lab3/part2 (bubbly-granite-362015)$ kubectl create -f nginx-configmap.yaml
configmap/nginx-configuration created
```

Creating the configmap.

```
slikcaustic1@cloudshell:~/SOFE4790U-lab3/part2 (bubbly-granite-362015)$ kubectl create -f circuitbreaker.yaml
deployment.apps/circuitbreaker created
service/circuitbreaker created
```

Creating the circuit breaker deployment and service.

```
slikcaustic1@cloudshell:~/SOFE4790U-lab3/part2 (bubbly-granite-362015)$ kubectl get services
NAME                TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)          AGE
backup-deployment   LoadBalancer   10.108.6.34     34.152.63.218   80:30637/TCP     8m29s
circuitbreaker      LoadBalancer   10.108.13.128   35.203.71.168   80:31993/TCP     103s
dummy-deployment    LoadBalancer   10.108.5.125    35.234.244.62   80:32704/TCP     19m
kubernetes          ClusterIP      10.108.0.1      <none>          443/TCP          41d
mongodb-service     LoadBalancer   10.108.13.162   35.234.248.86   3306:30766/TCP   34d
mysql-service       LoadBalancer   10.108.8.33     34.152.15.114   3306:31301/TCP   39d
```

Getting the external IP of the circuitbreaker (35.203.71.168)

```
slikcaustic1@cloudshell:~/SOFE4790U-lab3/part2 (bubbly-granite-362015)$ curl -v http://35.203.71.168
*   Trying 35.203.71.168:80...
* Connected to 35.203.71.168 (35.203.71.168) port 80 (#0)
> GET / HTTP/1.1
> Host: 35.203.71.168
> User-Agent: curl/7.74.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Server: nginx/1.13.7
< Date: Fri, 21 Oct 2022 01:01:54 GMT
< Content-Type: text/html; charset=utf-8
< Content-Length: 29
< Connection: keep-alive
< X-Powered-By: Express
< ETag: W/"1d-Cvi4umLI3rerUQoAYYxO8GMsv2w"
<
* Connection #0 to host 35.203.71.168 left intact
SOMERESPONSE FROM 10.104.1.23slikcaustic1@cloudshell:~/SOFE4790U-lab3/part2 (bubbly-granite-362015)$
```

Testing the circuit breaker and looking at the ip of the local machine.

```
SOMERESPONSE FROM 10.104.1.23slikcaustic1@cloudshell:~/SOFE4790U-lab3/part2 (bubbly-granite-362015)$ curl -d "" -s -D - http://35.203.71.168/fakeerrormodeon
HTTP/1.1 200 OK
Server: nginx/1.13.7
Date: Fri, 21 Oct 2022 01:03:47 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 19
Connection: keep-alive
X-Powered-By: Express
ETag: W/"13-4VHxHCNFxsLVz0uMznou/JYPrAM"

OK FROM 10.104.1.23slikcaustic1@cloudshell:~/SOFE4790U-lab3/part2 (bubbly-granite-362015)$ curl -v http://35.203.71.168
```

Executing a mimicked error. Local ip still ends in 23.

```
OK FROM 10.104.1.23slikcaustic1@cloudshell:~/SOFE4790U-lab3/part2 (bubbly-granite-362015)$ curl -v http://35.203.71.168
*   Trying 35.203.71.168:80...
* Connected to 35.203.71.168 (35.203.71.168) port 80 (#0)
> GET / HTTP/1.1
> Host: 35.203.71.168
> User-Agent: curl/7.74.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Server: nginx/1.13.7
< Date: Fri, 21 Oct 2022 01:04:22 GMT
< Content-Type: text/html; charset=utf-8
< Content-Length: 29
< Connection: keep-alive
< X-Powered-By: Express
< ETag: W/"1d-C3XZZfHbU1NQd0uDwLmgIlhD14k"
<
* Connection #0 to host 35.203.71.168 left intact
SOMERESPONSE FROM 10.104.1.24slikcaustic1@cloudshell:~/SOFE4790U-lab3/part2 (bubbly-granite-362015)$
```

After executing the mimicked error we test the circuit breaker by using the first curl command and can see that the local ip is now ending in 24, when it was previously 24. This indicates the circuit breaker is functioning properly.

```
slikcaustic1@cloudshell:~/SOFE4790U-lab3/part2 (bubbly-granite-362015)$ curl -d "" -s -D - http://35.203.71.168/fakeerrormodeoff
HTTP/1.1 200 OK
Server: nginx/1.13.7
Date: Fri, 21 Oct 2022 01:08:39 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 19
Connection: keep-alive
X-Powered-By: Express
ETag: W/"13-b44jnrF5+u/YHCuCXpAprlkhxhY"

OK FROM 10.104.1.24slikcaustic1@cloudshell:~/SOFE4790U-lab3/part2 (bubbly-granite-362015)$
```

Disabling the error (resetting it). The ip of the local machine ends in 24.

```
OK FROM 10.104.1.24slikcaustic1@cloudshell:~/SOFE4790U-lab3/part2 (bubbly-granite-362015)$ curl -v http://35.203.71.168
*   Trying 35.203.71.168:80...
* Connected to 35.203.71.168 (35.203.71.168) port 80 (#0)
> GET / HTTP/1.1
> Host: 35.203.71.168
> User-Agent: curl/7.74.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Server: nginx/1.13.7
< Date: Fri, 21 Oct 2022 01:09:42 GMT
< Content-Type: text/html; charset=utf-8
< Content-Length: 29
< Connection: keep-alive
< X-Powered-By: Express
< ETag: W/"1d-C3XZZfHbU1NQd0uDwLmgIlhD14k"
<
* Connection #0 to host 35.203.71.168 left intact
SOMERESPONSE FROM 10.104.1.24slikcaustic1@cloudshell:~/SOFE4790U-lab3/part2 (bubbly-granite-362015)$
```

Checking the local ip after resetting and the ip is still ending in 24, as that line was working it will continue to work.

**Part 3:**

```
slikcaustic1@cloudshell:~ (bubbly-granite-362015)$ kubectl create clusterrolebinding "cluster-admin-$(whoami)" --clusterrole=cluster-admin --user="$(gcloud config get-value core/account)"
Your active configuration is: [cloudshell-10691]
clusterrolebinding.rbac.authorization.k8s.io/cluster-admin-slikcaustic1 created
```

Creating a cluster with an admin role.

```
slikcaustic1@cloudshell:~ (bubbly-granite-362015)$ curl -SLsf https://dl.get-arkade.dev/ | sudo sh
arkade install openfaas --load-balancer
x86_64
Downloading package https://github.com/alexellis/arkade/releases/download/0.8.47/arkade as /tmp/arkade
Download complete.
```

Getting openfaas.

```
slikcaustic1@cloudshell:~ (bubbly-granite-362015)$ curl -SLsf https://cli.openfaas.com | sudo sh
Finding latest version from GitHub
0.14.11
Downloading package https://github.com/openfaas/faas-cli/releases/download/0.14.11/faas-cli as /tmp/faas-cli
Download complete.

Running with sufficient permissions to attempt to move faas-cli to /usr/local/bin
New version of faas-cli installed to /usr/local/bin
Creating alias 'faas' for 'faas-cli'.
 ___                  _____           ____
/ _ \ _ __   ___ _ __ |  ___|_ _  __ _/ ___|
| | | | '_ \ / _ \ '_ \| |_ / _` |/ _` \___ \
| |_| | |_) |  __/ | | |  _| (_| | (_| |___) |
 \___/| .__/ \___|_| |_|_|  \__,_|\__,_|____/
      |_|

CLI:
 commit:  8820d8e4a15dab900d8a7e8fc271851ccb94012e
 version: 0.14.11
```

Getting openfaas client.

```
slikcaustic1@cloudshell:~ (bubbly-granite-362015)$ kubectl -n openfaas get deployments -l "release=openfaas, app=openfaas"
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
alertmanager        1/1     1            1           3m10s
basic-auth-plugin   1/1     1            1           3m10s
gateway             1/1     1            1           3m10s
nats                1/1     1            1           3m10s
prometheus          1/1     1            1           3m10s
queue-worker        1/1     1            1           3m10s
```

Verifying that openfaas is running.

```
slikcaustic1@cloudshell:~ (bubbly-granite-362015)$ kubectl rollout status -n openfaas deploy/gateway
deployment "gateway" successfully rolled out
```

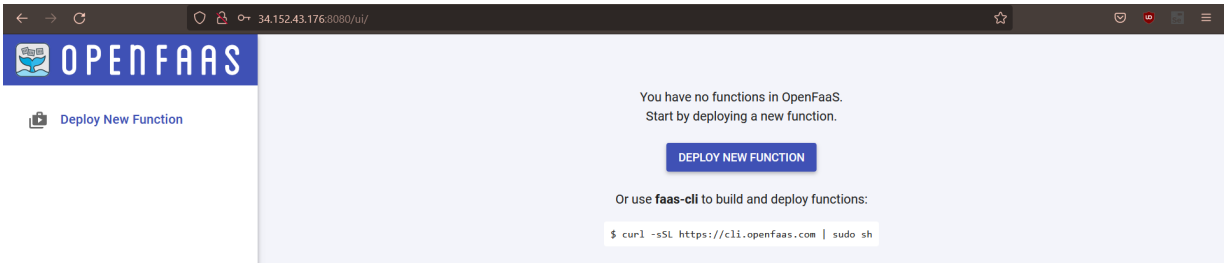
Checking that openfaas has been rolled out.

```
slikcaustic1@cloudshell:~ (bubbly-granite-362015)$ kubectl get svc -o wide gateway-external -n openfaas
NAME               TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)          AGE     SELECTOR
gateway-external   LoadBalancer   10.108.6.226   34.152.43.176   8080:32696/TCP   7m58s   app=gateway
```

Getting the openfaas external IP: 34.152.43.176

```
slikcaustic1@cloudshell:~ (bubbly-granite-362015)$ export OPENFAAS_URL="34.152.43.176:8080"
PASSWORD=$(kubectl get secret -n openfaas basic-auth -o jsonpath="{.data.basic-auth-password}" | base64 --decode; echo)
echo $PASSWORD
echo -n $PASSWORD | faas-cli login --username admin --password-stdin
5SmUu8xj0NS6zLc0bO466E6P6
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://34.152.43.176:8080
```

Getting the password and logging into the gateway. Password is: 5SmUu8xj0NS6zLc0bO466E6P6




Logging into openfaas through the browser using the 34.152.43.176:8080/ui/

```
slikcaustic1@cloudshell:~ (bubbly-granite-362015)$ mkdir ~/OpenFaaS
cd ~/OpenFaaS
faas-cli template pull
faas-cli new --list
Fetch templates from repository: https://github.com/openfaas/templates.git at
2022/10/22 21:04:53 Attempting to expand templates from https://github.com/openfaas/templates.git
2022/10/22 21:04:53 Fetched 17 template(s) : [csharp dockerfile go java11 java11-vert-x node node12 node12-debian node14 node16 node17 php7 php8 python python3 python3-debian ruby] from https:
//github.com/openfaas/templates.git
Languages available as templates:
- csharp
- dockerfile
- go
- java11
- java11-vert-x
- node
- node12
- node12-debian
- node14
- node16
- node17
- php7
- php8
- python
- python3
- python3-debian
- ruby
```

Creating directory and pulling templates.

```
slikcaustic1@cloudshell:~/OpenFaaS (bubbly-granite-362015)$ faas-cli new --lang node12 --prefix us.gcr.io/bubbly-granite-362015 main
Folder: main created.
  ___                   _____           ____
 / _ \ _ __   ___ _ __ |  ___|_ _  __ _/ ___|
| | | | '_ \ / _ \ '_ \| |_ / _` |/ _` \___ \
| |_| | |_) |  __/ | | |  _| (_| | (_| |___) |
 \___/| .__/ \___|_| |_|_|  \__,_|\__,_|____/
      |_|


Function created in folder: main
Stack file written: main.yml

Notes:
You have created a new function which uses Node.js 12.

npm i --save can be used to add third-party packages like request or cheerio
npm documentation: https://docs.npmjs.com/

Unit tests are run at build time via "npm run", edit package.json to specify
how you want to execute them.
```

Creating an empty nodeJS function.

OpenFaaS > main > handler.js > ...

```
1  'use strict'
2
3  module.exports = async (event, context) => {
4      var parameters=JSON.stringify(event.body)
5    return context
6      .status(200)
7      .succeed(parameters)
8  }
```

Updating handler.js.

```
slikcaustic1@cloudshell:~/OpenFaaS (bubbly-granite-362015)$ cd ~/OpenFaaS
faas-cli build -f main.yml
[0] > Building main.
Image: us.gcr.io/bubbly-granite-362015/main:latest built.
[0] < Building main done in 21.54s.
[0] Worker done.

Total build time: 21.54s
slikcaustic1@cloudshell:~/OpenFaaS (bubbly-granite-362015)$
```

Building docker image

```
slikcaustic1@cloudshell:~/OpenFaaS (bubbly-granite-362015)$ docker push us.gcr.io/bubbly-granite-362015/main
Using default tag: latest
The push refers to repository [us.gcr.io/bubbly-granite-362015/main]
bebb255a5b67: Pushed
8e9fd02b7432: Pushed
49ed69b379c6: Pushed
ff159fd1cd47: Pushed
1b83c76eb93a: Pushed
236f14b62181: Pushed
2de17cee83f1: Pushed
5149a82bab67: Pushed
d86bda25f4c1: Pushed
2d1fd3cd4c03: Pushed
b2844da8506c: Pushed
7f30cde3f699: Layer already exists
fe810f5902cc: Layer already exists
dfd8c046c602: Layer already exists
4fc242d58285: Layer already exists
latest: digest: sha256:518b8d4906714eea97df8cbb72ded12eeff7beec1b262e46bc30c87b2b39b4a2 size: 3659
```

Pushing the created image to the registry.

```
slikcaustic1@cloudshell:~/OpenFaaS (bubbly-granite-362015)$ 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://34.152.43.176:8080/function/main
```

Deploying the image to openfaas.

```
slikcaustic1@cloudshell:~/OpenFaaS (bubbly-granite-362015)$ curl http://34.152.43.176:8080/function/main -H 'Content-Type: application/json' -d '{ "Name": "Square", "Color": "Red", "Dimensions": 2 }'
{"Name":"Square","Color":"Red","Dimensions":2}slikcaustic1@cloudshell:~/OpenFaaS (bubbly-granite-362015)$
```

Sending a JSON object, and that same JSON object is echoed back.

```
slikcaustic1@cloudshell:~/OpenFaaS (bubbly-granite-362015)$ cd ~/OpenFaaS
faas-cli new --lang node12 --prefix us.gcr.io/bubbly-granite-362015 decorator
Folder: decorator created.
  ___                   _____           ____
 / _ \ _ __   ___ _ __ |  ___|_ _  __ _/ ___|
| | | | '_ \ / _ \ '_ \| |_ / _` |/ _` \___ \
| |_| | |_) |  __/ | | |  _| (_| | (_| |___) |
 \___/| .__/ \___|_| |_|_|  \__,_|\__,_|____/
      |_|


Function created in folder: decorator
Stack file written: decorator.yml

Notes:
You have created a new function which uses Node.js 12.

npm i --save can be used to add third-party packages like request or cheerio
npm documentation: https://docs.npmjs.com/

Unit tests are run at build time via "npm run", edit package.json to specify
how you want to execute them.
```

Creating a new empty nodeJS function.

```
const request = require('sync-request');

module.exports = async (event, context) => {
    var obj = event.body;
    if (obj['Name'] === undefined) {
        obj['Name'] = 'Nameless';
    }
    if (obj['Color'] === undefined) {
        obj['Color'] = 'Transparent';
    }
    var res = request('POST', 'http://34.152.43.176:8080/function/main', {

        body: JSON.stringify(obj)

    });

    console.log(res["body"].toString())
    return context.status(200).succeed(res["body"].toString('utf8', 1, res["body"].length-1).replace(/\\\"/g,'\"'));
```

Updating the handler.js file in the decorator directory.

```
{
    "name": "openfaas-function",
    "version": "1.0.0",
    "description": "OpenFaaS Function",
    "main": "handler.js",
    ▷ Debug
    "scripts": {
      "test": "echo \"Error: no test specified\" && exit 0"
    },
    "keywords": [],
    "author": "OpenFaaS Ltd",
    "license": "MIT",
    "dependencies": {
      "sync-request": "^6.1.0"
    }
  }
```

Adding sync requests to package.json.

```
slikcaustic1@cloudshell:~/OpenFaaS (bubbly-granite-362015)$ cd ~/OpenFaaS
faas-cli build -f decorator.yml
[0] > Building decorator.
Image: us.gcr.io/bubbly-granite-362015/decorator:latest built.
[0] < Building decorator done in 12.11s.
[0] Worker done.

Total build time: 12.11s
```

Building the decorator docker image.

```
slikcaustic1@cloudshell:~/OpenFaaS (bubbly-granite-362015)$ docker push us.gcr.io/bubbly-granite-362015/decorator
Using default tag: latest
The push refers to repository [us.gcr.io/bubbly-granite-362015/decorator]
6245ae223a34: Pushed
49daf6f97ccf: Pushed
5013d0f17a90: Pushed
3326f6520e9b: Pushed
1b83c76eb93a: Layer already exists
236f14b62181: Layer already exists
2de17cee83f1: Layer already exists
5149a82bab67: Layer already exists
d86bda25f4c1: Layer already exists
2d1fd3cd4c03: Layer already exists
b2844da8506c: Layer already exists
7f30cde3f699: Layer already exists
fe810f5902cc: Layer already exists
dfd8c046c602: Layer already exists
4fc242d58285: Layer already exists
latest: digest: sha256:a8d28b3c5138b9f697c1121413aaf8f5b38ecc4f6063ad1f0fd8f3e63fe934e5 size: 3663
```

Pushing the image to the registry.

```
slikcaustic1@cloudshell:~/OpenFaaS (bubbly-granite-362015)$ faas-cli deploy -f decorator.yml
Deploying: decorator.
WARNING! You are not using an encrypted connection to the gateway, consider using HTTPS.

Deployed. 202 Accepted.
URL: http://34.152.43.176:8080/function/decorator
```

Deploying the decorator to openfaas.

```
slikcaustic1@cloudshell:~/OpenFaaS (bubbly-granite-362015)$ curl http://34.152.43.176:8080/function/decorator -H 'Content-Type: application/json' -d '{ "Name": "Square", "Dimensions": 2 }'
{"Name":"Square","Dimensions":2,"Color":"Transparent"}slikcaustic1@cloudshell:~/OpenFaaS (bubbly-granite-362015)$
```

Now when a JSON object with the attributes of shape, and dimensions, the response will now contain a 3rd attribute of colour. The output reflects this and is functioning correctly.

**Discussion:**

The problem is that distributed applications need to be examined to see if they are working properly. This is known as health monitoring in this pattern. Through the use of health monitoring the checks are done with amalgamating two attributes. The first being the health checks that are done when requested by the health verification endpoint. The second being the analysis of what is returned by the check. Also assisting in the solution of this problem is there are tools and services that assist in auditing these web applications, and it is simple to make these tools/services endpoints that have the job to check the functionality of the system. Below is a list of checks that these services/tools perform:

- **Measuring response times:** The sum of the network latency, and the time that the service or application performs the request. If the sum is high, then there could be a potential problem on the network side (load balancing, traffic), or unoptimized usage of message transfer protocols.
- **Validating the response code:** Ensure that the correct response code is returning like 200 for OK.
- **Checking the content of the response:** Checking if the returned value of the response code 200 (OK) or something else. Also if the value is 20 it will still check other aspects of

the page that will verify if the returned webpage is correct like a test phrase on the page.

- **Checking resources/services:** Checks the content delivery NW that is utilized to send content from global caches.
- **Checking expiration of SSL certs:** Checks if the SSL certificate is expired or not.
- **Measuring response time of DNS lookup:** Looks up the URL for the service/application being used to see if there is DNS failure, or check the response time.
- **Validating URL from the DNS lookup response:** To stop malicious redirections of requests if there is an attack aimed at the DNS server.

The Above checks also are requirements for the system to assure that it is working as intended.

The problem is that distributed applications and services need to be analyzed to see if they are working properly. This is done through health monitoring. The tools/services assist in the auditing process of these applications, and it is simple to create these tools/services. But in order to create these tools there are some requirements that need to be followed. Like Measuring response times, validating the response codes, checking the content of the response, checking the resources/services, checking the expiration of SSL certificates, measuring the response time of DNS lookup, and validating URL from the DNS lookup.

In part 2 the requirements are followed as well through the use of checking if there are errors in the system using endpoints created in the server.js. The errors in the system are detected using these endpoints and the requests are sent elsewhere, where they can be dealt with instead of bombarding the erroneous part indefinitely. Some of the requirements are used in part 3 where the handler for both the decorator and the main, are sending a successful response code when the operation is done.

Requirements in part 2 are met as this implementation functions as a health endpoint monitor. Part 3 meets its requirements as the decorator implemented transforms the inputs that are sent to the function, where the decorator checks if the request was missing, and then adds missing fields with the set default value to the request that was sent from the user.

**Design:**

Persistent volumes are an important feature as if persistent volumes are not implemented, any of the data that has been created, and or modified when the container is running will be lost. But with the use of K8s persistent volumes (PV), data to be stored after the container is no longer running. PVs allow storage to be mounted to a K8, which additionally allows the data to be shared among different nodes. PVs are running for as long as operations are conducted, which can outlive the container. PVs can be implemented through the use of PersistentVolume which uses storage classes, which are like a resource that is a part of the cluster. PersistentVolumeClaim is also used which will request storage by a user in need. The PersistentVolumeClaim uses PersistentVolume resources like RAM, and CPU attention; they can also be mounted with ReadWriteMany, ReadWriteOnce. A Kubernetes guide was used to assist in understanding Persistent Volumes [1].

```
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
slikcaustic@cloudshell:~ (wise-brook-366921)$ minikube ssh
docker@minikube:~$ sudo mkdir /mnt/data
docker@minikube:~$ sudo sh -c "echo 'Hello from Kubernetes storage' > /mnt/data/index.html"
docker@minikube:~$ cat /mnt/data/index.html
Hello from Kubernetes storage
docker@minikube:~$
```

Start minikube, and enter into minikube using ssh, create the directory /mnt/data/ and create a file index.html with a message that will be used to save later.

pvvolume.yaml

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pervol
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/mnt/data"
```

Pv.yaml file. Kind sets it to a persistent volume. Setting the name to pv-volume. The storageClassName is set to manual which binds the claim requests to the now created PV. It also sets the size to 5GB. Additionally, the access mode is set to read read and write.

```
slikcaustic@cloudshell:~ (wise-brook-366921)$ kubectl apply -f pvvolume.yaml
persistentvolume/pervol created
slikcaustic@cloudshell:~ (wise-brook-366921)$ kubectl get pv
NAME     CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                 STORAGECLASS   REASON   AGE
pervol   5Gi        RWO            Retain           Bound    default/pervolclaim   manual                  12m
```

Creating the persistent volume.

pvclaim.yaml

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

Set the kind to be PersistentVolumeClaim. This requests the volume of 3GB which will provide read and write for a single node.

```
slikcaustic@cloudshell:~ (wise-brook-366921)$ kubectl apply -f pvclaim.yaml
persistentvolumeclaim/pervolclaim created
```

Creating pvclaim.

```
slikcaustic@cloudshell:~ (wise-brook-366921)$ kubectl get pvc
NAME           STATUS   VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE
pervolclaim    Bound    pervol   5Gi        RWO            manual         8m32s
```

Display the persistent volume claims.

pvpod.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pervolpod
spec:
  volumes:
    - name: pervolstorage
      persistentVolumeClaim:
        claimName: pervolclaim
  containers:
    - name: pervolcontainer
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: pervolstorage
```

Pvpod.yaml will create the pod that is going to be used by the claim pv that was previously created. In this case the claim is a volume.

```
slikcaustic@cloudshell:~ (wise-brook-366921)$ kubectl apply -f pvpod.yaml
pod/pervolpod created
```

Create the pvpod.

```
slikcaustic@cloudshell:~ (wise-brook-366921)$ kubectl exec -it pervolpod -- /bin/bash
root@pervolpod:/# curl http://localhost
Hello from Kubernetes storage
```

Open a shell that is running the pod. Check the output of the contents that are in the index.html file that was created using minikube. Since the message was the same as the message initially created earlier then the implementation of persistent volume is working correctly.

**Conclusion:**

After completing the lab's steps we now know and are comfortable with creating NodeJS services and Docker images. We are now familiar with Function as a Service (FaaS) and the configuration of nginx and circuit breakers. Through the completion of these steps we are now able to configure OpenFaaS on the Google Cloud Platform, and are now familiar with the Decorator Pattern that was implemented, and could now understand other implementations.

Works Cited

[1] “Configure a pod to use a persistentvolume for storage,” *Kubernetes*, 21-Sep-2022. [Online]. Available: https://kubernetes.io/docs/tasks/configure-pod-container/configure-persistent-volume-storage/. [Accessed: 27-Oct-2022].

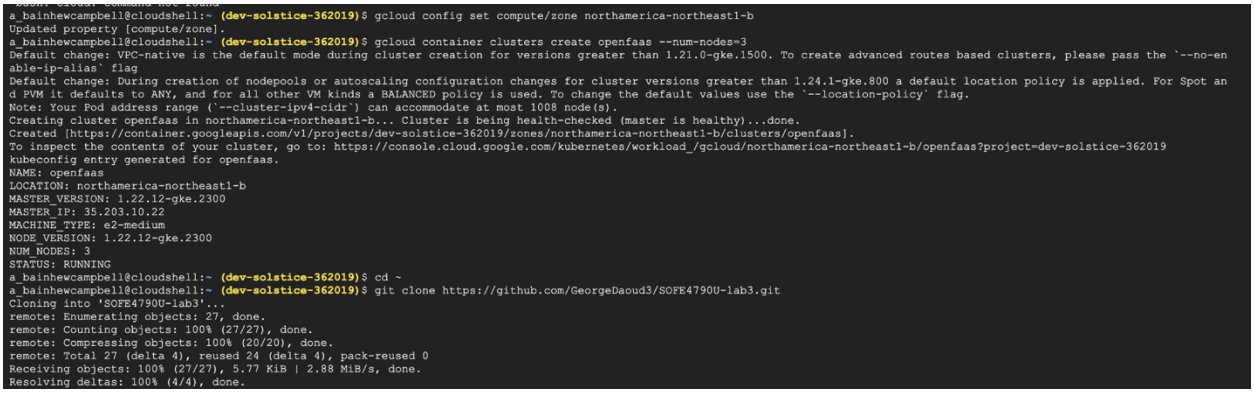
**Lab 3: Deploying a Circuit Breaking ambassador and FaaS**
**Member: Alexander Campbell**

## Introduction:

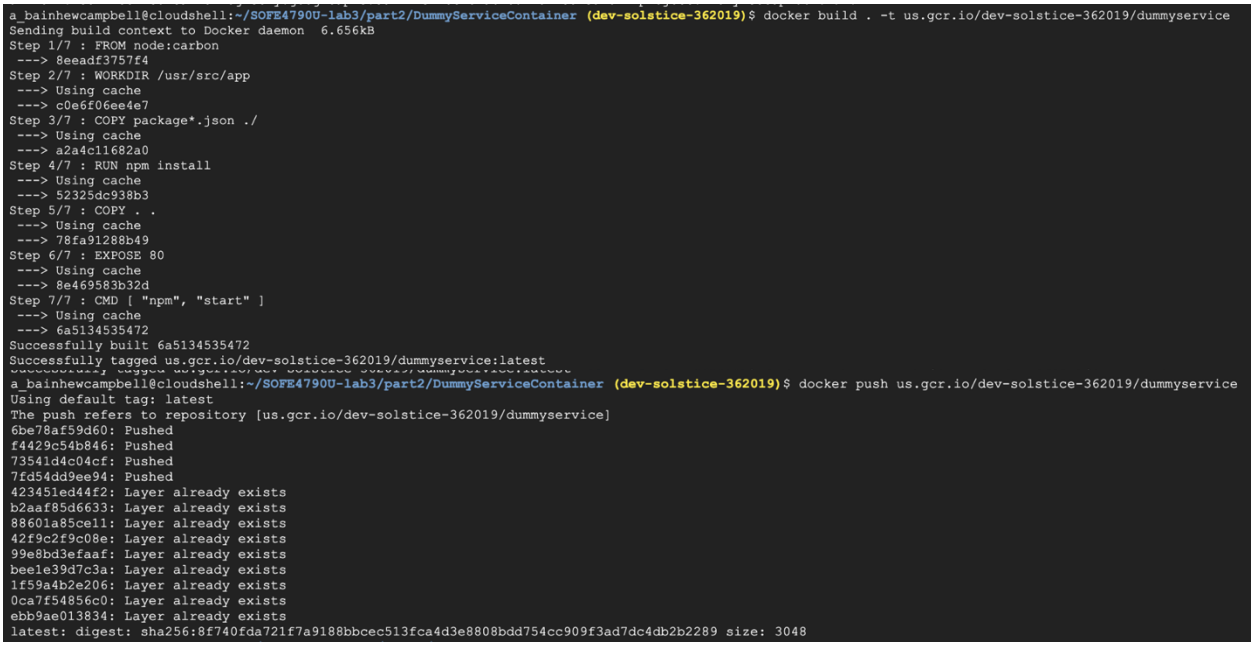
This lab gave us an introduction to deploying a circuit breaker ambassador and utilizing the Function as a Service. Below is the detailed description of what happened as I was unable to record due to completing this lab in a busy environment.

## Part 2 Procedure:

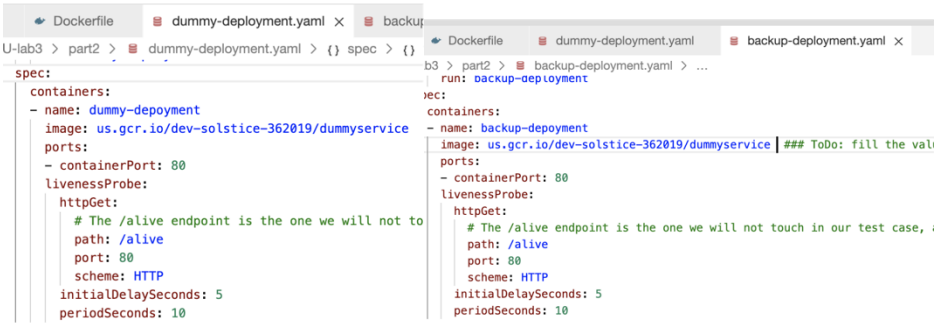
First I did the initial setup of the clusters and then cloned in the required Github repository




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



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



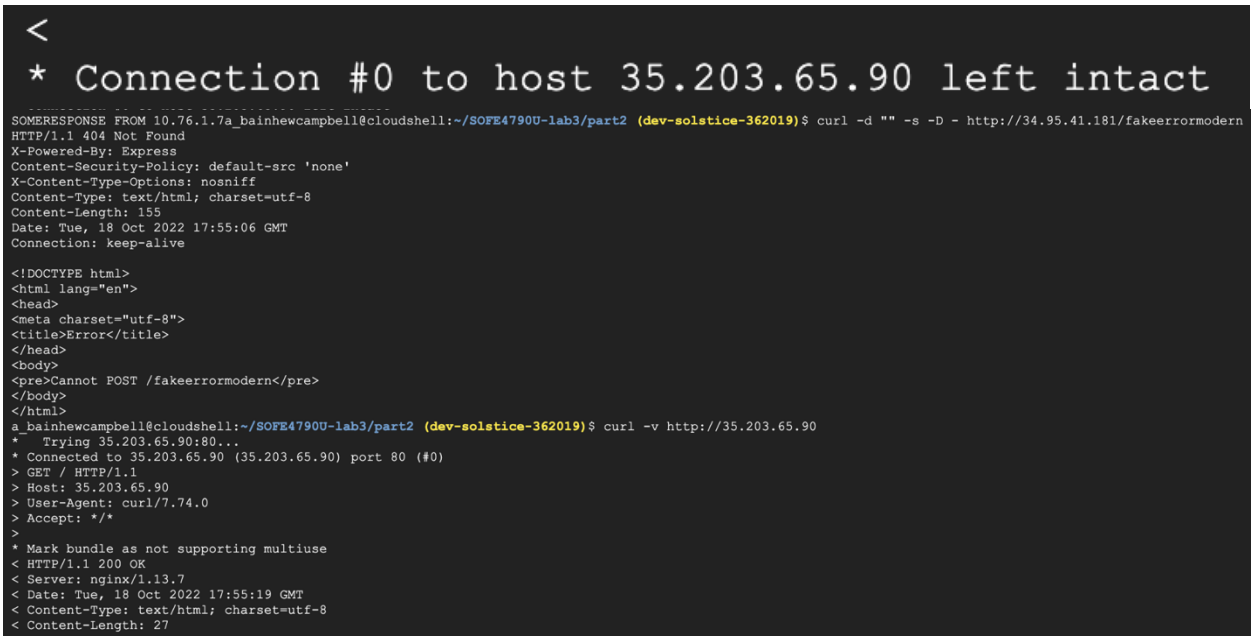

```
a_bainhewcampbell@cloudshell:~/SOFE4790U-lab3/part2 (dev-solstice-362019)$ kubectl create -f dummy-deployment.yaml
deployment.apps/dummy-deployment created
a_bainhewcampbell@cloudshell:~/SOFE4790U-lab3/part2 (dev-solstice-362019)$ kubectl expose deployment dummy-deployment --port=80 --type=LoadBalancer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

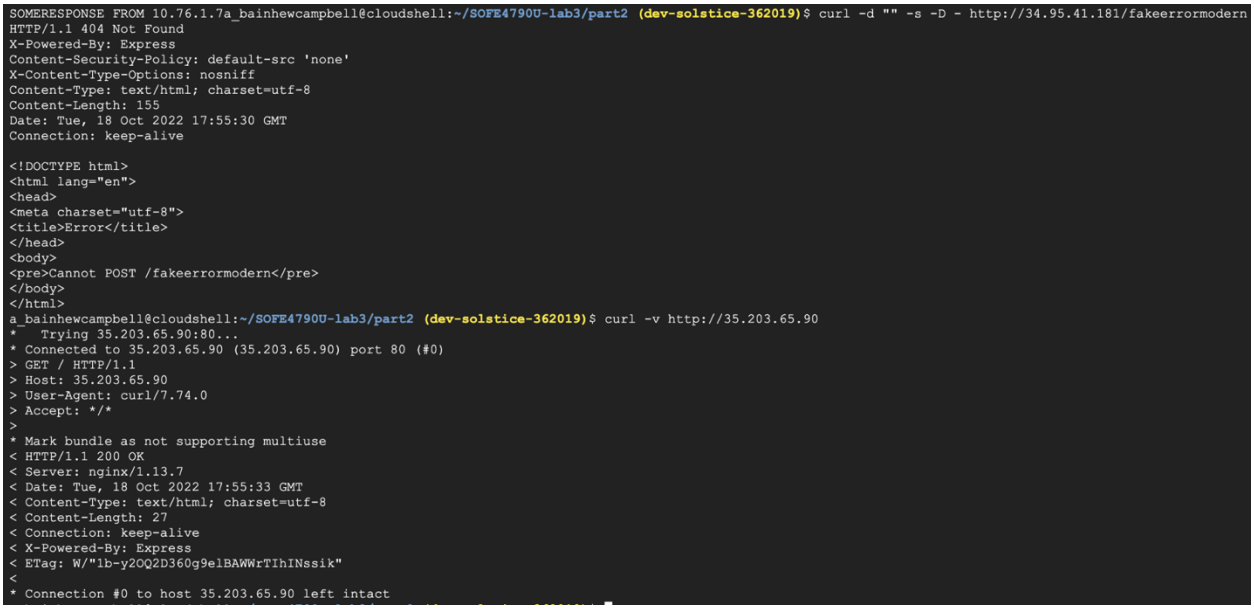
The circuit breaker was then tested

```
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
```

```
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
```



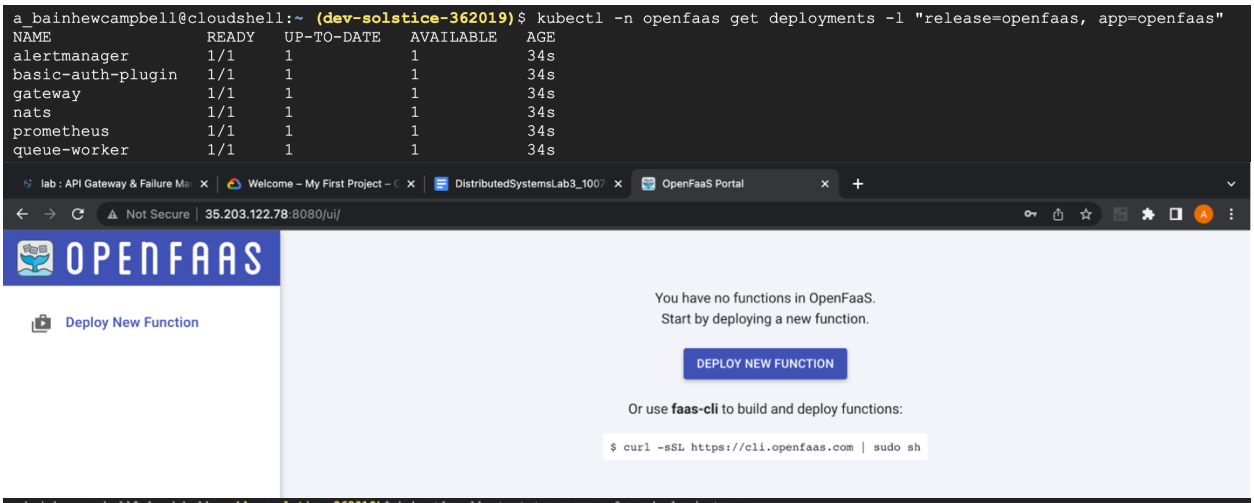

The error was reset and curl was run one last time

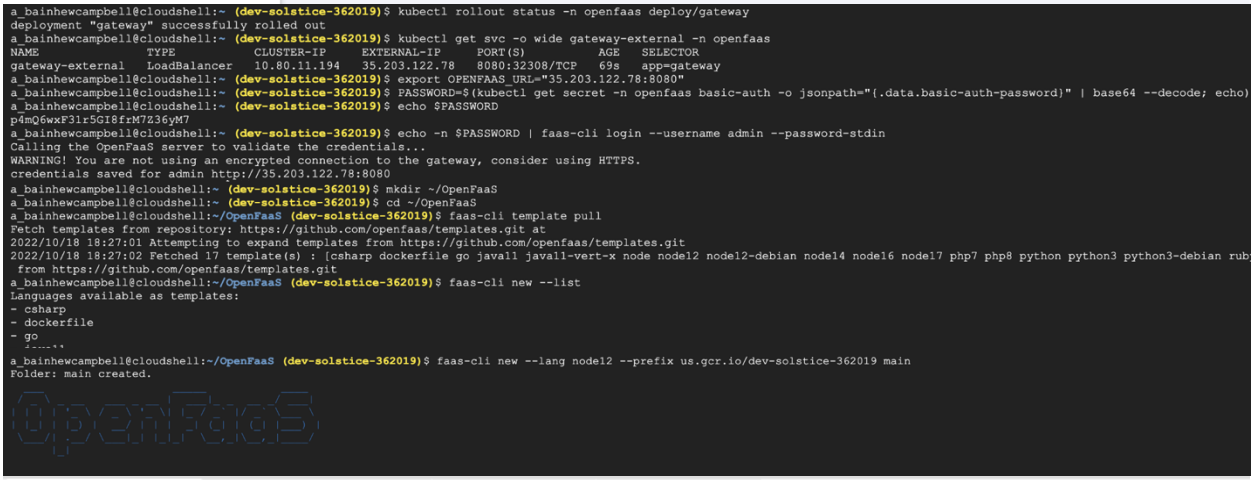


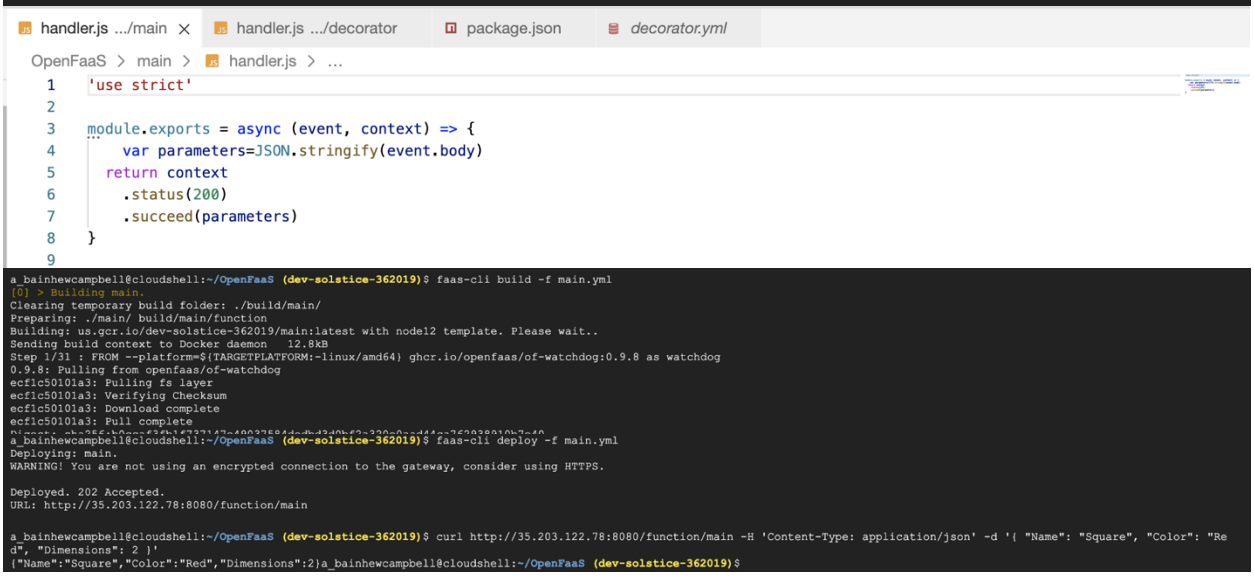

**Part 3 Procedure:**

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

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







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

```
[0] < Building decorator done in 7.92s.
[0] Worker done.

Total build time: 7.92s
a_bainhewcampbell@cloudshell:~/OpenFaaS (dev-solstice-362019)$ docker push us.gcr.io/dev-solstice-362019/decorator
Using default tag: latest
The push refers to repository [us.gcr.io/dev-solstice-362019/decorator]
12db82892bab: Pushed
b48deddeee7f: Pushed
26251b4bd202: Layer already exists
907b6d15636b: Layer already exists
cfe9944e75b2: Layer already exists
5533a2852e19: Layer already exists
09191fcf38e7: Layer already exists
80dc630f97ac: Layer already exists
d86bda25f4c1: Layer already exists
756f0cebffd0: Layer already exists
a_bainhewcampbell@cloudshell:~/OpenFaaS (dev-solstice-362019)$ faas-cli deploy -f decorator.yml
Deploying: decorator.
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/decorator

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

lab : API Gateway & F | DistributedSystemsLa | Welcome – My First P | 35.203.122.78:8080/

Not Secure | 35.203.122.78:8080/function/decorator

{"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.

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.

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

```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"
```

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:/#
```

**Lab 3: Deploying a Circuit Breaking ambassador and FaaS**
**Member: Mamun Hossain**

## Introduction:

This lab gives us an introduction to understand how to deploy a circuit breaker file and utilize the function as a service. We have also learned how to access FaaS services via the UI and invoke the requests accordingly.

## Part 1: How to solve the problem and the requirements:

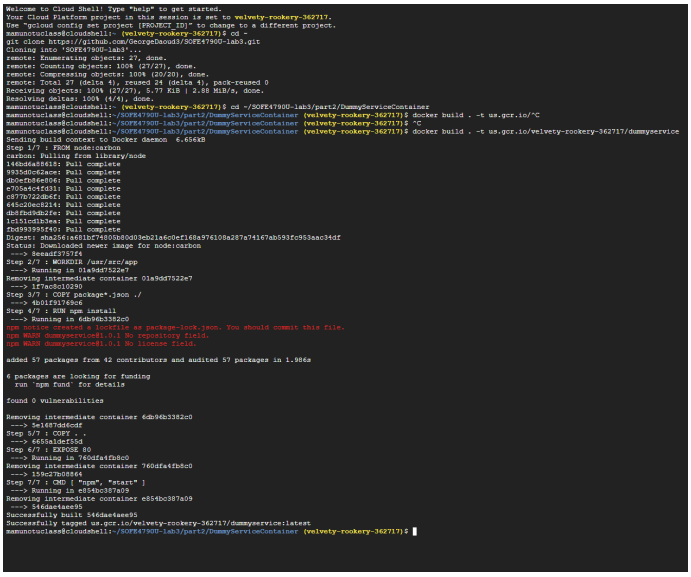
The problem at hand was the many factors that had affected cloud-hosted applications, and those factors were such as network latency, performance and availability of the underlying compute and storage systems and the network between them.

The way this problem was solved was by implementing health monitoring of the clusters and the nodes to each endpoint of the application. It will check if the application or services respond to the request of the health monitor and it will also analyze the endpoint of the tool or framework that performs the check.

The requirements to do this health monitor check is to implement an Agent that will have the appropriate certificates and in doing so, will send pings to the application endpoint to ensure that over the network everything is working, and will check for 200 (OK) status to be returned to ensure data errors are minimized.

## Part 2: Procedure:

Here we can see that we initialize the setup with the required clusters and then clone the repository and input the dummy service containers and services and then push them with the docker command.



Here we are able to see the projects list to see what is in the gcloud.

Afterwards we can push it with the docker command to see what layers need to be pushed.

```
Successfully tagged us.gcr.io/velvety-rookery-362717/dummyservice:latest
mamunotuclass@cloudshell:~/SOFE4790U-lab3/part2/DummyServiceContainer (velvety-rookery-362717)$ gcloud projects list
PROJECT_ID: mamunhossain
NAME: MamunHossain
PROJECT_NUMBER: 418585279789

PROJECT_ID: sunny-effort-366715
NAME: SOFE4640 GoogleMapKey
PROJECT_NUMBER: 685349366104
```

```
PROJECT_ID: velvety-rookery-362717
NAME: My First Project
PROJECT_NUMBER: 38427508957
mamunotuclass@cloudshell:~/SOFE4790U-lab3/part2/DummyServiceContainer (velvety-rookery-362717)$ docker push us.gcr.io/velvety-rookery-362717/dummyservice
Using default tag: latest
The push refers to repository [us.gcr.io/velvety-rookery-362717/dummyservice]
fdb15b87583c: Pushed
abd302d5b471: Pushed
5f873b52387a: Pushed
17e0dfafdf13: 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:a70f5b4579e2eb15b7dedeb145f33015a63db97bfb34533bfbf770d151aa38e6 size: 3048
mamunotuclass@cloudshell:~/SOFE4790U-lab3/part2/DummyServiceContainer (velvety-rookery-362717)$
```

We can now expose the deployments after we create the backup deployments and while we go into the editor to make the required adjustments to the files provided in the git clone.

Altered files for backup-deployment.yaml

```
SOFE4790U-lab3 > part2 > backup-deployment.yaml > ...
12      run: backup-deployment
13    plate:
14    metadata:
15      labels:
16        run: backup-deployment
17    spec:
18      containers:
19      - name: backup-depoyment
20        image: us.gcr.io/vast-alcove-367116/dummyservice
21        ports:
22        - containerPort: 80
23        livenessProbe:
24          httpGet:
25            # The /alive endpoint is the one we will not touch in our test case, a
26            path: /alive
27            port: 80
```

Here, we can see after we create the configmap, we can use the curl command to execute the circuit breaker UP and check it in the local machine in the last line of each response.

While we see the last line remaining unchanged after each test, but the dummy service having a different line in each response, check test the circuit breaker one more time.

## Part 3:

We initialized the cluster admin role binding to then deploy OpenFaaS through the GKE, afterwards installing FaaS and as we can see below the confirmation of that and ArKade. We then use the commands to test that FaaS has been successfully initialized.

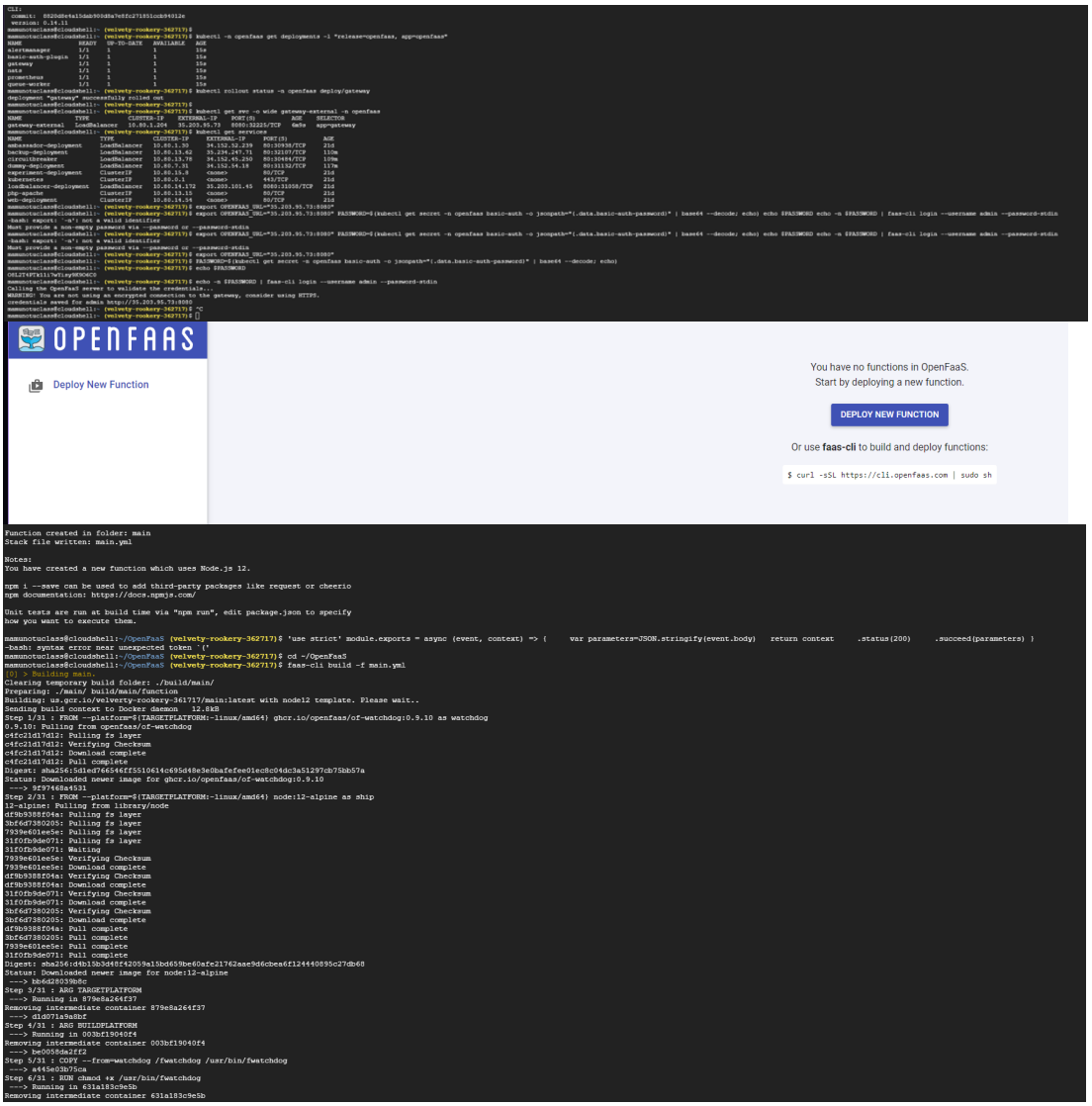
The installation process of FaaS and ArKade.

Successful verification of FaaS.


Using the instructions, we will use the commands to find out the password to our localized FaaS UI and from there we will open it after the installation.

Verification of successful entry into FaaS UI.

After verification, we can go into the editor and alter the main/handler.js file to convert the event.body parameters of the JSON object to a string, and get the command that we want.




Here we are building the main.yml file after editing the FaaS main/handler.js file.

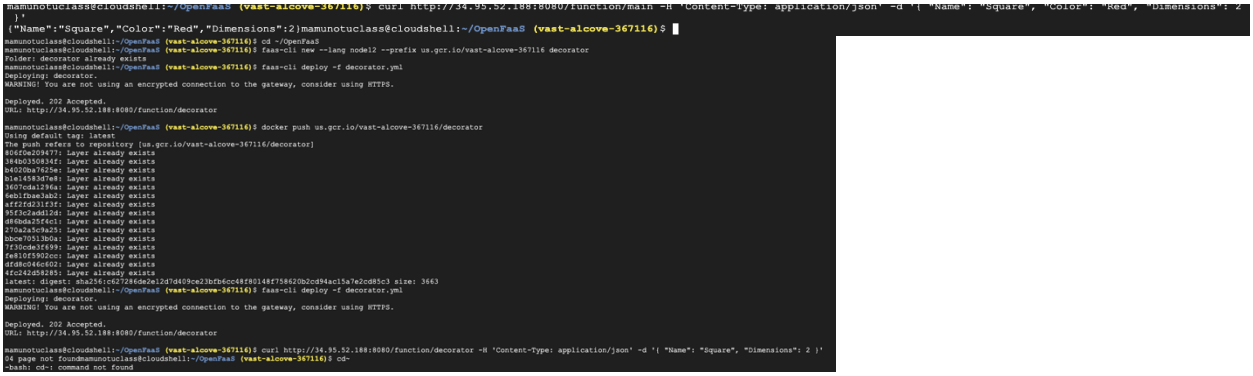
```
Digest: sha256:d4b15b3d48f42059a15bd659be60afe21762aae9d6cbea6f124440895c27db68
Status: Downloaded newer image for node:12-alpine
 ---> bb6d28039b8c
Step 3/31 : ARG TARGETPLATFORM
 ---> Running in 879e8a264f37
Removing intermediate container 879e8a264f37
 ---> d1d071a9a8bf
Step 4/31 : ARG BUILDPLATFORM
 ---> Running in 003bf19040f4
Removing intermediate container 003bf19040f4
 ---> be0058da2ff2
Step 5/31 : COPY --from=watchdog /fwatchdog /usr/bin/fwatchdog
 ---> a445e03b75ca
Step 6/31 : RUN chmod +x /usr/bin/fwatchdog
 ---> Running in 631a183c9e5b
Removing intermediate container 631a183c9e5b
 ---> dcb64a05fe38
Step 7/31 : RUN apk --no-cache add curl ca-certificates     && addgroup -S app && adduser -S -g app app
 ---> Running in 81e4b0a9eef0
fetch https://dl-cdn.alpinelinux.org/alpine/v3.15/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.15/community/x86_64/APKINDEX.tar.gz
(1/5) Installing ca-certificates (20220614-r0)
(2/5) Installing brotli-libs (1.0.9-r5)
(3/5) Installing nghttp2-libs (1.46.0-r0)
(4/5) Installing libcurl (7.80.0-r4)
(5/5) Installing curl (7.80.0-r4)
Executing busybox-1.34.1-r5.trigger
```

Edited OpenFaaS handler.js file.

We then run the command and get the string input as desired

After repeating the same processes as the previous task, we defined the decorator with the file provided, making minor alterations with the IP and the file itself to make it work and give the desired output shown below.

```
OpenFaaS > main > handler.js > ...
1   'use strict'
2
3   module.exports = async (event, context) => {
4       var parameters=JSON.stringify(event.body)
5     return context
6       .status(200)
7       .succeed(parameters)
8   }
```




File alteration of decorator/handler.js.

## Discussion:

In summary, the main problem at hand was to understand how latency works and how the applications need to have health monitoring checks to ensure that it is working to its highest functionality. We need to understand that we cannot always monitor the hardware in the system for every device, as that can end up being costly and wasting time in the long run. With the health monitoring system we had discussed in part 1, we can use it effectively to ensure that we can understand and monitor the network latency, performance and availability of the underlying compute and storage systems with the networks between it.

We will be gathering different values and monitor the health checks as we did in the lab. In summary for part 1, we were able to confirm the need for implementing health monitoring of clusters and the nodes to each endpoint of the application to ensure that all the checkpoints were in place to monitor the node health. The way we were able to accomplish this with part 2 and 3 was by implementing several services: Backup service, implemented service, and a circuit breaker. We had used the circuit breaker several times and then paired it with a dummy service to ensure that the circuit breaker was working as it was intended to do. The way this

would work is that the circuit breaker would be tested to make sure that the endpoints have the same value so it can be deemed to be a healthy endpoint.

For part 3, we had used the decorator function and the main function to distinguish the difference between the two and their interactions with the application and to view the values that get sent to the applications via JSON files that get converted into strings. The decorator endpoint ensures the validity of the checkpoint for the health monitoring by responding to any errors in the system and fixing them accordingly.

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

```
'use strict'
const request = require('sync-request');

module.exports = async (event, context) => {
    var obj = event.body;
    if (obj['Name'] === undefined) {
        obj['Name'] = 'Nameless';
    }
    if (obj['Color'] === undefined) {
        obj['Color'] = 'Transparent';
    }
    var res = request('POST', 'http://34.95.52.188:8080/function/decorator', {
        body: JSON.stringify(obj)
    });
    console.log(res["body"].toString())
    return context.status(200).succeed(res["body"].toString('utf8', 1, res["body"].length-
1).replace(/\\\"/g,'\"'));
}
```

## Design:

As we know, Kubernetes provides persistent volumes and it is necessary to have these volumes in a distributed system because if these volumes were not present, any data that has been previously created or altered will become null and void when the container restarts its processes. An example of when persistent volumes are needed is when let's say for example you want to be a developer at an enterprise, you want to make sure that your project is scalable. This way, if the consumer requests for something that can have several new environments to ensure that the product they want can run in different environments to their hearts desire. As we can see below, we will show how to configure a YAML file to implement the example and then running it and test it with persistent volume. We will be using “minikube” based off a guide we had found on the Internet doing our own research.

We install minikube using the commands and then check for all the pods to be working to make sure that the application will be working.

```
mamunotuclass@cloudshell:~ (vast-alcove-367116)$ minikube start
* minikube v1.27.0 on Debian 11.5 (amd64)
  - MINIKUBE_FORCE_SYSTEMD=true
  - MINIKUBE_HOME=/google/minikube
  - MINIKUBE_WANTUPDATENOTIFICATION=false
! Kubernetes 1.25.0 has a known issue with resolv.conf. minikube is using a workaround that should work for most use cases.
! For more information, see: https://github.com/kubernetes/kubernetes/issues/112135
* Using the docker driver based on existing profile
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
* Updating the running docker "minikube" container ...
* Preparing Kubernetes v1.25.0 on Docker 20.10.17 ...
  - kubelet.cgroups-per-qos=false
  - kubelet.enforce-node-allocatable=""
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
mamunotuclass@cloudshell:~ (vast-alcove-367116)$ kubectl get pods
No resources found in default namespace.
mamunotuclass@cloudshell:~ (vast-alcove-367116)$ kubectl get pods -A
NAMESPACE     NAME                               READY   STATUS    RESTARTS      AGE
kube-system   coredns-565d847f94-n6fwd           1/1     Running   1 (57s ago)   3m36s
kube-system   etcd-minikube                      1/1     Running   2 (57s ago)   3m48s
kube-system   kube-apiserver-minikube            1/1     Running   2 (47s ago)   3m48s
kube-system   kube-controller-manager-minikube   1/1     Running   2 (57s ago)   3m48s
kube-system   kube-proxy-v9wj7                   1/1     Running   2 (57s ago)   3m37s
kube-system   kube-scheduler-minikube            1/1     Running   1 (62s ago)   3m48s
kube-system   storage-provisioner                1/1     Running   3 (43s ago)   3m46s
mamunotuclass@cloudshell:~ (vast-alcove-367116)$ minikube dashboard
* Enabling dashboard ...
  - Using image docker.io/kubernetesui/metrics-scraper:v1.0.8
  - Using image docker.io/kubernetesui/dashboard:v2.6.0
* Verifying dashboard health ...
* Launching proxy ...
* Verifying proxy health ...
* Opening http://127.0.0.1:41731/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/ in your default browser...
  http://127.0.0.1:41731/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/
```

We can now enter the node and create persistent volumes as seen below that the pods can utilize as persistent volumes.

```
mamunotuclass@cloudshell:~ (vast-alcove-367116)$ kubectl apply -f https://k8s.io/examples/pods/storage/pv-volume.yaml
persistentvolume/task-pv-volume created
mamunotuclass@cloudshell:~ (vast-alcove-367116)$ kubectl apply -f https://k8s.io/examples/pods/storage/pv-claim.yaml
persistentvolumeclaim/task-pv-claim created
mamunotuclass@cloudshell:~ (vast-alcove-367116)$ kubectl apply -f https://k8s.io/examples/pods/storage/pv-pv-pod
error: unable to read URL "https://k8s.io/examples/pods/storage/pv-pv-pod", server reported 404 Not Found, status code=404
mamunotuclass@cloudshell:~ (vast-alcove-367116)$ kubectl apply -f https://k8s.io/examples/pods/storage/pv-pod.yaml
pod/task-pv-pod created
mamunotuclass@cloudshell:~ (vast-alcove-367116)$ kubectl get pvc task-pv-claim
NAME            STATUS   VOLUME           CAPACITY   ACCESS MODES   STORAGECLASS   AGE
task-pv-claim   Bound    task-pv-volume   10Gi       RWO            manual         45s
mamunotuclass@cloudshell:~ (vast-alcove-367116)$
```

From here, we can see the node being re entered by the command as seen below to ensure the persistent volume is being used as needed as shown created earlier into the node.

```
NAME            STATUS   VOLUME           CAPACITY   ACCESS MODES   STORAGECLASS   AGE
task-pv-claim   Bound    task-pv-volume   10Gi       RWO            manual         45s
mamunotuclass@cloudshell:~ (vast-alcove-367116)$ 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 [193 kB]
Get:6 http://deb.debian.org/debian bullseye-updates/main amd64 Packages [14.6 kB]
Fetched 8600 kB in 2s (4735 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:/#
```

Below are the yaml files in the editor

```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
```