Auto Zone Coding Challenge – Solution

Question1.

1. Deploy a simple microservices based application using any language of your choice/ You can
use publicly available demo apps that satisfies the following requirements.
a. An application that returns some data to https requests
b. Is highly available.
c. Make sure they have different services and can handle different requests based on path of requests e.g., abc.com/path1 should reach to service1 & abc.com/path2 should reach to service2.
d. Securely stores and accesses its web security certificate (this cert can be any dummy file)

Solution:

In the zipped directory, there are three separate directories:

1. **Basic**
2. **Istio**
3. **Helm**

Basic:

Code in basic directory is fully functional and it covers all the functions mentioned in above points of Question1. All codes executed successfully in minikube on my personal laptop.

I have set minikube-example.com as domain name for minikube ip on my laptop's host file. Below are the output from the different paths:

Minikube-example.com/service1 → Hello from service1

Minikube-example.com/service2→ Hello from service2

Public Repository, which I have created in Dockerhub is somusumit, same is mentioned in the code too.

Ssl-secret.yaml file is used to store security certificates.(It's the dummy one).

For the points:
2. Bonus:
a. Applications/pods should be self-healing in nature.
b. Pods should start to get traffic only once application is started & healthy.

Different probes are added in code, which are:

1. LivenessProbe
2. ReadinessProbe
3. StartupProbe

For the points:
c. Organize Kubernetes resource manifests using Kustomize and/or Helm.
d. Deploy the application into a service mesh.

Refer to directory helm, for implementation of code using helm. Code is fully functional

Refer to directory istio, where the deployment is done in Istio service mesh. Code is fully functional

Add a namespace label to instruct Istio to automatically inject Envoy sidecar proxies when you deploy the application

kubectl label namespace default istio-injection=enabled

I have injected it in default namespace only.

Terraform

Question 2.

Write a Terraform code to deploy Cloudfunction services into multiple regions following the best
practices. (Assume the code should be reusable for multiple apps & can support multiple environments)

For solution to above question, I have developed the code using terraform module.

By providing the values for credentials (path to service account json file) in main.tf in root directory, code can be executed.

Sample function is created in node. Function url should return:
Hello from Auto Zone

Multiple environments can be supported by using by referencing to different tfvars file according to environment. For example, the environment specific file which I have used here is environments/dev/dev.tfvars, which can be modified according to usage.