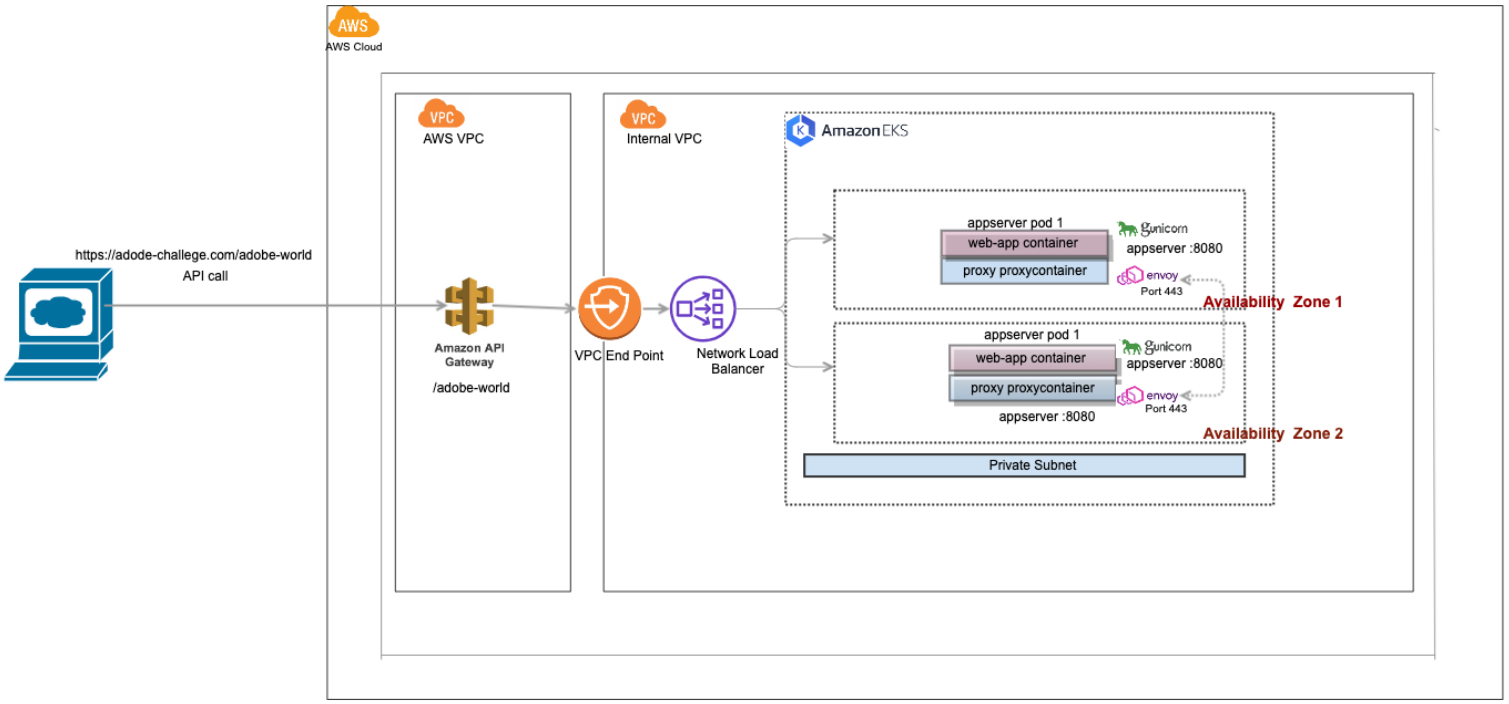
Challenge

Provide the design for an HTTP based service that returns the simple HTML message "Hello, Adobe!". This service will run in a container and should only expose ports needed to be accessed via http/https.

Solution

Create an EKS cluster with managed nodes. Then deploy “Hello, Adobe” applications and expose them using an internal Network Load Balancer for the application. Then, create a VpcLink, and create an API Gateway HTTP API with a route on.



EKS Cluster

Create AWS Virtual Private Cloud (VPC) and subnets across two Availability Zones, and Deploy EKS cluster. EKS compute nodes are in the private subnets and are accessible only via API Gateway. Gives more control to filter the unwanted traffic right at the entry point.

Application server

Application server pod runs with two containers, application server and proxy container

The application is running as an App server using “Gunicorn flask server”, running on port 8080. This server provide response “Hello Adobe” whenever you call the endpoint

istio proxy container running as service mesh, all the pod to pod traffic routed through this proxy.

Service endpoint “app-server-service.svc.service.local.8080” exposed for accessing from web server pod

Network LoadBalancer

AWS Network Load Balancer (NLB) is provisioned that load balances network traffic on web app server

AWS API Gateway

Create “AWS API Gateway” as a gateway application firewall and only allow access to API from a Virtual Private Cloud (VPC). Configure custom domains for APIs on Amazon API Gateway using SSL/TLS certificates provisioned and managed by AWS Certificate Manager (ACM).

Build and Deploy

App server build Docker file available at
https://github.com/rayees09/adobe-challenge/tree/main/challenge2/build

```
# git clone
# git checkout main
# cd challenge2/build
# docker build -t app-server .
```

You can wrap the deployment in HELM deployment for test and production deployment