Assessment submission

Github repos link:
1. https://github.com/shuvo85cstee/provision-gke-cluster
2. https://github.com/shuvo85cstee/simple-nodejs-app

First repository have the terraform code which I used to provision a fully private GKE cluster due to security concern.

Terraform workspaces have been used to provision this cluster as it allows to reuse/ remove duplications of terraform codes and allows to deploy different Kubernetes clusters that require different configuration values.

Terraform states are kept in GCS bucket to enable sharing, locking and versioning.

A jump host is provisioned using terraform to access private GKE clusters control plane and run kubectl commands from there.

This jump host also used as GitHub actions runners to facilitate CI/CD for the application which are pushed in this repo https://github.com/shuvo85cstee/simple-nodejs-app

Its a very simple node application which is deployed in GKE via CI/CD powered by Github actions. Helm is used to template the Kubernetes objects. All the codes are available in repo.

CI/CD has been configured using Github actions and details can be found here https://github.com/shuvo85cstee/simple-nodejs-app/tree/main/.github/workflows

A nginx proxy server has been provisioned using terraform and ansible -> https://github.com/shuvo85cstee/provision-gke-cluster/blob/main/infrastructure/nginx-proxy.tf

This nginx proxy proxy the traffic to GKE pods which are exposed via NodePort service. Nginx proxy will only allow request that have container specific header as http_x_case_study = “kadmos”
https://github.com/shuvo85cstee/provision-gke-cluster/blob/main/infrastructure/files/deploy_nginx_web/files/default

Jump host and nginx proxy both are running Debian 11 Operating have UFW enabled.

Further requirement analysis:
The service should be scaled if it receives more than 35 requests/second (the minimum pod count should be 1 and the maximum 11.

Kubernetes does not natively provide to auto scale resources based on http requests. To achieve this we need external metrics. Horizontal Pods Autoscaling (HPA) is a method of autoscaling in which we can supply our metrics based on what we want to increase our number of resources dynamically.

To autoscale application based on HTTP request , we need to capture the metrics of your HTTP requests from the application. For example for .net applications we can use Prometheus and instrumenting your application using appropriate metrics.

The application I used for assessment, does not have metrics exposed to autoscale based on request. I would need additional time to configure this. Currently resources based HPA has been enabled for application.

Self Signed SSL certificate:
I have provisioned a self sign certificate and used in nginx proxy to secure the connection. In ordet to make https work, both public certificate and private key need to reside in web server. The public certificate is public; no protection needed - server privileges or otherwise. Protection is required for private key.

Possible case for securing private key could be not keeping in default location, make sure only super admin (root) can have access of this and ssl-cert group have access to it. Also limiting the permission for private key like **chmod 600 xyz.key** will enhance security

Unusual traffic:
Unusual traffic is observed in the server logs of the cloud provider. One of the kubernetes services is sending many requests to an unknown ip address.

- What would you do in this scenario?
- What could you do to prevent such a scenario?

In response to unusual traffic observed in the server logs of the cloud provider specifically with one of the Kubernetes services sending many requests to an unknown IP address, here are steps to take and preventive measures to consider:

- Analyse the logs and identify the Kubernetes service responsible for the unusual traffic.Determine the source IP addresses and destination IP addresses involved.
- If the traffic appears to be malicious or unauthorised, isolate the affected Kubernetes service or pod to prevent further damage or spread of the issue.
- Temporarily block the unknown IP address or range of IP addresses from accessing the Kubernetes service or the entire cluster to mitigate the risk of further attacks.
- Review and update firewall rules to restrict traffic only to trusted sources and deny traffic from unauthorised or unknown IP addresses.
- Use Kubernetes Network Policies to define rules for controlling the flow of traffic to and from pods within the cluster. Restrict communication only necessary & trusted sources.
- Enable encryption for network traffic within the Kubernetes cluster using protocols like TLS to protect data in transit from eavesdropping or interception.
- Set up robust monitoring and alerting systems to detect and respond to unusual network activity in real-time. Configure alerts to notify administrators of suspicious patterns or anomalies.