SRE Image Resize Exercise

GKE Cluster:

I have created the GKE Cluster using github GKE Module repo and GCP best practices for creating the GKE.

References:

1. https://cloud.google.com/architecture/best-practices-for-running-cost-effective-kubernetes-applications-on-gke
2. https://cloud.google.com/kubernetes-engine/docs/best-practices/scalability
3. https://registry.terraform.io/providers/hashicorp/google/latest/docs/resources/container_cluster
4. https://github.com/terraform-google-modules/terraform-google-kubernetes-engine

Our helm chart will create the following resources. I have created the helm chart yaml using the GCP & kubernetes best practices which I have mentioned in each resource reference section.

01 BackendConfig:

Create a BackendConfig with CDN enable for caching the content on the google edge node. It will improve the performance of the app as well as reduce the cost of the server.

References:

1. https://cloud.google.com/kubernetes-engine/docs/how-to/ingress-features#cloud_cdn

02 Service:

Create a service with BackendConfig attached for accessing the deployment pods.

References:

1. https://kubernetes.io/docs/concepts/services-networking/service/
2. https://cloud.google.com/kubernetes-engine/docs/how-to/ingress-features#cloud_cdn

03 Deployment:

Create a deployment for pods with health check(liveness and readiness probe) and resources(request & limits of CPU & Memory). Main application will be deployed on this deployment pod.

References:

1. https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/

04 Ingress:

Create an ingress controller with Google cloud external load balancer and rules for backend service map.
References:

1. https://cloud.google.com/kubernetes-engine/docs/how-to/load-balance-ingress

05 HPA:

Create a HPA with server & custom-metric which we can fetch from Prometheus and stackdriver using adapter. If we enable custom-metric then we have to provide the adapter for prometheus or stackdriver.
References:

1. https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale-walkthrough/#autoscaling-on-multiple-metrics-and-custom-metrics
2. https://cloud.google.com/kubernetes-engine/docs/tutorials/autoscaling-metrics#custom-metric_1
3. https://github.com/GoogleCloudPlatform/k8s-stackdriver/tree/master/custom-metrics-stackdriver-adapter
4. https://github.com/kubernetes-sigs/prometheus-adapter/blob/master/docs/config.md

Monitoring:

We will monitor our cluster with prometheus and grafana. Prometheus is an open-source application used for metrics-based monitoring and alerting. It calls out to your application, pulls real-time metrics, compresses and stores them in a time-series database.
helm install prometheus stable/prometheus-operator

Load Testing:

Load testing will be performed with Jmeter.
References:

1. https://jmeter.apache.org/
2. https://www.guru99.com/jmeter-performance-testing.html