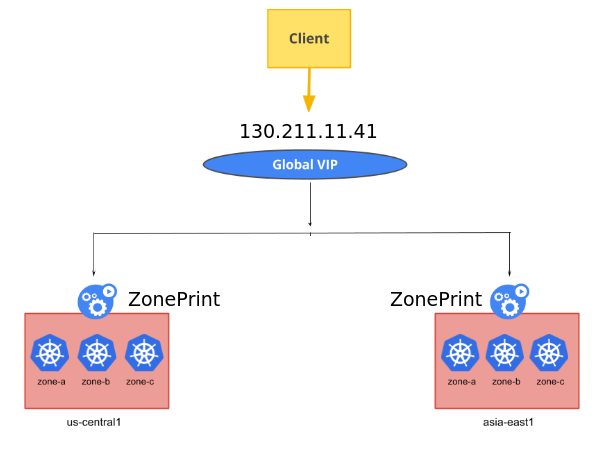
**Deploy Zoneprinter to Kubernetes federation**

Why would I want to use Federation anyway?

High Availability -> Achieve true high availability by deploying multiple clusters into different AZs/Regions/Cloud Providers instead of simply increasing the number of instances of your single cluster.

Geographic Coverage -> One of the huge advantages of using Federation, and the one that we're going to tackle in this tutorial too, is reducing latency for end-users by spreading the load of your application into different endpoints across the globe.

It is accessible at <http://35.241.12.227/>



==================**Create global DNS Zone** ===============================

gcloud dns managed-zones create gfed --description "Kubernetes Federation Zone" --dns-name devops.vishalvyas.com

gcloud dns managed-zones describe gfed

=====================================================================

=====**Create Cluster in Asia,points kubectl to it, user to the cluster-admin role** =======

gcloud container clusters create asia --zone asia-southeast1-a --scopes cloud-platform

gcloud container clusters get-credentials asia --zone asia-southeast1-a

kubectl create clusterrolebinding cluster-admin-binding --clusterrole cluster-admin --user $(gcloud config get-value account)

=====================================================================

======**Create Cluster in America,points kubectl to it, user to the cluster-admin role** ====

gcloud container clusters create america --zone us-central1-a --scopes cloud-platform

gcloud container clusters get-credentials america --zone us-central1-a

kubectl create clusterrolebinding cluster-admin-binding --clusterrole cluster-admin --user $(gcloud config get-value account)

=====================================================================

======**Asia: rename default GKE cluster context to more representative names**=======

kubectl config set-context asia-context --cluster gke\_vishal-209706\_asia-southeast1-a\_asia --user gke\_**vishal-209706**\_asia-southeast1-a\_asia

kubectl config delete-context gke\_**vishal-209706**\_asia-southeast1-a\_asia

=====================================================================

**Don’t forget to replace vishal-209706\_us with your own GCP project id.**

====**America : rename default GKE cluster context to more representative names**======

kubectl config set-context america-context --cluster gke\_vishal-209706\_us-central1-a\_america --user gke\_**vishal-209706**\_us-central1-a\_america

kubectl config delete-context gke\_**vishal-209706**\_us-central1-a\_america

=====================================================================

====================== **To view and delete context** =========================

kubectl config get-contexts

kubectl config delete-context global-context

=====================================================================

=======**We now have everything in place to create a federated cluster**==============

kubefed init global-context --host-cluster-context=america-context --dns-zone-name="devops.vishalvyas.com." --dns-provider="google-clouddns"

...

…….

This step is the most crucial since it creates the federated control plane. After a few minutes, you should see the below output.

Creating a namespace federation-system for federation system components... done

Creating federation control plane service.............. done

Creating federation control plane objects (credentials, persistent volume claim)... done

Creating federation component deployments... done

Updating kubeconfig... done

Waiting for federation control plane to come up..................... done

Federation API server is running at: 35.202.187.107

=====================================================================

===========**Join all the clusters to the federated control plane**===============

kubefed --context=global-context join asia --cluster-context=asia-context --host-cluster-context=**america-context**

kubefed --context=global-context join america --cluster-context=america-context --host-cluster-context=**america-context**

Check Cluster :

kubectl --context=global-context get clusters

=====================================================================

=============== **Create Default name space** ===============================

kubectl --context=global-context create ns default

=====================================================================

==================**Time to download KUBEMCI**=============================

https://storage.googleapis.com/kubemci-release/release/latest/bin/linux/amd64/kubemci

chmod +x ./kubemci

sudo mv kubemci /usr/local/bin

=====================================================================

==================**Pass each cluster configuration to kubemci.**=================

KUBECONFIG=$HOME/mcikubeconfig gcloud container clusters get-credentials asia --zone asia-southeast1-a --project vishal-209706

KUBECONFIG=$HOME/mcikubeconfig gcloud container clusters get-credentials america --zone us-central1-a --project vishal-209706

=====================================================================

==================**Create a static IP address for our global ingress**==============

$ gcloud compute addresses create g-ingress --global

=====================================================================

===================**To check global-context clusters**=========================

kubectl --context=global-context get clusters

=====================================================================

====**Deploy Zone Printer to prints the details of the zone where it is being served** =====

Deploy :

kubectl --context=global-context create deployment zoneprinter --image=gcr.io/google-samples/zone-printer:0.1

Create Service :

kubectl --context=global-context create service nodeport zoneprinter --tcp=80:80 --node-port=30061

Scale Container :

kubectl --context=global-context scale deployment zoneprinter --replicas=4

===========**Create and run Ingress file to deploy zoneprinter**====================

**Vim ingress.yaml**

==========================

zoneprinteringress.yaml

apiVersion: extensions/v1beta1

kind: Ingress

metadata:

name: zoneprinter

annotations:

kubernetes.io/ingress.global-static-ip-name: "k-ingress"

spec:

backend:

serviceName: zoneprinter

servicePort: 80

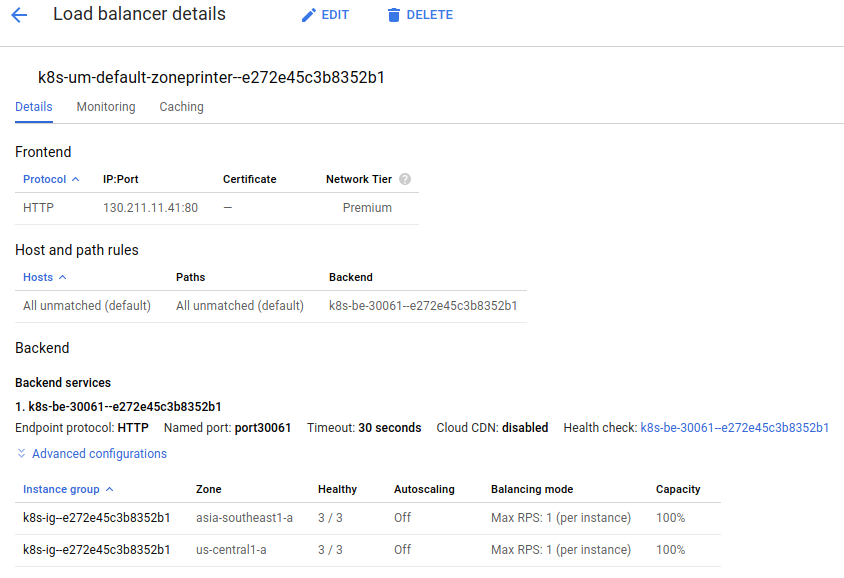
=================================

Run Ingress in global context :

kubectl --context=global-context create -f ingress.yaml

This single step results in a series of configuration tasks involving the creation of load balancers, firewall rules, health checks, and proxies.

Keep checking the GCP Console for the health status. Once all the clusters are registered with the ingress, it should show all the clusters as healthy. Wait till you see 3/3 in the Healthy column of Backend services section. It typically takes a few minutes for the health check to pass.



It’s time for the final test. To verify the geo-routing feature, I am using <https://www.geoscreenshot.com> to check from different locations.

