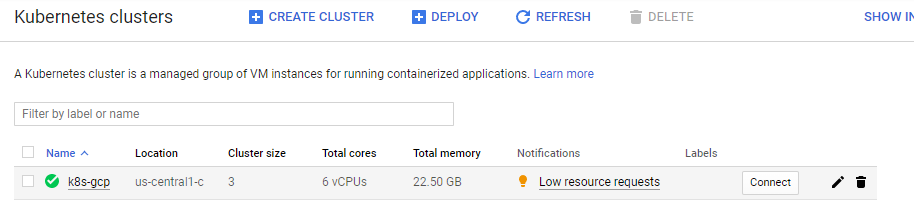
CANARY Deployment in GKE using ISTIO

**DevOps Tools included**

1. GCP Kubernetes
2. Docker
3. Istio

**Steps:**

1. Create GCP Kubernetes Engine with 3 nodes (default) with 2 Core CPUs each



1. Login into Google Cloud Shell by clicking **Connect.**
2. Install Istio 1.0.3 version
   1. Create Istio namespace

***kubectl create namespace istio-system***

* 1. Istio Installation

***cd ~***

***wget https://github.com/istio/istio/releases/download/1.0.3/istio-1.0.3-linux.tar.gz***

***tar -xzvf istio-1.0.3-linux.tar.gz***

***cd istio-1.0.3***

***echo 'export PATH="$PATH:/home/siva1220/istio-1.0.3/bin"' >> ~/.profile***

* 1. Apply CRDs

***kubectl apply -f ~/istio-1.0.3/install/kubernetes/helm/istio/templates/crds.yaml***

* 1. with no mutual TLS authentication

***kubectl apply -f ~/istio-1.0.3/install/kubernetes/istio-demo.yaml***

* 1. Check applied CRDs

***kubectl get crds***

1. **Canary deployment**
   1. **With Version 1 (v1)**
      1. Relevant YAML files are below



* + 1. Commands to apply
       1. ***kubectl apply -f <(/home/siva1220/istio-1.0.3/bin/istioctl kube-inject -f helloworld.yaml)***
       2. ***kubectl apply -f helloworld-gw.yaml***
    2. Command to get Ingress Gateway External IP

***kubectl get svc istio-ingressgateway -n istio-system | awk 'END {print $4}'***

* + 1. To verify the deployment using External IP with curl command

***curl 104.198.45.25:80/hello***

* 1. **With updated Version 2 (v2)**
     1. Relevant YAML Files are below



* + 1. Commands to apply
       1. ***kubectl apply -f <(/home/siva1220/istio-1.0.3/bin/istioctl kube-inject -f helloworld-v2.yml)***
       2. ***kubectl apply -f helloworld-v2-routing.yml***
    2. To verify the deployment using External IP with curl command.
       1. Normal user mode

***curl 104.198.45.25:80 -H "host: hello.example.com"***

* + - 1. Canary user mode

***curl 104.198.45.25:80 -H "host: hello.example.com" -H "end-user: siva"***

***Note :***

I have implemented using ‘end user’ category, we shall implement in ‘Weight’ category as well

**Spinnaker deployment in Kubernetes**

***PROJECT\_ID=${DEVSHELL\_PROJECT\_ID} ~/cloudshell\_open/spinnaker-for-gcp/scripts/install/setup\_properties.sh***

***~/cloudshell\_open/spinnaker-for-gcp/scripts/install/setup.sh***

**Spinnaker – Creation and deploying pipeline**

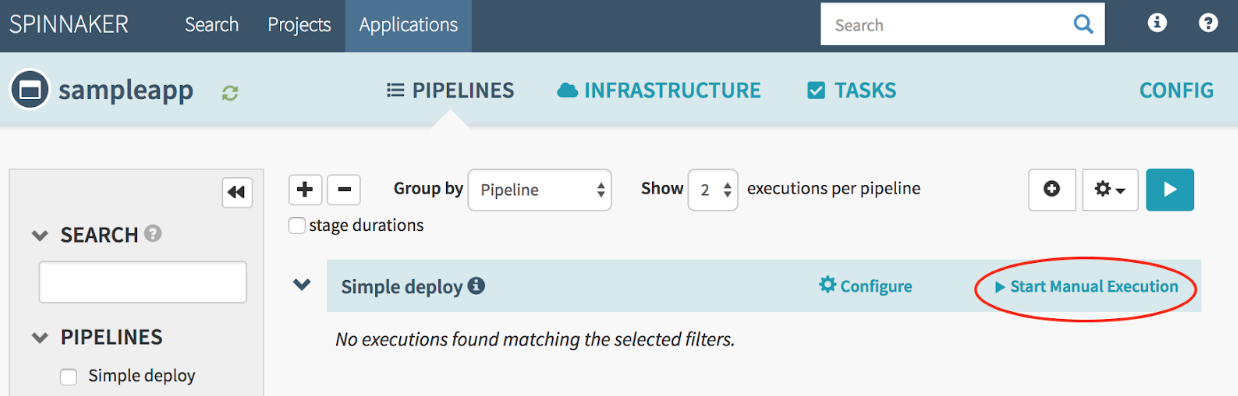
**create the pipeline with the provided JSON file**

***cd ~***

***wget https://raw.githubusercontent.com/spinnaker/spinnaker/master/solutions/kayenta/pipelines/simple-deploy.json***

***sed "s/my-kubernetes-account/spinnaker-install-account/g" simple-deploy.json > updated-simple-deploy.json***

***spin pipeline save --file updated-simple-deploy.json***



**Creation of Pod:**

***kubectl -n default run injector --generator=run-pod/v1 --image=alpine:3.10 -- \***

***/bin/sh -c "apk add --no-cache curl; \***

***while true; do curl -sS --max-time 3 \***

***http://sampleapp:8080/; done"***

**Create and Run Canary deployment using Spinnaker**

***cd ~***

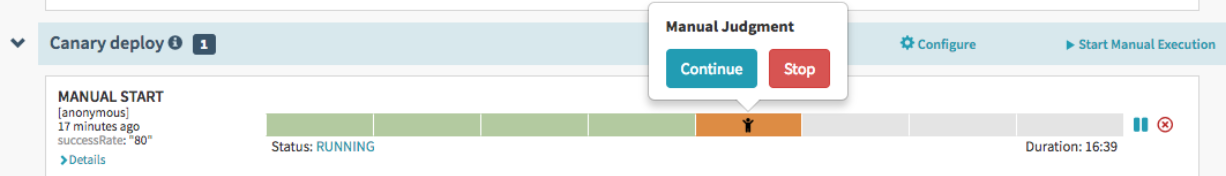
***wget https://raw.githubusercontent.com/spinnaker/spinnaker/master/solutions/kayenta/pipelines/canary-deploy.json***

***export PIPELINE\_ID=$(spin pipeline get -a sampleapp -n 'Simple deploy' | jq -r '.id')***

***jq '(.stages[] | select(.refId == "9") | .pipeline) |= env.PIPELINE\_ID | (.stages[] | select(.refId == "8") | .pipeline) |= env.PIPELINE\_ID' canary-deploy.json | \***

***sed "s/my-kubernetes-account/spinnaker-install-account/g" > updated-canary-deploy.json***

***spin pipeline save --file updated-canary-deploy.json***



Manual Judgement stage

***kubectl -n default get pods*** - command to see the new pods labelled canary and baseline:

Based on the canary deployment and the baseline analysis, we can proceed with the increment of the canary deployment request percentage. Once you are 100 percent fine with the canary deployment. Route the traffic to the new deployment and delete the baseline.

Thanks

Siva

+49 15145798782