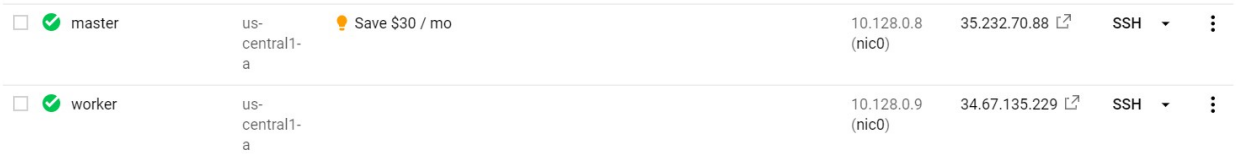
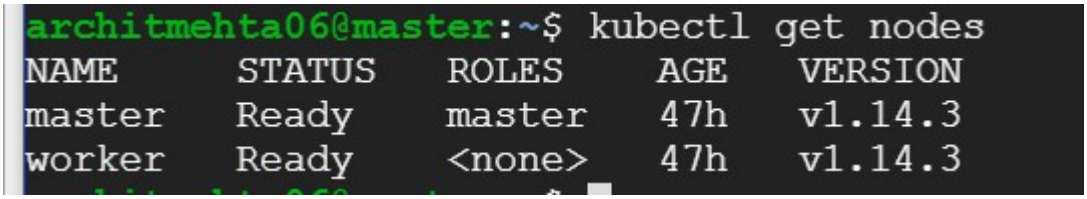
Tasks:

1. Create a Highly available Kubernetes cluster manually using Google Compute Engines (GCE). Do not create a Kubernetes hosted solution using Google Kubernetes Engine (GKE). Use Kubeadm(preferred)/kubespray. **Do not use kops**.
   ➔ Used kubeadm for installation and flannel as overlay network. Created 2 node cluster 1-master, 1-slave
   ➔ Created 2 VM on GCP



Created kubernetes cluster with 2 VMs



architmehta06@master:~$ kubectl cluster-info
Kubernetes master is running at https://10.128.0.8:6443
KubeDNS is running at https://10.128.0.8:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
architmehta06@master:~$ kubectl config current-context
kubernetes-admin@kubernetes

2. Create a CI/CD pipeline using Jenkins (or a CI tool of your choice) outside Kubernetes cluster (not as a pod inside Kubernetes cluster).
   → Skipped

3. Create a development namespace.
   ➔ kubectl create namespace development

4. Deploy guest-book application (or any other application which you think is more suitable to showcase your ability, kindly justify why you have chosen a different application) in the development namespace.
   ➔ Skipped

5. Install and configure Helm in Kubernetes
   ➔ Steps to install helm
   architmehta06@master:~$ curl https://raw.githubusercontent.com/kubernetes/helm/master/scripts/get | bash

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  7001  100  7001    0     0  83718      0 --:--:-- --:--:-- --:--:-- 83345
Downloading https://get.helm.sh/helm-v2.14.1-linux-amd64.tar.gz
Preparing to install helm and tiller into /usr/local/bin
helm installed into /usr/local/bin/helm
tiller installed into /usr/local/bin/tiller
Run 'helm init' to configure helm.
architmehta06@master:~$ kubectl --namespace kube-system create sa tiller
serviceaccount/tiller created
architmehta06@master:~$ kubectl create clusterrolebinding tiller --clusterrole cluster-admin
--serviceaccount=kube-system:tiller clusterrolebinding.rbac.authorization.k8s.io/tiller created
architmehta06@master:~$ helm init --service-account tiller
Creating /home/architmehta06/.helm
Creating /home/architmehta06/.helm/repository
Creating /home/architmehta06/.helm/repository/cache
Creating /home/architmehta06/.helm/repository/local
Creating /home/architmehta06/.helm/plugins
Creating /home/architmehta06/.helm/starters
Creating /home/architmehta06/.helm/cache/archive
Creating /home/architmehta06/.helm/repository/repositories.yaml
Adding stable repo with URL: https://kubernetes-charts.storage.googleapis.com
Adding local repo with URL: http://127.0.0.1:8879/charts
$HELM_HOME has been configured at /home/architmehta06/.helm.

Tiller (the Helm server-side component) has been installed into your Kubernetes Cluster.

Please note: by default, Tiller is deployed with an insecure 'allow unauthenticated users'
policy.
To prevent this, run `helm init` with the --tiller-tls-verify flag.
For more information on securing your installation see:
https://docs.helm.sh/using_helm/#securing-your-helm-installation
architmehta06@master:~$ helm repo update
Hang tight while we grab the latest from your chart repositories...
...Skip local chart repository
...Successfully got an update from the "stable" chart repository
Update Complete.
architmehta06@master:~$ kubectl get deploy,svc tiller-deploy -n kube-system
NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.extensions/tiller-deploy   1/1     1            1           37s

NAME                  TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)     AGE
service/tiller-deploy   ClusterIP   10.102.177.111   <none>        44134/TCP   37s
```

➔ Installed helm

```
architmehta06@master:~$ helm version
Client: &version.Version{SemVer:"v2.14.1", GitCommit:"5270352a09c7e8b6e8c9593002a73535276507c0", GitTreeState:"clean"}
Server: &version.Version{SemVer:"v2.14.1", GitCommit:"5270352a09c7e8b6e8c9593002a73535276507c0", GitTreeState:"clean"}
```

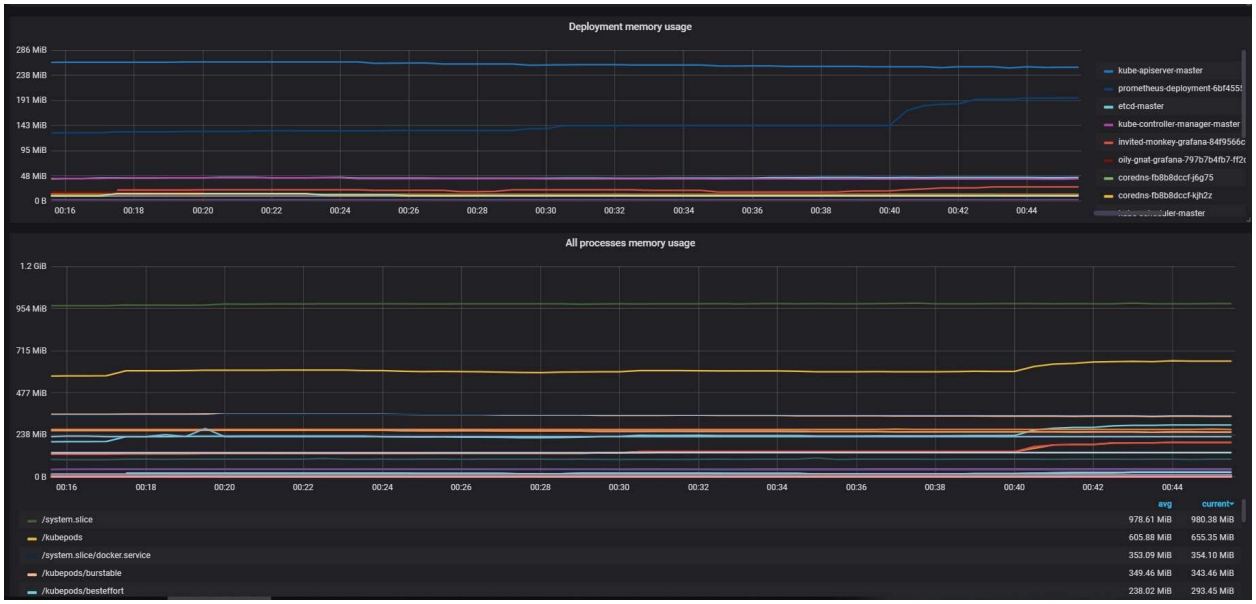
6. Use Helm to deploy the application on Kubernetes Cluster from CI server.
   ➔ Skipped

7. Create a monitoring namespace in the cluster.
   → kubectl create namespace monitoring

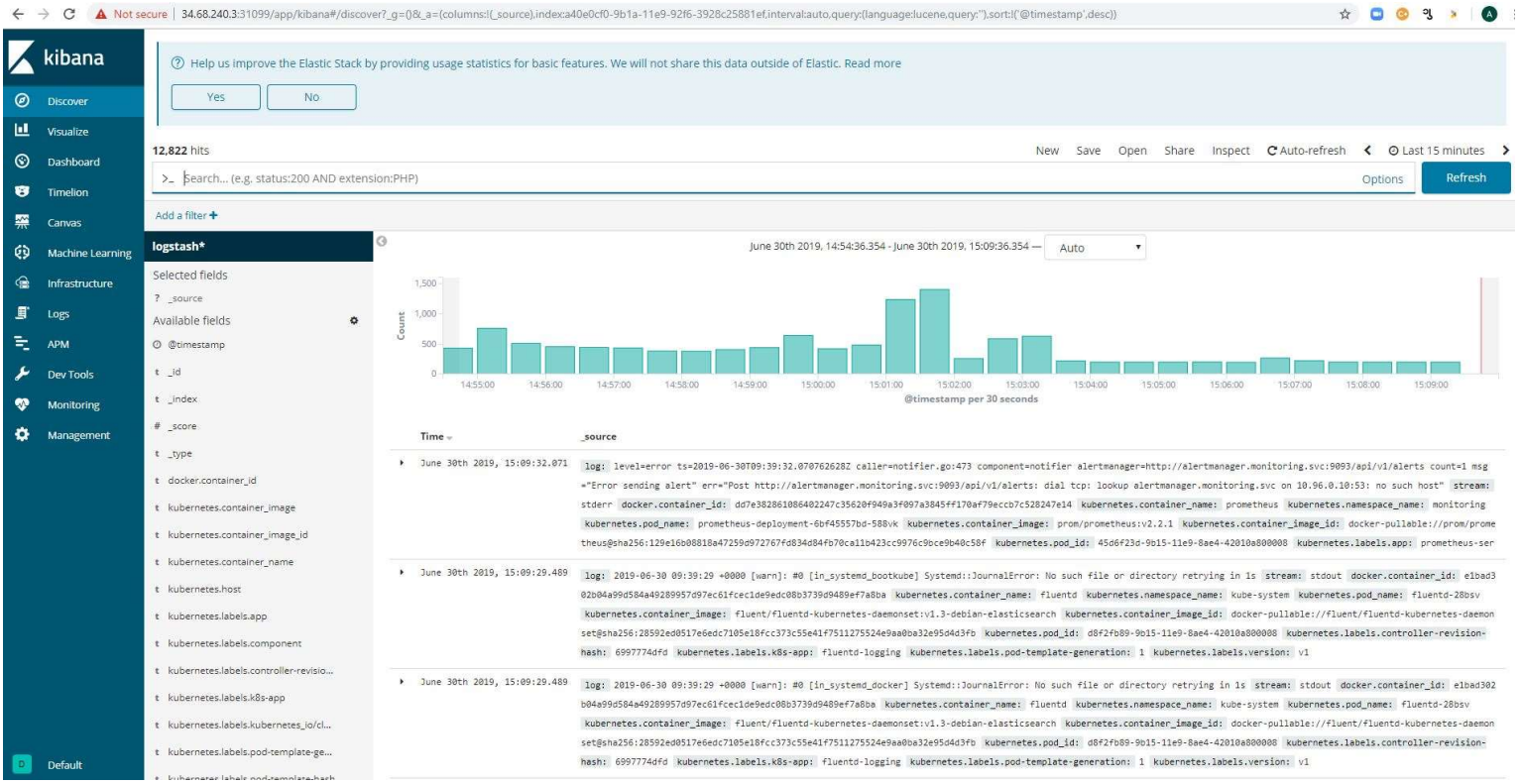
8. Setup Prometheus (in monitoring namespace) for gathering host/container metrics along with health check status of the application.
   → http://35.232.70.88:30000

9. Create a dashboard using Grafana to help visualize the Node/Container/API Server etc. metrices from Prometheus server. Optionally create a custom dashboard on Grafana
   →http://35.232.70.88:30233/

   Deployment Matrics : http://35.232.70.88:30233/d/XOE4JCfmz/kubernetes-deployment-metrics?orgId=1



10. Setup log analysis using Elasticsearch, Fluentd (or Filebeat), Kibana.



11. Demonstrate Blue/Green and Canary deployment for the application (For e.g. Change the background color or font in the new version etc.,)
→ Skipped

12. Write a wrapper script (or automation mechanism of your choice) which does all the steps above.
    → Refer: automation2.py

13. Document the whole process in a README file at the root of your repo. Mention any pre-requisites in the README.