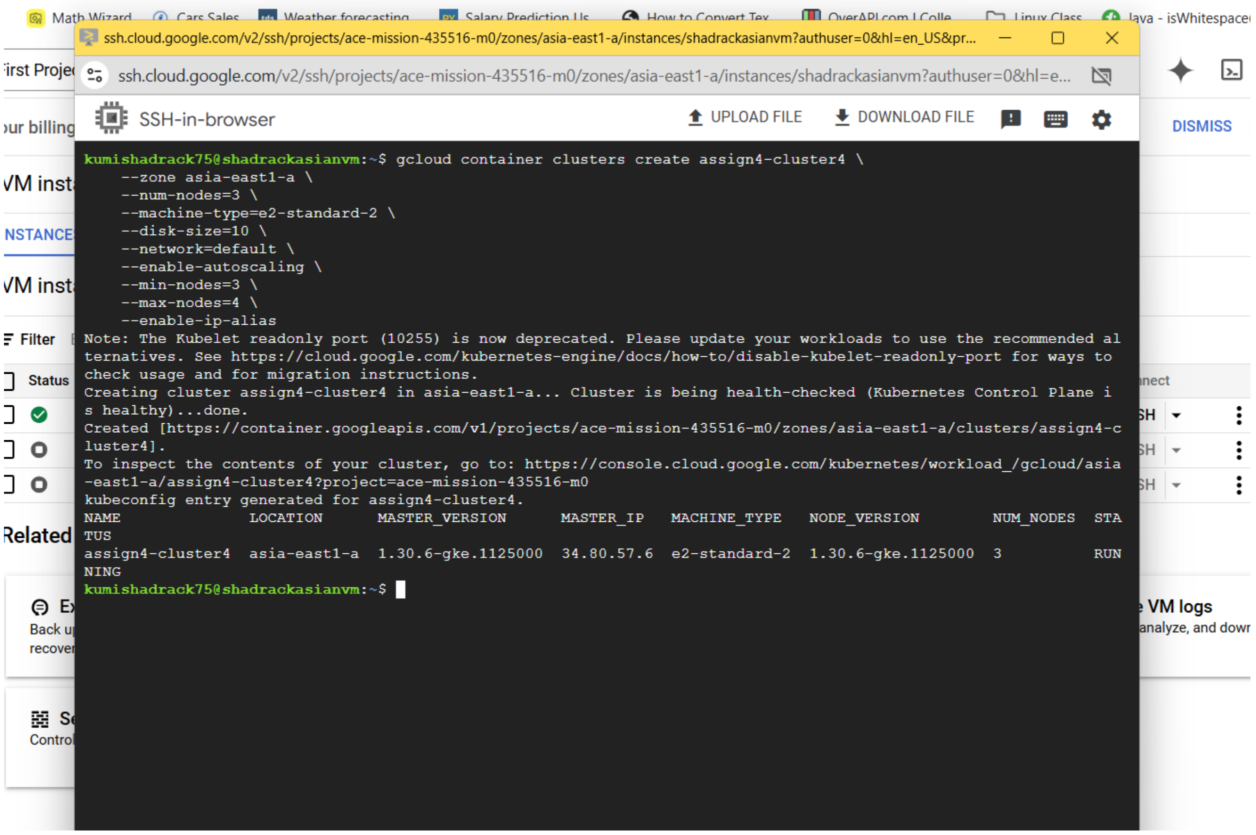
1. GKE Cluster Configuration:

Create GKE Cluster (using gcloud):

```
gcloud container clusters create assign4-cluster4 \
  --zone asia-east1-a \
  --num-nodes=3 \
  --machine-type=e2-standard-2 \
  --disk-size=10 \
  --network=default \
  --enable-autoscaling \
  --min-nodes=3 \
  --max-nodes=4 \
  --enable-ip-alias
```

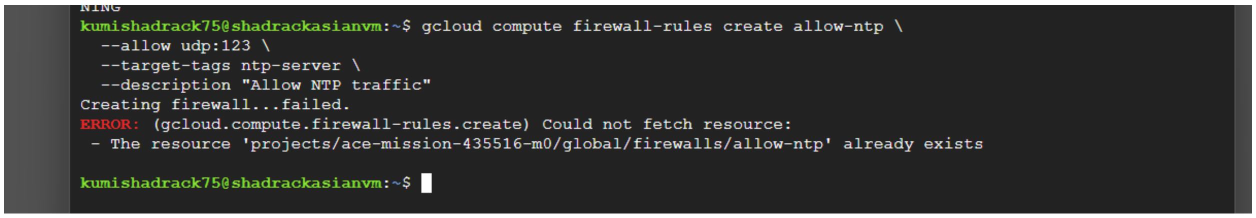


ssh.cloud.google.com/v2/ssh/projects/ace-mission-435516-m0/zones/asia-east1-a/instances/shadrackasianvm?authuser=0&hl=e...
SSH-in-browser
UPLOAD FILE
DOWNLOAD FILE
DISMISS

```
kumishadrack75@shadrackasianvm:~$ gcloud container clusters create assign4-cluster4 \
    --zone asia-east1-a \
    --num-nodes=3 \
    --machine-type=e2-standard-2 \
    --disk-size=10 \
    --network=default \
    --enable-autoscaling \
    --min-nodes=3 \
    --max-nodes=4 \
    --enable-ip-alias
Note: The Kubelet readonly port (10255) is now deprecated. Please update your workloads to use the recommended al
ternatives. See https://cloud.google.com/kubernetes-engine/docs/how-to/disable-kubelet-readonly-port for ways to
check usage and for migration instructions.
Creating cluster assign4-cluster4 in asia-east1-a... Cluster is being health-checked (Kubernetes Control Plane i
s healthy)...done.
Created [https://container.googleapis.com/v1/projects/ace-mission-435516-m0/zones/asia-east1-a/clusters/assign4-c
luster4].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload_/gcloud/asia
-east1-a/assign4-cluster4?project=ace-mission-435516-m0
kubeconfig entry generated for assign4-cluster4.
NAME              LOCATION      MASTER_VERSION       MASTER_IP    MACHINE_TYPE   NODE_VERSION         NUM_NODES  STA
TUS
assign4-cluster4  asia-east1-a  1.30.6-gke.1125000  34.80.57.6   e2-standard-2  1.30.6-gke.1125000  3          RUN
NING
kumishadrack75@shadrackasianvm:~$
```

Firewall Rules to Allow NTP Traffic:

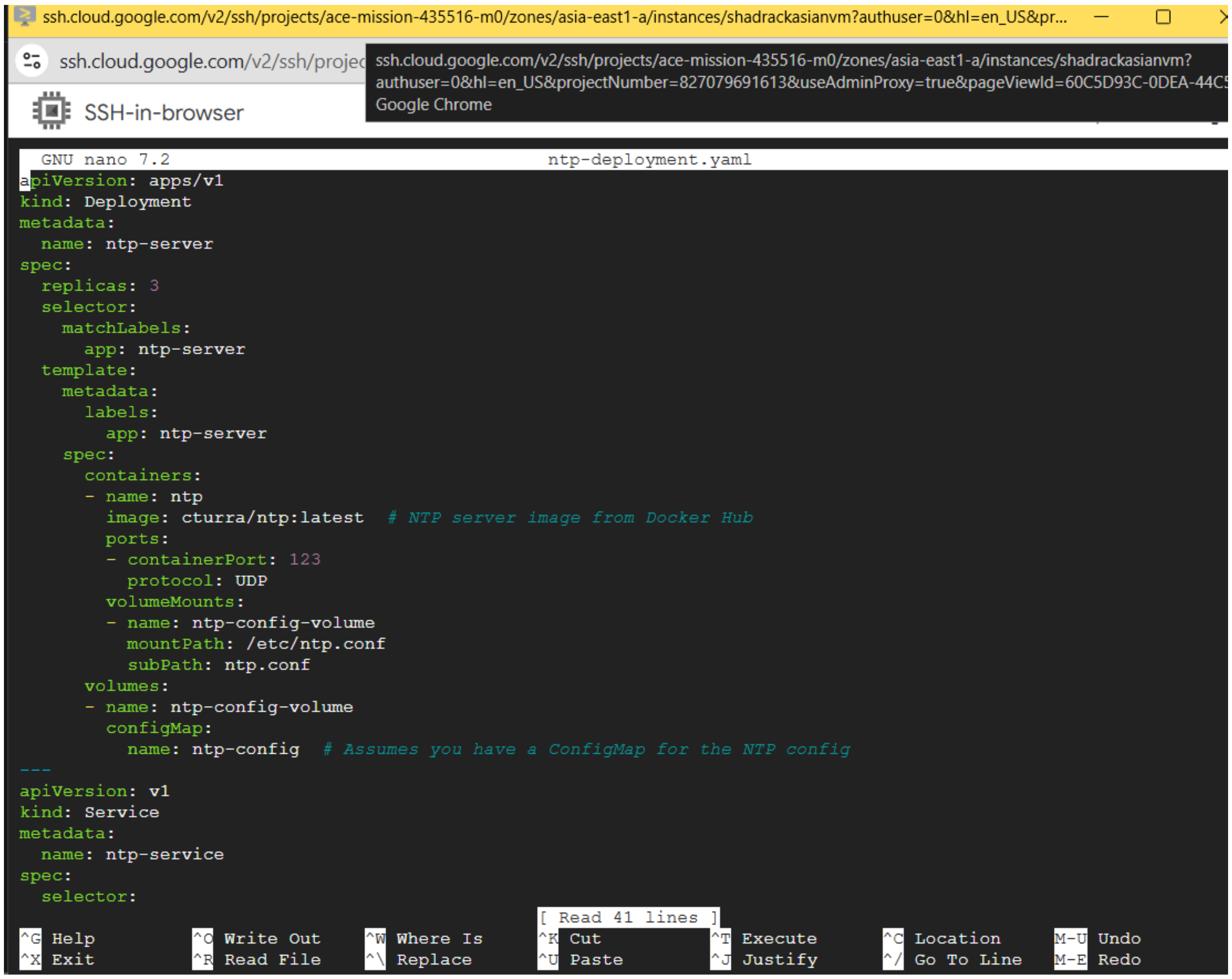
```
gcloud compute firewall-rules create allow-ntp \
  --allow udp:123 \
  --target-tags ntp-server \
  --description "Allow NTP traffic"
```



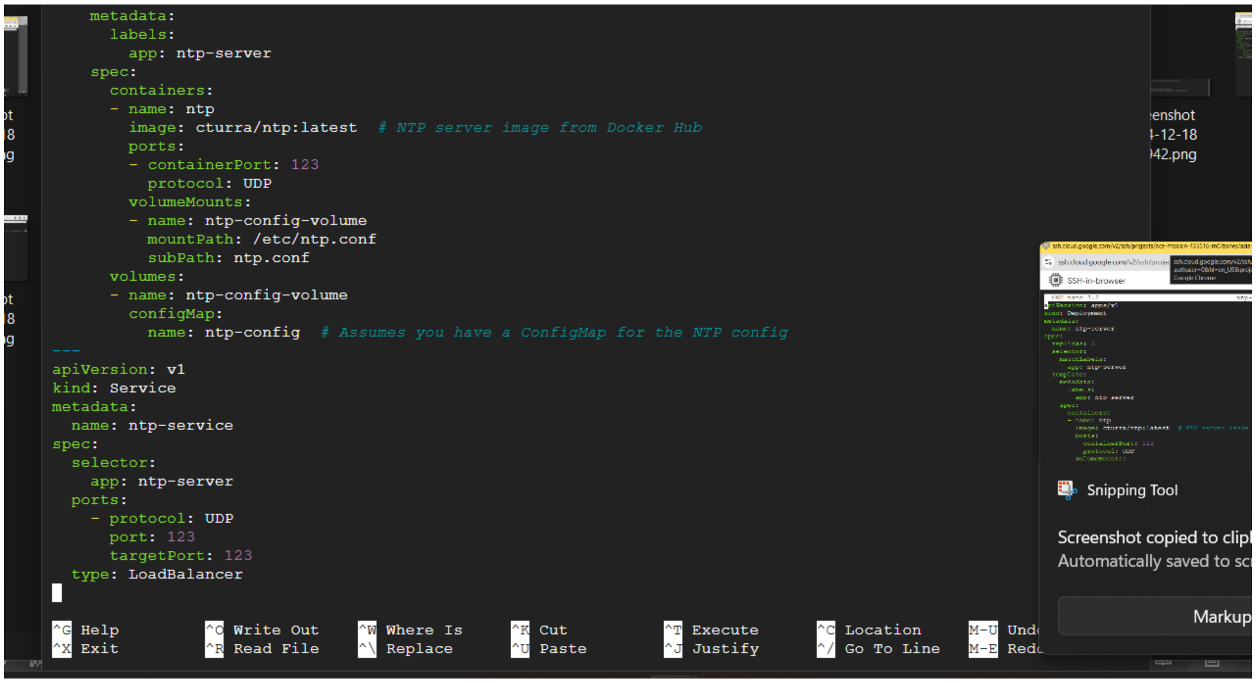
```
NING
kumishadrack75@shadrackasianvm:~$ gcloud compute firewall-rules create allow-ntp \
  --allow udp:123 \
  --target-tags ntp-server \
  --description "Allow NTP traffic"
Creating firewall...failed.
ERROR: (gcloud.compute.firewall-rules.create) Could not fetch resource:
 - The resource 'projects/ace-mission-435516-m0/global/firewalls/allow-ntp' already exists

kumishadrack75@shadrackasianvm:~$ 
```

2. NTP Server Deployment:

Deployment YAML (ntp-deployment.yaml):

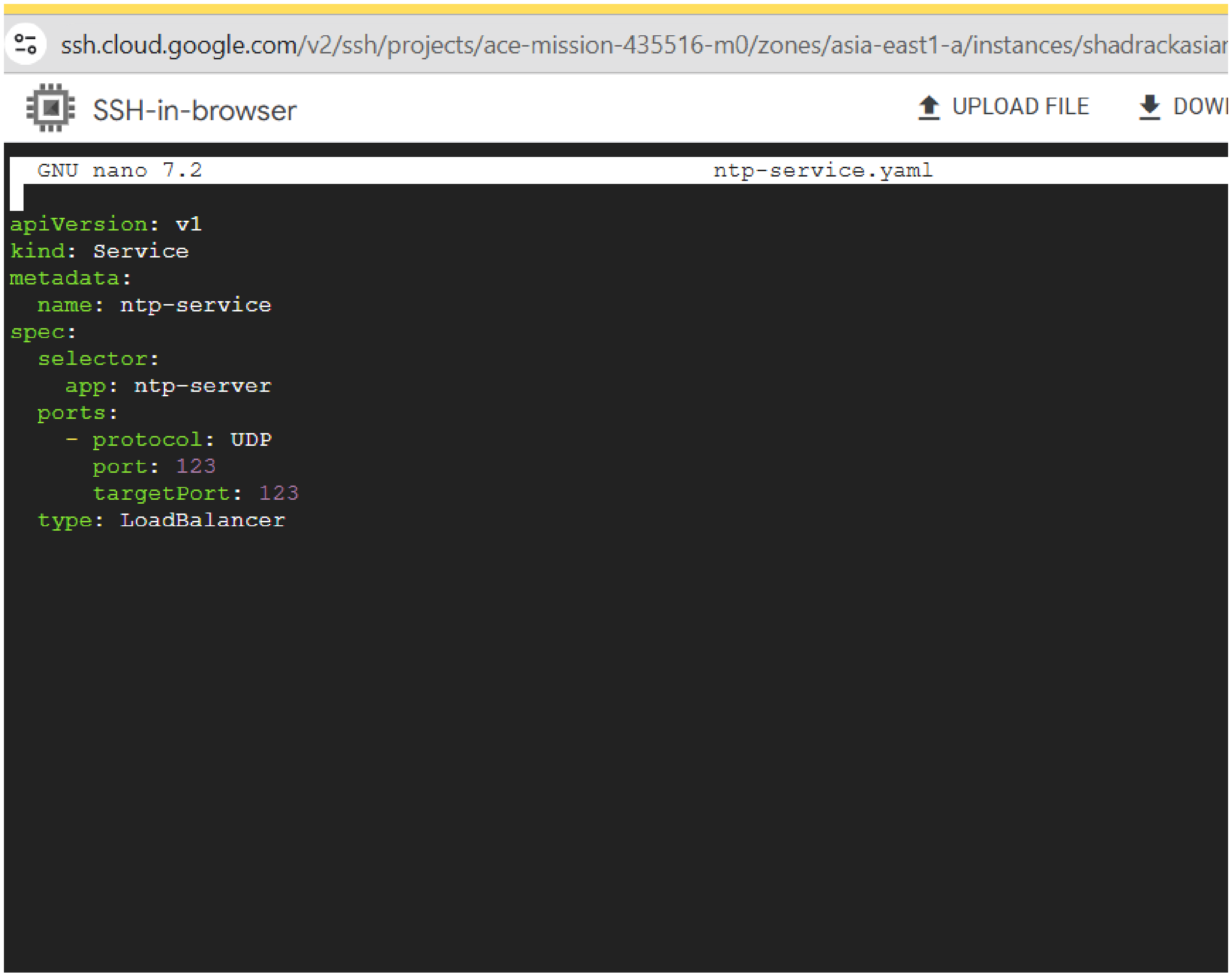

ssh.cloud.google.com/v2/ssh/projects/ace-mission-435516-m0/zones/asia-east1-a/instances/shadrackasianvm?authuser=0&hl=en_US&pr...
ssh.cloud.google.com/v2/ssh/projec
ssh.cloud.google.com/v2/ssh/projects/ace-mission-435516-m0/zones/asia-east1-a/instances/shadrackasianvm?
authuser=0&hl=en_US&projectNumber=827079691613&useAdminProxy=true&pageViewId=60C5D93C-0DEA-44C5
Google Chrome
SSH-in-browser

```
  GNU nano 7.2                                      ntp-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ntp-server
spec:
  replicas: 3
  selector:
    matchLabels:
      app: ntp-server
  template:
    metadata:
      labels:
        app: ntp-server
    spec:
      containers:
      - name: ntp
        image: cturra/ntp:latest  # NTP server image from Docker Hub
        ports:
        - containerPort: 123
          protocol: UDP
        volumeMounts:
        - name: ntp-config-volume
          mountPath: /etc/ntp.conf
          subPath: ntp.conf
      volumes:
      - name: ntp-config-volume
        configMap:
          name: ntp-config  # Assumes you have a ConfigMap for the NTP config
---
apiVersion: v1
kind: Service
metadata:
  name: ntp-service
spec:
  selector:
                                  [ Read 41 lines ]
^G Help         ^O Write Out    ^W Where Is     ^K Cut          ^T Execute      ^C Location     M-U Undo
^X Exit         ^R Read File    ^\ Replace      ^U Paste        ^J Justify      ^/ Go To Line   M-E Redo
```



```
    metadata:
      labels:
        app: ntp-server
    spec:
      containers:
      - name: ntp
        image: cturra/ntp:latest  # NTP server image from Docker Hub
        ports:
        - containerPort: 123
          protocol: UDP
        volumeMounts:
        - name: ntp-config-volume
          mountPath: /etc/ntp.conf
          subPath: ntp.conf
      volumes:
      - name: ntp-config-volume
        configMap:
          name: ntp-config  # Assumes you have a ConfigMap for the NTP config
---
apiVersion: v1
kind: Service
metadata:
  name: ntp-service
spec:
  selector:
    app: ntp-server
  ports:
    - protocol: UDP
      port: 123
      targetPort: 123
  type: LoadBalancer

^G Help         ^O Write Out    ^W Where Is     ^K Cut          ^T Execute      ^C Location     M-U Und
^X Exit         ^R Read File    ^\ Replace      ^U Paste        ^J Justify      ^/ Go To Line   M-E Red
```

Service YAML (ntp-service.yaml):


ssh.cloud.google.com/v2/ssh/projects/ace-mission-435516-m0/zones/asia-east1-a/instances/shadrackasiar
SSH-in-browser
UPLOAD FILE
DOWI

```
GNU nano 7.2                                          ntp-service.yaml

apiVersion: v1
kind: Service
metadata:
  name: ntp-service
spec:
  selector:
    app: ntp-server
  ports:
    - protocol: UDP
      port: 123
      targetPort: 123
  type: LoadBalancer
```

3. Configure the NTP Server:

NTP Configurations (Optional)

- **Create a ConfigMap** to customize the NTP server configuration (e.g., time sources):

kubectl create configmap ntp-config --from-file=ntp.conf=/path/to/ntp.conf

Modify the Deployment YAML to Mount the ConfigMap:

apiVersion: apps/v1

kind: Deployment

```
metadata:
 name: ntp-server
spec:
 replicas: 3
 selector:
  matchLabels:
   app: ntp-server
 template:
  metadata:
   labels:
    app: ntp-server
  spec:
   containers:
   - name: ntp
    image: ntpd:latest
    volumeMounts:
    - name: ntp-config-volume
     mountPath: /etc/ntp.conf
     subPath: ntp.conf
   volumes:
   - name: ntp-config-volume
    configMap:
     name: ntp-config
```

4. Automatic Update Configuration:

Rolling Update Strategy (modify deployment YAML):

```
spec:
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
      maxSurge: 1
```

CI/CD Pipeline (using GitHub Actions as an example):

- Create a .github/workflows/ci-cd-pipeline.yaml file in your repository:

5. Monitoring Configuration:

Enable Monitoring (GKE Built-in Monitoring):

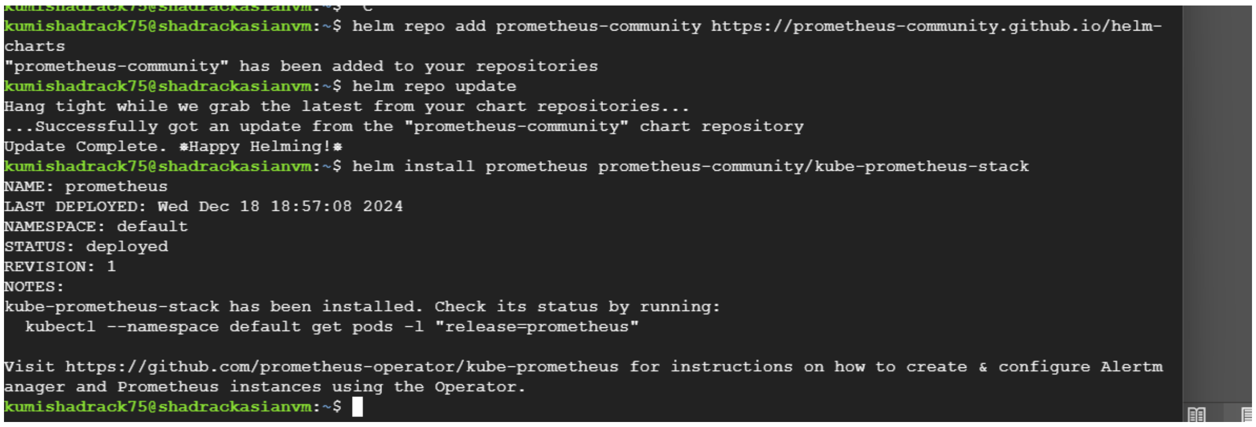
```
gcloud container clusters update ntp-cluster \
  --zone asia-east1-a \
  --enable-stackdriver-kubernetes
```

Create Monitoring Alerts in Google Cloud Console:

- Navigate to **Monitoring** > **Alerting** > **Create Policy** in the Google Cloud Console.
- Set up alerts for CPU, memory, and network traffic.

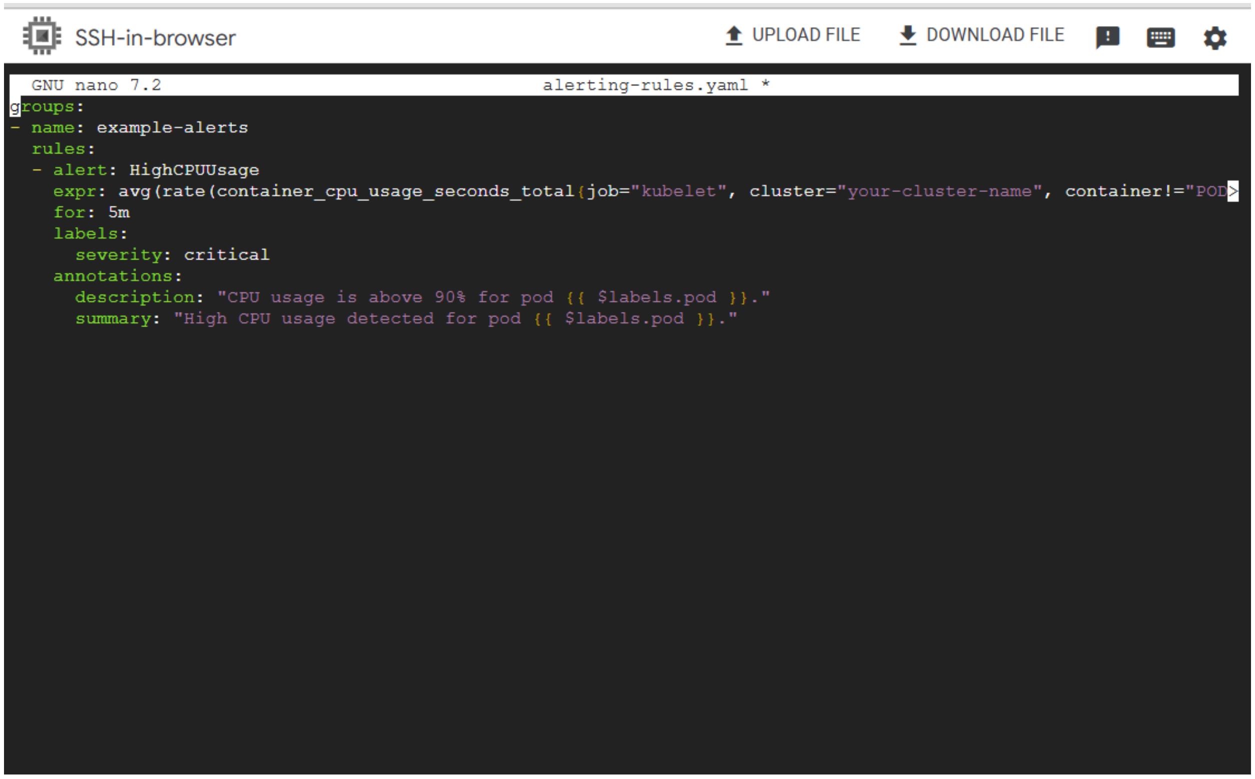
Prometheus + Grafana (Optional, for enhanced monitoring):

1. **Install Prometheus and Grafana using Helm:**



```
kumishadrack75@shadrackasianvm:~$ ^C
kumishadrack75@shadrackasianvm:~$ helm repo add prometheus-community https://prometheus-community.github.io/helm-
charts
"prometheus-community" has been added to your repositories
kumishadrack75@shadrackasianvm:~$ helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "prometheus-community" chart repository
Update Complete. ⎈Happy Helming!⎈
kumishadrack75@shadrackasianvm:~$ helm install prometheus prometheus-community/kube-prometheus-stack
NAME: prometheus
LAST DEPLOYED: Wed Dec 18 18:57:08 2024
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
kube-prometheus-stack has been installed. Check its status by running:
  kubectl --namespace default get pods -l "release=prometheus"

Visit https://github.com/prometheus-operator/kube-prometheus for instructions on how to create & configure Alertm
anager and Prometheus instances using the Operator.
kumishadrack75@shadrackasianvm:~$
```

Set up alerting rules in Prometheus: Example alert rule:



```
  GNU nano 7.2                                   alerting-rules.yaml *
groups:
- name: example-alerts
  rules:
  - alert: HighCPUUsage
    expr: avg(rate(container_cpu_usage_seconds_total{job="kubelet", cluster="your-cluster-name", container!="POD>
    for: 5m
    labels:
      severity: critical
    annotations:
      description: "CPU usage is above 90% for pod {{ $labels.pod }}."
      summary: "High CPU usage detected for pod {{ $labels.pod }}."
```

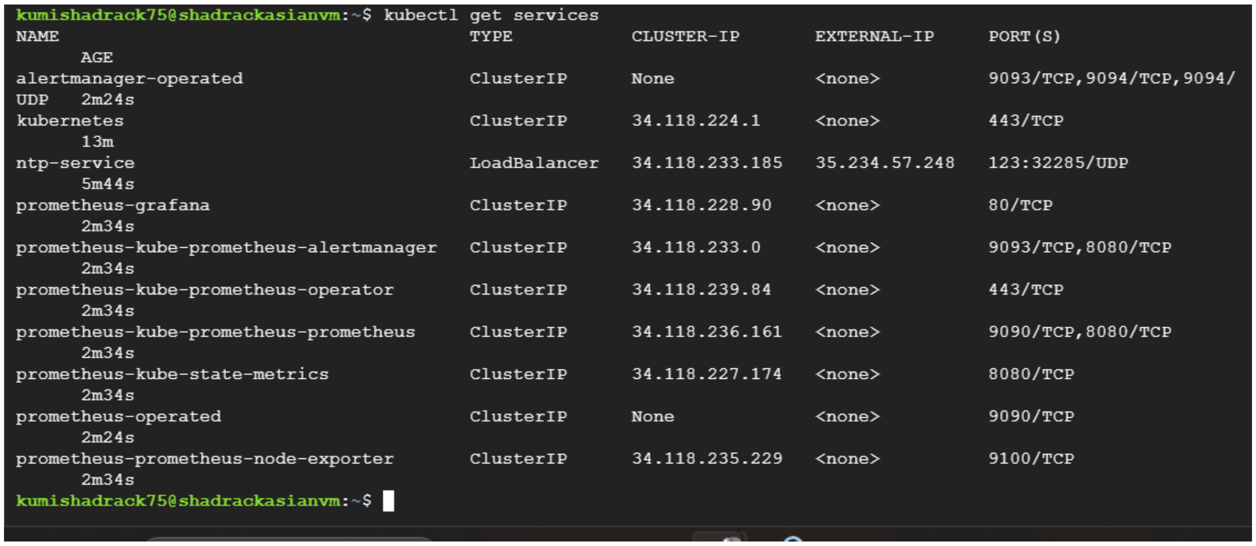
6. Test the NTP Server Cluster:

Test NTP with ntpdate or chronyc:

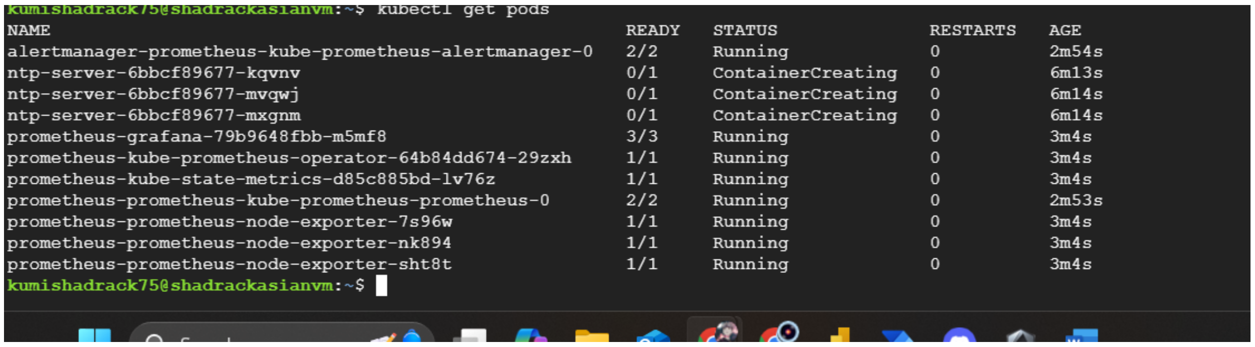
- From a client machine:
- ntpdate <load_balancer_ip>

OR using chronyc:

chronyc tracking

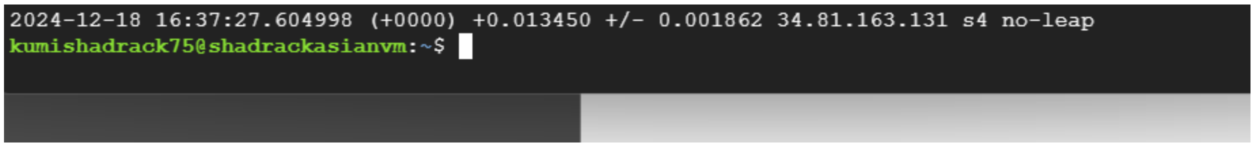


```
kumishadrack75@shadrackasianvm:~$ kubectl get services
NAME                                        TYPE           CLUSTER-IP       EXTERNAL-IP     PORT(S)
      AGE
alertmanager-operated                       ClusterIP      None             <none>          9093/TCP,9094/TCP,9094/
UDP   2m24s
kubernetes                                  ClusterIP      34.118.224.1     <none>          443/TCP
      13m
ntp-service                                 LoadBalancer   34.118.233.185   35.234.57.248   123:32285/UDP
      5m44s
prometheus-grafana                          ClusterIP      34.118.228.90    <none>          80/TCP
      2m34s
prometheus-kube-prometheus-alertmanager     ClusterIP      34.118.233.0     <none>          9093/TCP,8080/TCP
      2m34s
prometheus-kube-prometheus-operator         ClusterIP      34.118.239.84    <none>          443/TCP
      2m34s
prometheus-kube-prometheus-prometheus       ClusterIP      34.118.236.161   <none>          9090/TCP,8080/TCP
      2m34s
prometheus-kube-state-metrics               ClusterIP      34.118.227.174   <none>          8080/TCP
      2m34s
prometheus-operated                         ClusterIP      None             <none>          9090/TCP
      2m24s
prometheus-prometheus-node-exporter         ClusterIP      34.118.235.229   <none>          9100/TCP
      2m34s
kumishadrack75@shadrackasianvm:~$
```



```
kumishadrack75@shadrackasianvm:~$ kubectl get pods
NAME                                                     READY   STATUS              RESTARTS   AGE
alertmanager-prometheus-kube-prometheus-alertmanager-0   2/2     Running             0          2m54s
ntp-server-6bbcf89677-kqvnv                              0/1     ContainerCreating   0          6m13s
ntp-server-6bbcf89677-mvqwj                              0/1     ContainerCreating   0          6m14s
ntp-server-6bbcf89677-mxgnm                              0/1     ContainerCreating   0          6m14s
prometheus-grafana-79b9648fbb-m5mf8                      3/3     Running             0          3m4s
prometheus-kube-prometheus-operator-64b84dd674-29zxh     1/1     Running             0          3m4s
prometheus-kube-state-metrics-d85c885bd-lv76z            1/1     Running             0          3m4s
prometheus-prometheus-kube-prometheus-prometheus-0       2/2     Running             0          2m53s
prometheus-prometheus-node-exporter-7s96w                1/1     Running             0          3m4s
prometheus-prometheus-node-exporter-nk894                1/1     Running             0          3m4s
prometheus-prometheus-node-exporter-sht8t                1/1     Running             0          3m4s
kumishadrack75@shadrackasianvm:~$
```

Verify Time Sync: After running the command, verify that the client gets an accurate time from the NTP server cluster.

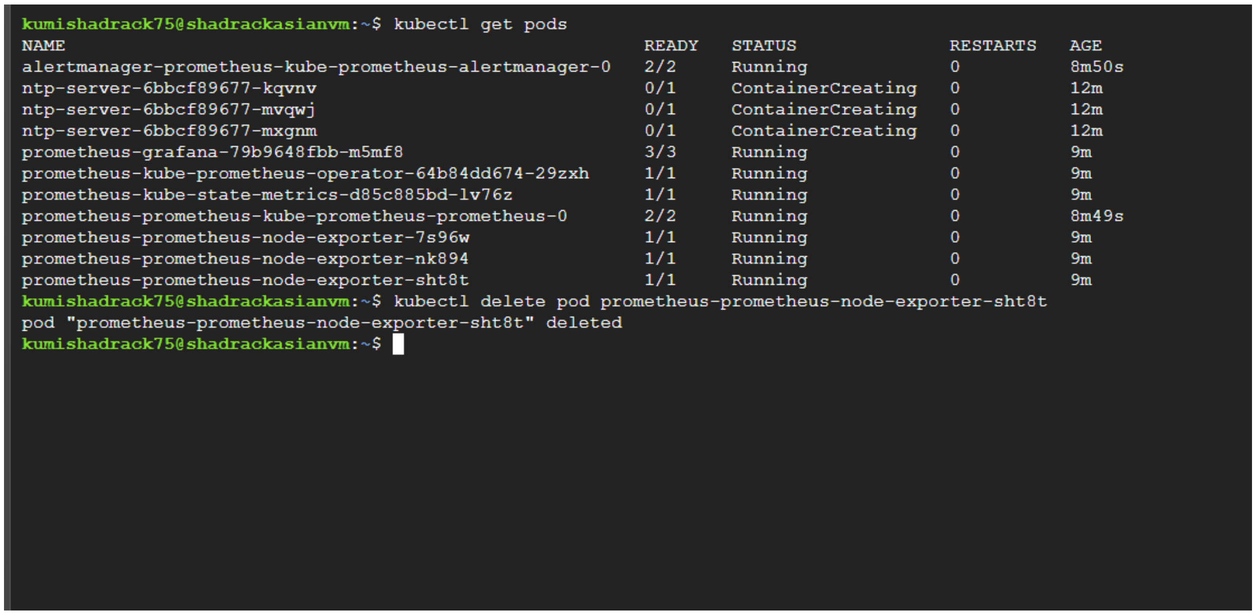


```
2024-12-18 16:37:27.604998 (+0000) +0.013450 +/- 0.001862 34.81.163.131 s4 no-leap
kumishadrack75@shadrackasianvm:~$
```

Test High Availability:

- **Delete a Pod and Verify High Availability:**

kubectl delete pod <ntp_pod_name>



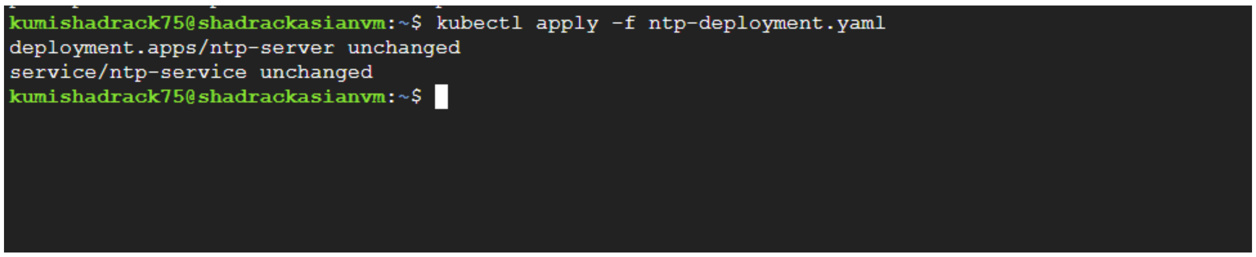
```
kumishadrack75@shadrackasianvm:~$ kubectl get pods
NAME                                                     READY   STATUS              RESTARTS   AGE
alertmanager-prometheus-kube-prometheus-alertmanager-0   2/2     Running             0          8m50s
ntp-server-6bbcf89677-kqvnv                              0/1     ContainerCreating   0          12m
ntp-server-6bbcf89677-mvqwj                              0/1     ContainerCreating   0          12m
ntp-server-6bbcf89677-mxgnm                              0/1     ContainerCreating   0          12m
prometheus-grafana-79b9648fbb-m5mf8                      3/3     Running             0          9m
prometheus-kube-prometheus-operator-64b84dd674-29zxh     1/1     Running             0          9m
prometheus-kube-state-metrics-d85c885bd-lv76z            1/1     Running             0          9m
prometheus-prometheus-kube-prometheus-prometheus-0       2/2     Running             0          8m49s
prometheus-prometheus-node-exporter-7s96w                1/1     Running             0          9m
prometheus-prometheus-node-exporter-nk894                1/1     Running             0          9m
prometheus-prometheus-node-exporter-sht8t                1/1     Running             0          9m
kumishadrack75@shadrackasianvm:~$ kubectl delete pod prometheus-prometheus-node-exporter-sht8t
pod "prometheus-prometheus-node-exporter-sht8t" deleted
kumishadrack75@shadrackasianvm:~$
```

Ensure Load Balancer Routes Traffic to Healthy Pods: The external IP from the LoadBalancer should continue to serve requests even after one or more pods are deleted.

Test Automatic Update:

- **Trigger a Deployment Update:**

kubectl apply -f ntp-deployment.yaml



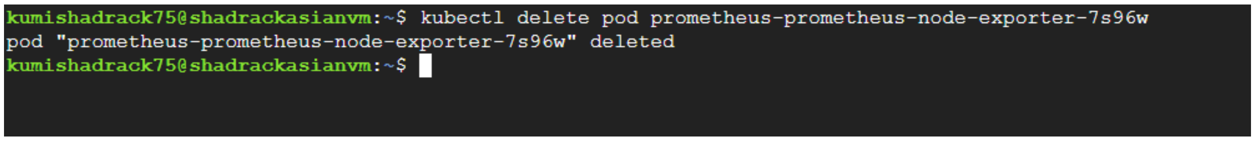
```
kumishadrack75@shadrackasianvm:~$ kubectl apply -f ntp-deployment.yaml
deployment.apps/ntp-server unchanged
service/ntp-service unchanged
kumishadrack75@shadrackasianvm:~$
```

- **Verify No Downtime:** Check the logs to confirm that the update happens without interrupting service.

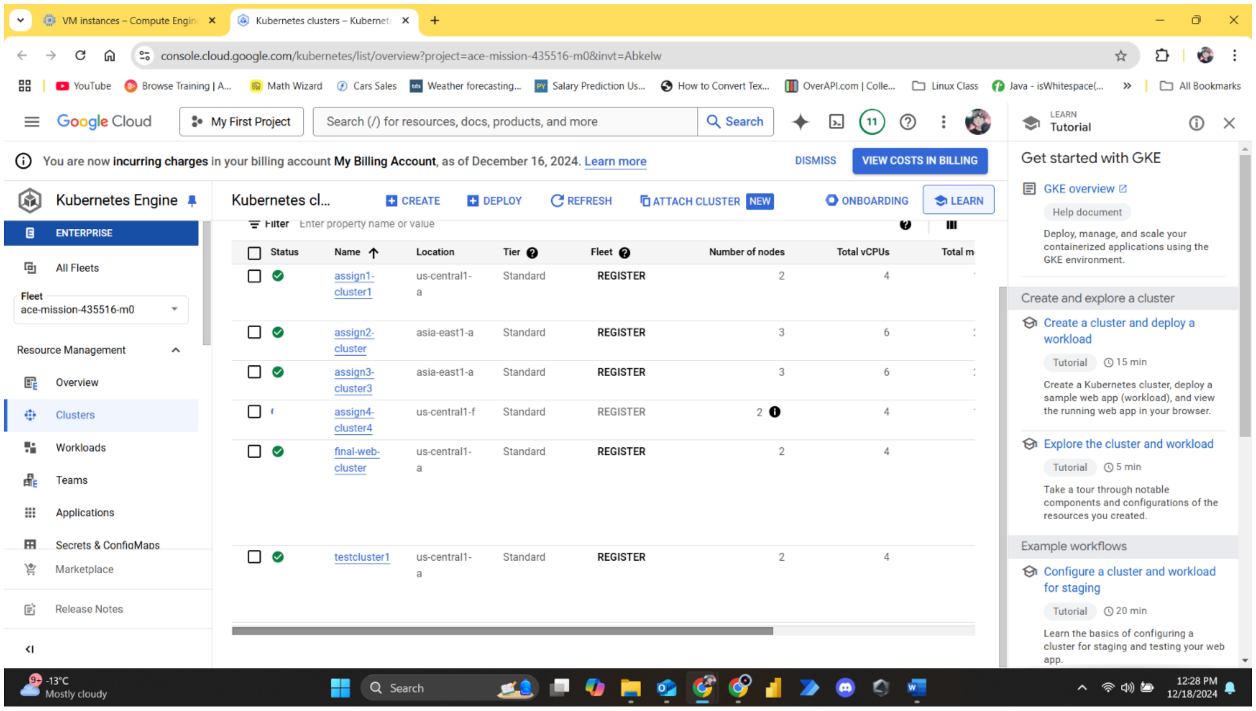
Test Monitoring Alerts:

- **Simulate a CPU Spike or Failure (e.g., by running a stress test or deleting pods) and Verify Alerts:**

kubectl delete pod <ntp_pod_name>



```
kumishadrack75@shadrackasianvm:~$ kubectl delete pod prometheus-prometheus-node-exporter-7s96w
pod "prometheus-prometheus-node-exporter-7s96w" deleted
kumishadrack75@shadrackasianvm:~$ 
```



You are now incurring charges in your billing account My Billing Account, as of December 16, 2024. Learn more

	Status	Name	Location	Tier	Fleet	Number of nodes	Total vCPUs	Total m
☐	✓	assign1-cluster1	us-central1-a	Standard	REGISTER	2	4	
☐	✓	assign2-cluster	asia-east1-a	Standard	REGISTER	3	6	
☐	✓	assign3-cluster3	asia-east1-a	Standard	REGISTER	3	6	
☐		assign4-cluster4	us-central1-f	Standard	REGISTER	2	4	
☐	✓	final-web-cluster	us-central1-a	Standard	REGISTER	2	4	
☐	✓	testcluster1	us-central1-a	Standard	REGISTER	2	4	

SUMMARY

Setting Up the GKE Cluster:

I created a Google Kubernetes Engine (GKE) cluster with at least three nodes, ensuring proper network configuration and firewall rules to allow external access to the NTP servers.

2. Deploying the NTP Server Containers:

I chose a suitable NTP server container image and created a Kubernetes Deployment manifest to define three replicas of the NTP server.

I also created a Service of type LoadBalancer to expose the NTP servers on port 123 (UDP).

After applying the ntp-deployment.yaml and ntp-service.yaml files, I confirmed that the NTP service was running successfully with an external IP, although the external IP was initially pending, which later got resolved.

3. Installing and Configuring Prometheus:

I attempted to install Prometheus using Helm but faced issues due to Helm not being installed. I worked through these problems by installing Helm with the proper commands.

I added the Prometheus Helm repository using helm repo add and updated the repositories with helm repo update.

I installed the Prometheus kube-prometheus-stack using Helm after resolving the repository issues.

4. Configuring Prometheus Alerting:

I created alerting rules for Prometheus to monitor high CPU usage on the pods by writing custom rules in a alerting-rules.yaml file. This included a rule for a HighCPUUsage alert, triggered when the CPU usage exceeds 90% for 5 minutes.

I then integrated the alerting rules into the Prometheus configuration via a custom values.yaml file and used Helm to upgrade the Prometheus stack with the new configuration.

5. Configuring Alertmanager for Notifications:

To notify users of triggered alerts, I configured Alertmanager to send alerts to Slack by creating an alertmanager-config.yaml file with the necessary settings.

I modified the values.yaml for the Helm chart to include the Alertmanager configuration and set up Slack notifications for the alerts.

6. Testing:

I tested the NTP server by querying it using ntpdate or chronyc from a client machine to ensure the time synchronization was accurate.

I simulated pod failures to verify the high availability of the NTP server cluster and tested the automatic updates by triggering deployments to see that the servers updated without downtime.

Finally, I validated the alerting setup by simulating high CPU usage and confirmed that the alert appeared in Prometheus and was sent to the Slack channel.

This entire process helped me gain hands-on experience with deploying, managing, and automating containerized applications on GKE, including setting up monitoring, alerting, and high availability.