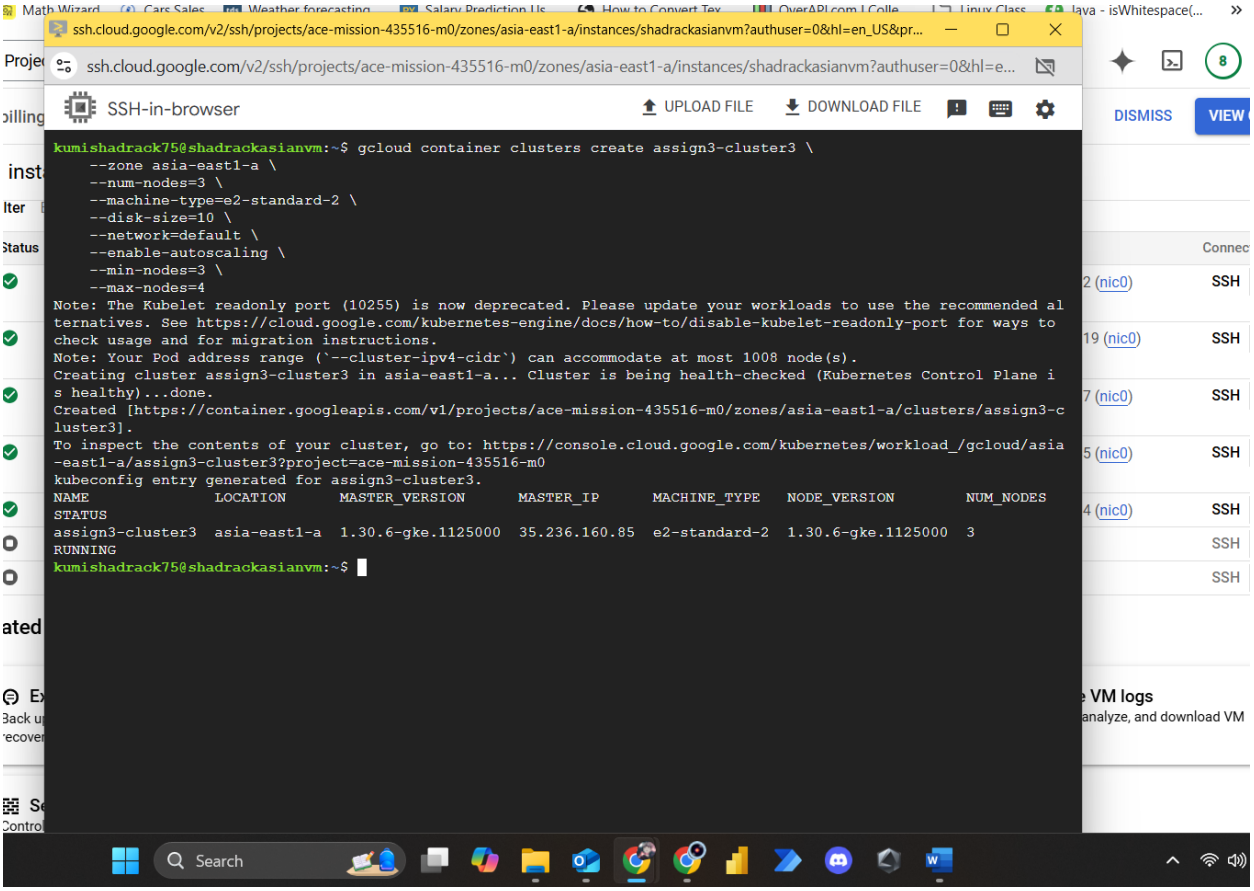
1. Create a GKE Cluster

First, create a Google Kubernetes Engine (GKE) cluster with at least three nodes.

Command to create the GKE cluster:

```
gcloud container clusters create assign3-cluster3 \
  --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
```



Network Configuration:

Ensure that the cluster is set up with the correct VPC and firewall rules.

Create a VPC network (if it does not exist):

gcloud compute networks create ntp-vpc --subnet-mode=auto

```
kumishadrack75@shadrackasianvm:~$ gcloud compute networks create ntp-vpc --subnet-mode=auto
ERROR: (gcloud.compute.networks.create) Could not fetch resource:
 - The resource 'projects/ace-mission-435516-m0/regions/us-central1/subnetworks/ntp-vpc' already exists
```

Create firewall rules to allow UDP traffic on port 123 (the NTP port):

```
kumishadrack75@shadrackasianvm:~$ gcloud compute firewall-rules create allow-ntp-traffic \
    --network ntp-vpc \
    --allow udp:123 \
    --source-ranges 0.0.0.0/0 \
    --target-tags ntp-server
Creating firewall...Created [https://www.googleapis.com/compute/v1/projects/ace-mission-435516-m0/global/firewal
ls/allow-ntp-traffic].
Creating firewall...done.
NAME                NETWORK  DIRECTION  PRIORITY  ALLOW    DENY  DISABLED
allow-ntp-traffic   ntp-vpc  INGRESS    1000      udp:123        False
kumishadrack75@shadrackasianvm:~$
```

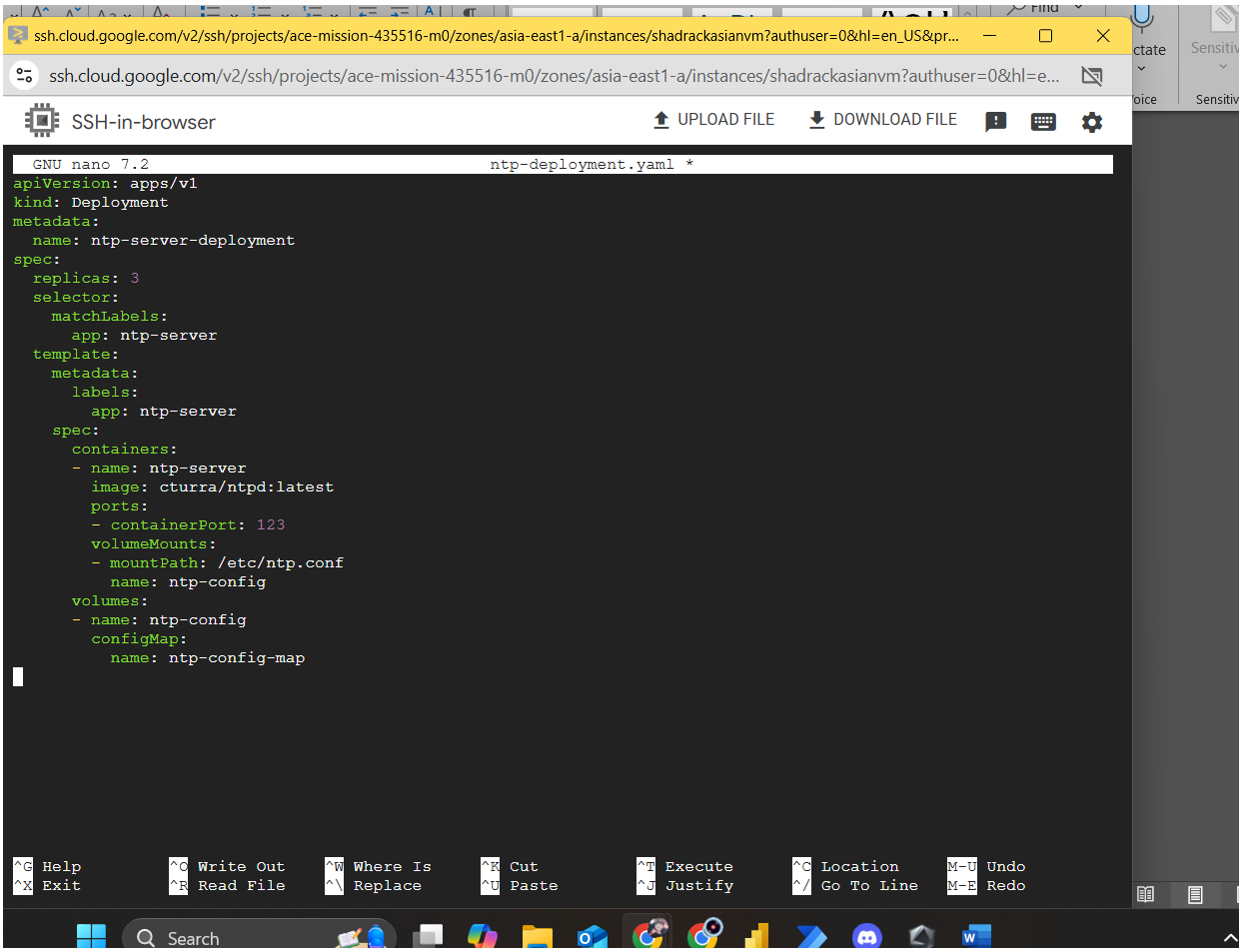
2. Deploy the NTP Server Containers

Choose a Container Image:

We will use the cturra/ntpd image from Docker Hub for this assignment.

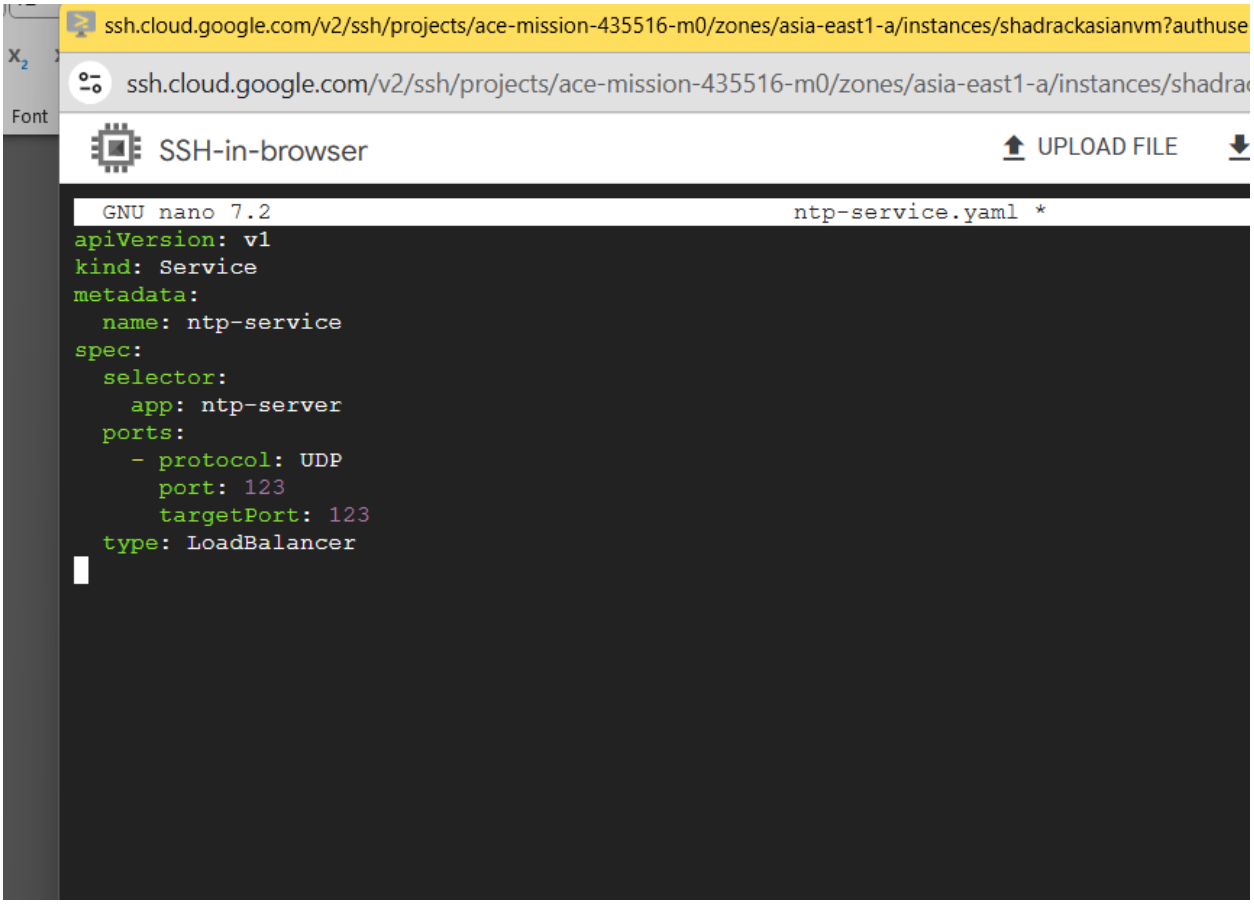
Kubernetes Deployment Manifest (ntp-deployment.yaml):

This YAML file defines the deployment of the NTP server with three replicas.



Kubernetes Service Manifest (ntp-service.yaml):

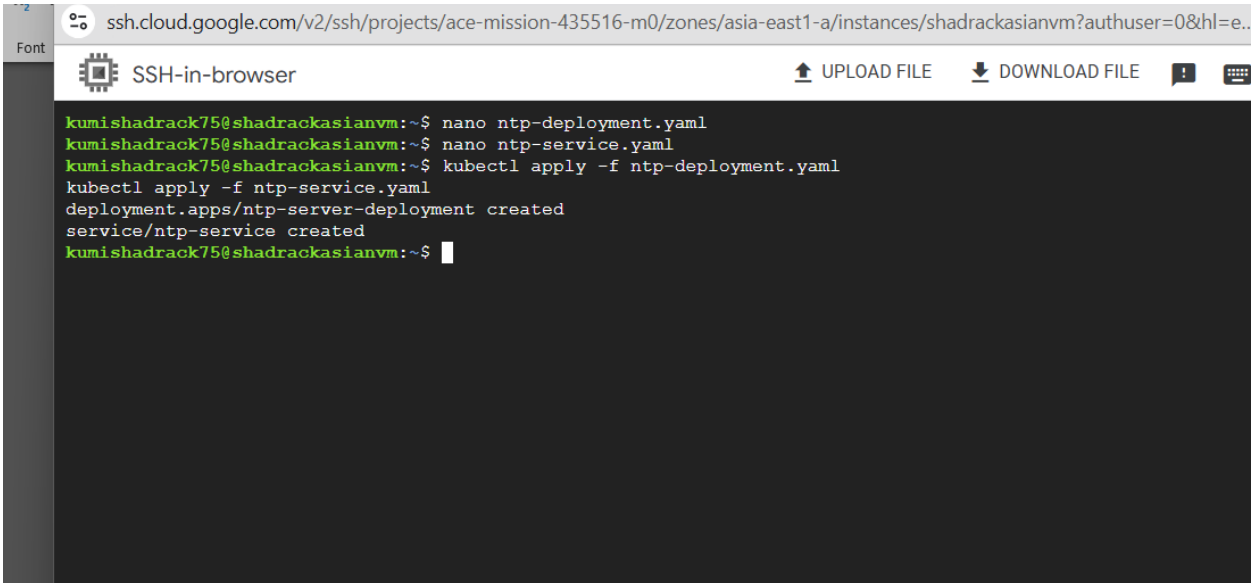
This YAML file creates a LoadBalancer service to expose the NTP server



Apply the Kubernetes Deployment and Service:

kubectl apply -f ntp-deployment.yaml

kubectl apply -f ntp-service.yaml



3. Configure the NTP Server

If the chosen container image allows customization (e.g., modifying ntp.conf), you can create a ConfigMap with the custom configuration.

Example ConfigMap for ntp.conf (ntp-config-map.yaml):

```
  GNU nano 7.2                                  ntp-config-map.yaml *
apiVersion: v1
kind: ConfigMap
metadata:
  name: ntp-config-map
data:
  ntp.conf: |
    # Use specific NTP time sources
    server time.google.com iburst
    server time.windows.com iburst
    restrict default kod nomodify notrap nopeer noquery

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

Apply the ConfigMap:

```
umishadrack75@shadrackasianvm:~$ nano ntp-deployment.yaml
umishadrack75@shadrackasianvm:~$ nano ntp-service.yaml
umishadrack75@shadrackasianvm:~$ kubectl apply -f ntp-deployment.yaml
ubectl apply -f ntp-service.yaml
eployment.apps/ntp-server-deployment created
ervice/ntp-service created
umishadrack75@shadrackasianvm:~$ nano ntp-config-map.yaml
umishadrack75@shadrackasianvm:~$ kubectl apply -f ntp-config-map.yaml
onfigmap/ntp-config-map created
umishadrack75@shadrackasianvm:~$
```

4. Implement Automatic Updates

Kubernetes Rolling Update:

The rolling update strategy is configured by default in Kubernetes deployments. When you update the deployment, Kubernetes will perform the update with no downtime by updating one pod at a time.

To update the NTP server (e.g., to a new version of the container image), use:

kubectl set image deployment/ntp-server-deployment ntp-server=cturra/ntpd:latest

CI/CD Pipeline (Optional):

You can set up a CI/CD pipeline using Jenkins, GitLab, or CircleCI. Here's an example of a simple GitLab CI/CD pipeline configuration (.gitlab-ci.yml)

stages:

- deploy

deploy:

stage: deploy

script:

- kubectl set image deployment/ntp-server-deployment ntp-server=cturra/ntpd:latest

only:

- main

This pipeline automatically deploys new images when changes are pushed to the main branch.

5. Test the NTP Server Cluster

Test Time Synchronization:

On a client machine, use ntpdate or chronyc to verify time synchronization from the LoadBalancer IP of the NTP service.

To get the external IP of the LoadBalancer, run:

```
kumishadrack75@shadrackasianvm:~$ sudo ntpdate -q 34.81.163.131
2024-12-18 16:37:27.604998 (+0000) +0.013450 +/- 0.001862 34.81.163.131 s4 no-leap
kumishadrack75@shadrackasianvm:~$ kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
ntp-server-7bb5bdb4c7-482cc       1/1     Running   0          6m32s
ntp-server-7bb5bdb4c7-lcbdk       1/1     Running   0          6m32s
ntp-server-7bb5bdb4c7-m4sh9       1/1     Running   0          6m32s
kumishadrack75@shadrackasianvm:~$
```

Test High Availability:

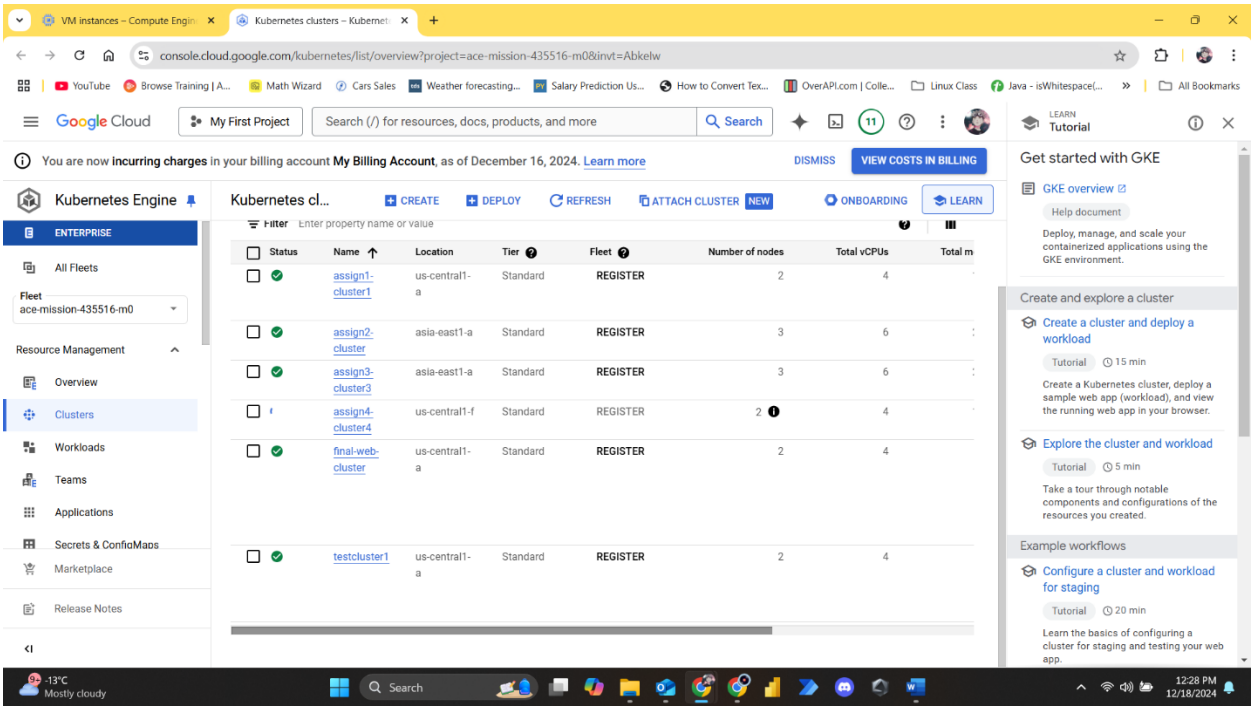
To test high availability, delete one of the NTP server pods and ensure that the LoadBalancer still routes traffic to the remaining healthy pods.

kubectl get pods

kubectl delete pod <pod_name>

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

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



The LoadBalancer should automatically route traffic to the other NTP server pods.

Test Automatic Update:

To test the rolling update process, trigger a deployment update:

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

kubectl set image deployment/ntp-server-deployment ntp-server=cturra/ntpd:<new-version>

SUMMARY

1. Deployment Setup for NTP Server: I started by working on deploying an NTP server on my Google Kubernetes Engine (GKE) cluster. Initially, I used the cturra/ntpd:latest image for the NTP server. I created the necessary Kubernetes deployment and service files, including a ConfigMap for configuration settings. However, I faced several issues related to pulling the Docker image.
2. Image Pull Issues: I encountered errors such as ImagePullBackOff when trying to pull the cturra/ntpd:latest image. This issue arose because the Docker image could not be found or had access restrictions. I tried several steps to fix the issue, including checking the service account permissions and attempting to use docker login to access Docker Hub, but the image still failed to pull.
3. Docker Daemon Permission Issues: I had trouble accessing Docker because of permission issues. To resolve this, I added my user to the Docker group and rebooted the system. This allowed me to run Docker commands without using sudo, but I still encountered issues with image access, resulting in error messages like "repository does not exist or may require 'docker login'."
4. ConfigMap Setup: I created a ConfigMap for the chrony.conf configuration and successfully applied it to the cluster. However, I faced mounting errors when Kubernetes tried to use this ConfigMap in the deployment. I ensured the ConfigMap was created and correctly applied.
5. Testing and Debugging: After deploying the NTP server, I attempted to test it by using the ntpdate command to sync time with the external IP of the NTP service, but encountered errors such as "no eligible servers." I also tried to pull the cturra/ntpd:latest image on Docker, but encountered access issues and eventually switched to troubleshooting Docker login credentials and access rights.

Next Steps: I realized that the image cturra/ntpd was either unavailable or private, so I recommended using an alternative public image like bitnami/chrony. I also suggested

checking the Docker Hub credentials and using a different image if the current one remained inaccessible.