**ON-PREM K8s CLUSTER USING VMs**

This project demonstrates the use of ‘kubeadm’ to deploy a Kubernetes cluster of 1 master and 2 worker nodes with multi-tier web application deployed and monitoring with Prometheus and Grafana.

**Prerequisites (host pc):**

Install

-Gitbash

-Virtualbox

-Vagrant

**NOTE:** host pc should also have more than 120GB free disk space and 16GB ram

**Step 1:**

[create vms and initialize cluster with kubeadm]

###On gitbash clone repo:

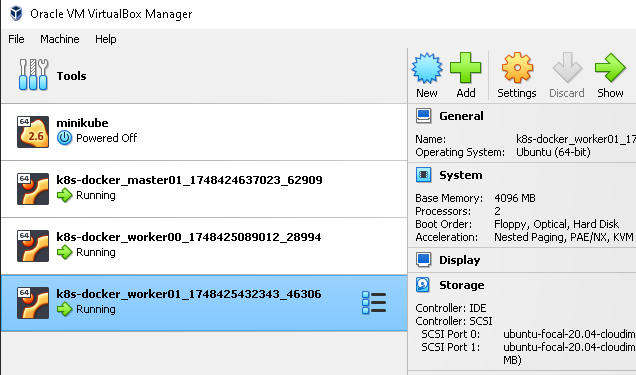
git clone -b main <https://github.com/yobami12/Aprofile-Project.git>

cd Aprofile-Project/k8s-docker

vagrant plugin install vagrant-disksize

vagrant plugin install vagrant-hostmanager

vagrant up



###reboot all vms

**Step 2:**

[add worker nodes to the cluster]

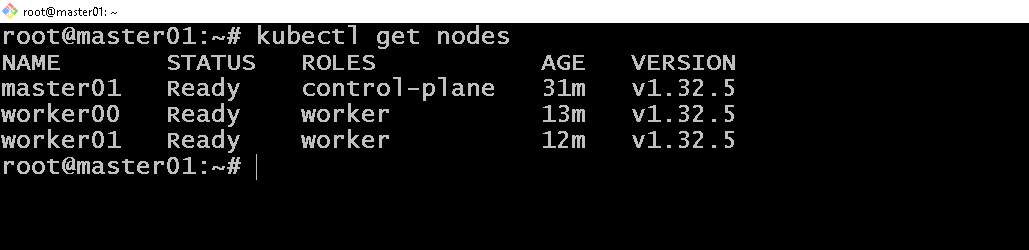
###login to master node vm(master01) and run:

kubeadm token create --print-join-command

###copy the output of the ‘kubeadm token …’ command above**;** login and paste it to each worker node to add them to the cluster.

###verify on the master node that the worker nodes have been added.

Kubectl get nodes



**Step 3:**

[deploy multi-tier web application in ‘default’ namespace]

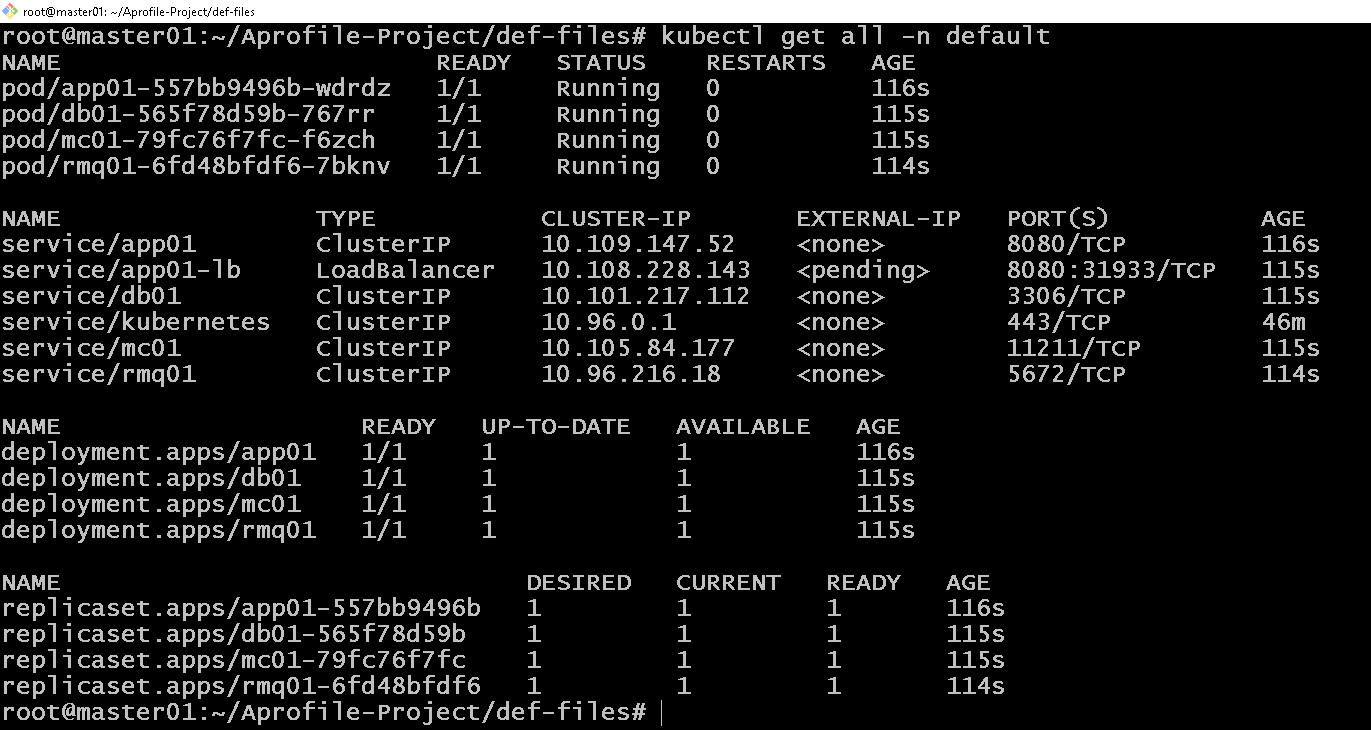
###on master node vm run:

git clone -b main <https://github.com/yobami12/Aprofile-Project.git>

cd Aprofile-Project/def-files

kubectl create -f **.**

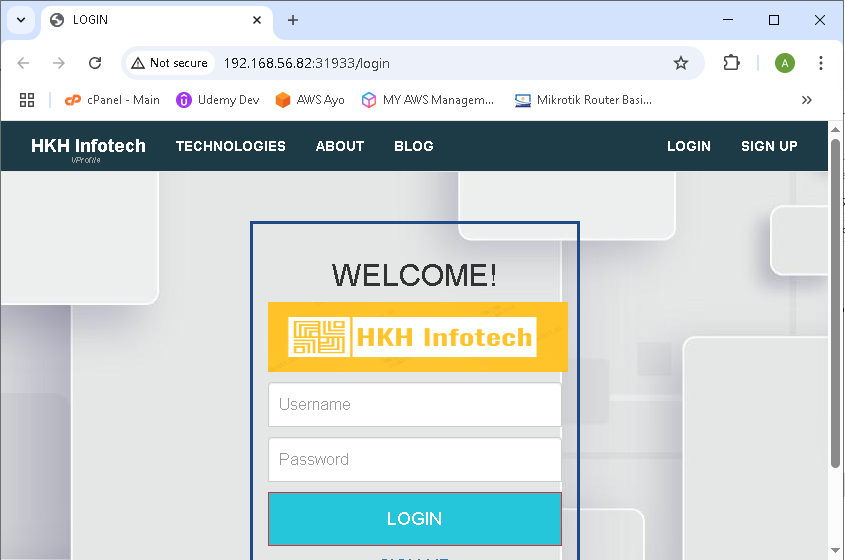
Kubectl get all -n default



###test application on the browser with url:

<http://192.168.56.82:31933>

**Note:** “**:31933**” is the auto assigned port to the “app01-lb” in this cluster. Replace with yours.



Username: admin\_vp

Password: admin\_vp

**Step 4:**

[deploy prometheus and grafana for monitoring]

**PROMETHEUS**

###create PVs - on master node vm run:

cat <<EOF > prometheus-pv.yaml

apiVersion: v1

kind: PersistentVolume

metadata:

name: storage-volume

spec:

capacity:

storage: 8Gi

accessModes:

- ReadWriteOnce

persistentVolumeReclaimPolicy: Retain

**storageClassName: standard**

hostPath:

path: /data

EOF

cat <<EOF > storage-pv.yaml

apiVersion: v1

kind: PersistentVolume

metadata:

name: storage-prometheus-pv

spec:

capacity:

storage: 2Gi

accessModes:

- ReadWriteOnce

persistentVolumeReclaimPolicy: Retain

**storageClassName: standard**

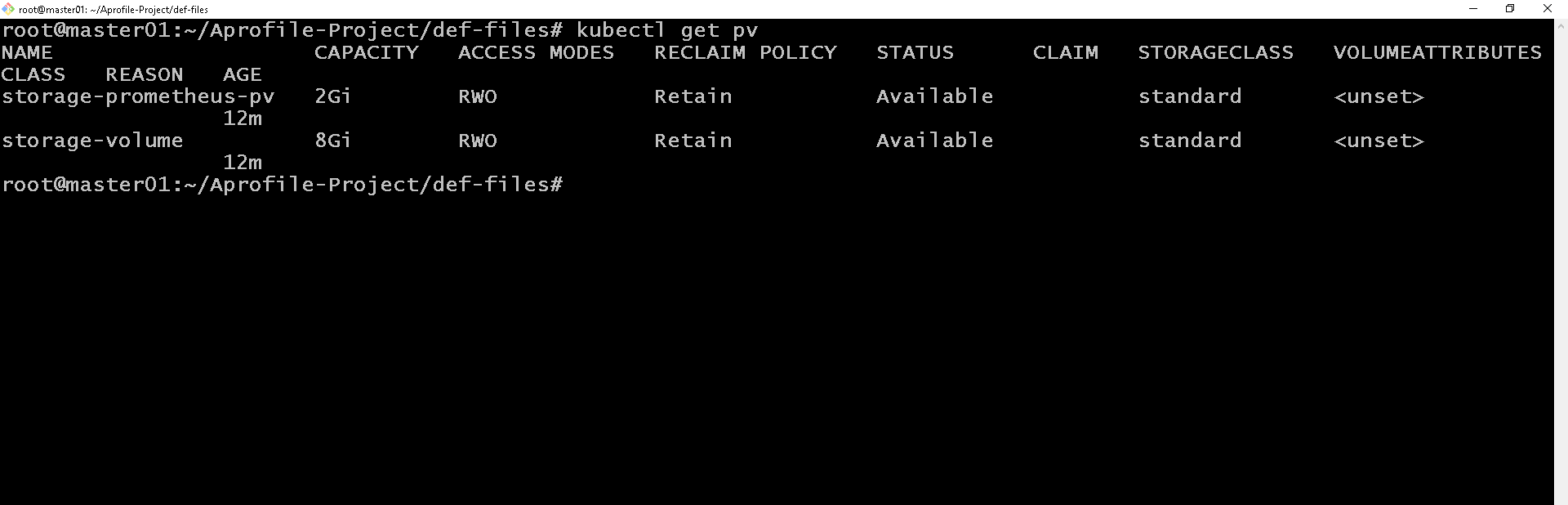
hostPath:

path: /data

EOF

kubectl create -f prometheus-pv.yaml,storage-pv.yaml

kubectl get pv



###add prometheus and grafana repo using helm cmd:

helm repo add prometheus-community <https://prometheus-community.github.io/helm-charts>

helm repo add grafana https://grafana.github.io/helm-charts

helm repo update

###install and deploy prometheus to cluster

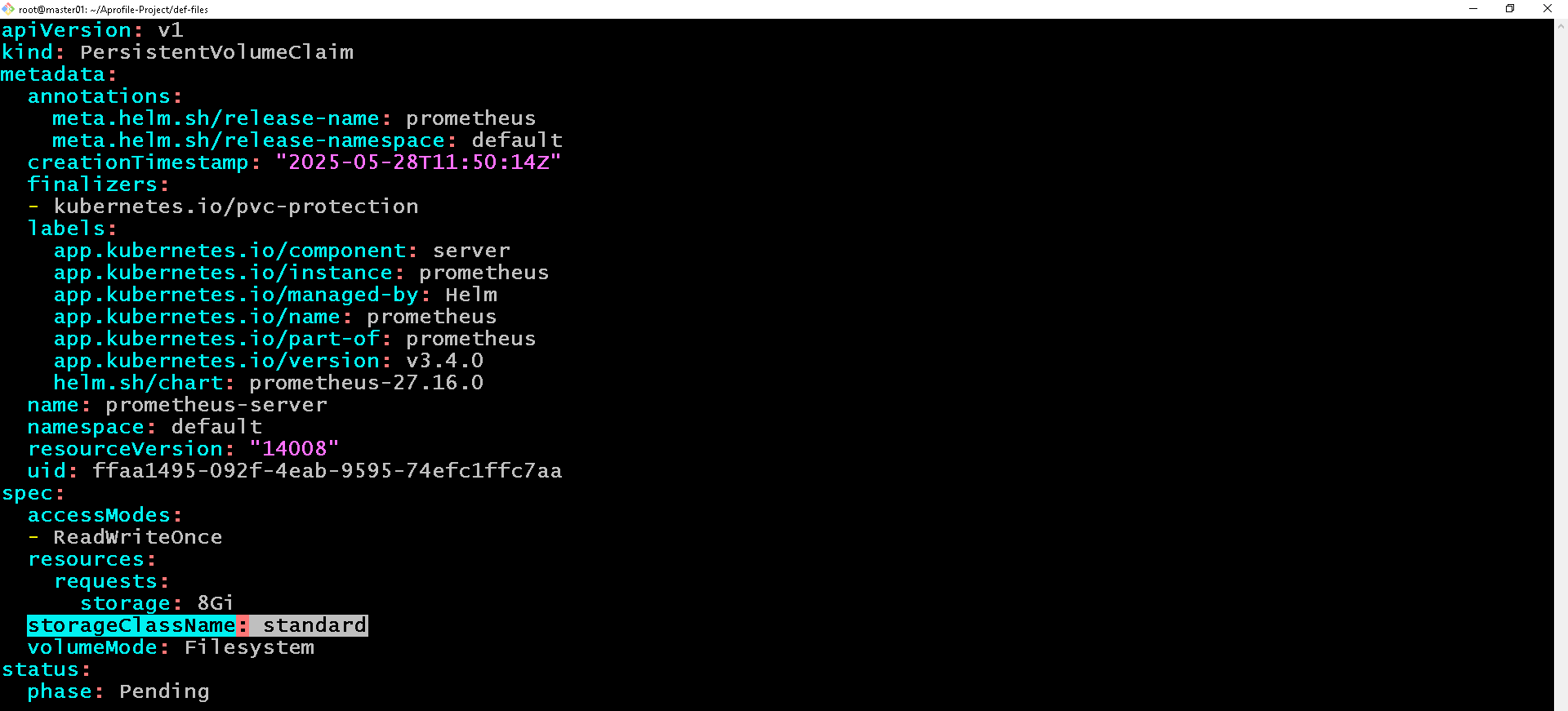
helm install prometheus prometheus-community/prometheus

**Note:** both ‘prometheus-alertmanager’ and ‘prometheus-server’ pods will be in pending state because the created pv and pvc are yet to be bound.

###Update pvc with created pv’s "storageClassName" so they can be bound.

kubectl edit pvc prometheus-server

###add “**storageClassName: standard**” as in below img



***add the highlighted line in your pvc yaml file just as in img. Save and quit.***

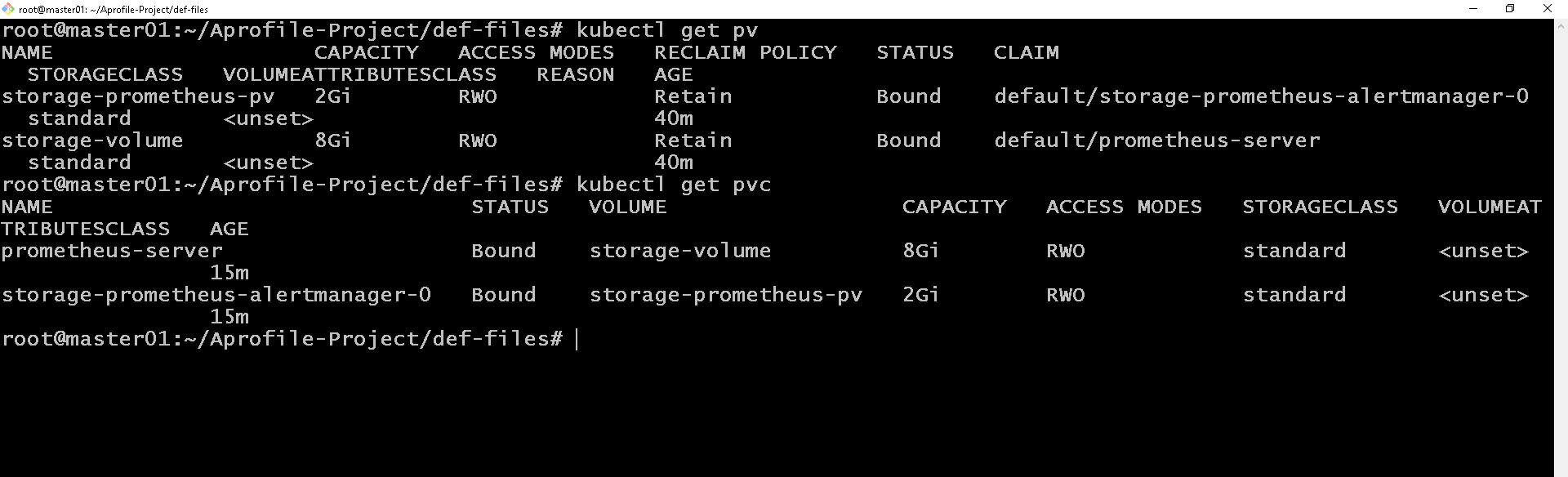
###do same here

kubectl edit pvc storage-prometheus-alertmanager-0

###verify that both pv and pvc are in ‘**bound**’ state

Kubectl get pv

Kubectl get pvc



***Both ‘prometheus-alertmanager’ and ‘prometheus-server’ pods should be running now because the created pv and pvc are in bound state.***

###but if the ‘prometheus-server’ pod is in ‘CrashLoopBackOff’ state.

Kubectl get pods



***prometheus-server in CrashLoopBackOff***

###edit ‘prometheus-server’ deployment file

kubectl edit deploy prometheus-server

“

fsGroup: 0

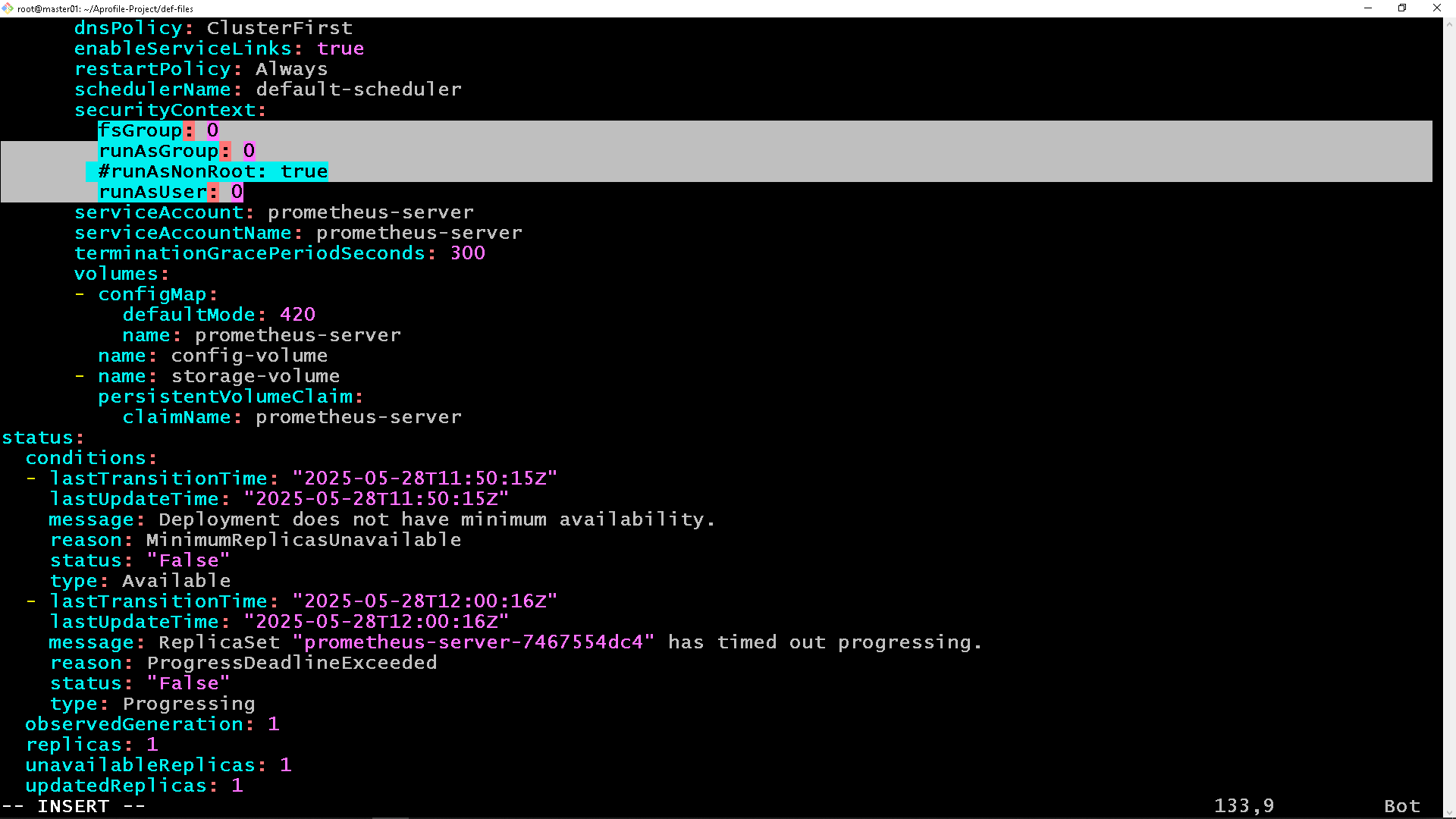
runAsGroup: 0

#runAsNonRoot: true

runAsUser: 0

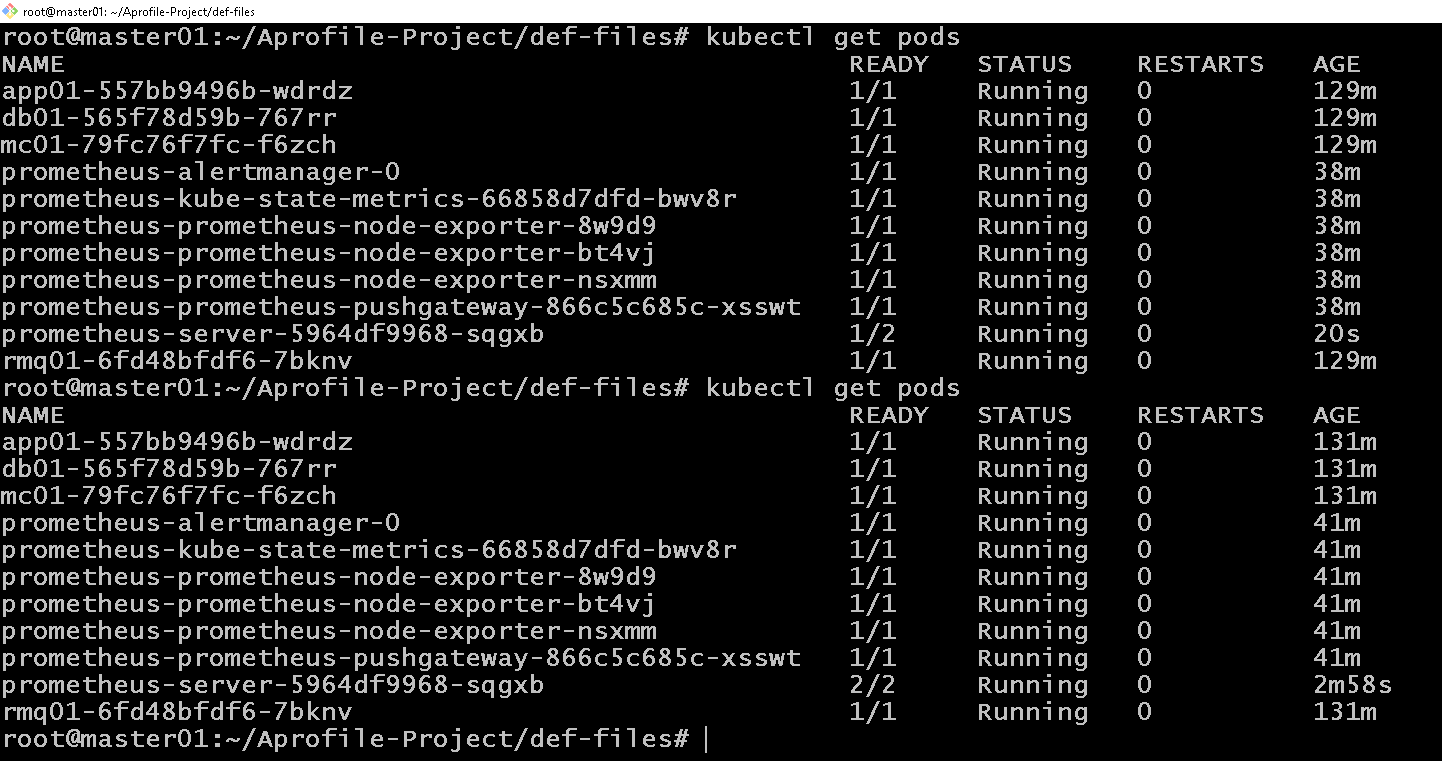
“

###scroll to lines with the above and update value as ‘0’ (digit) as in above.

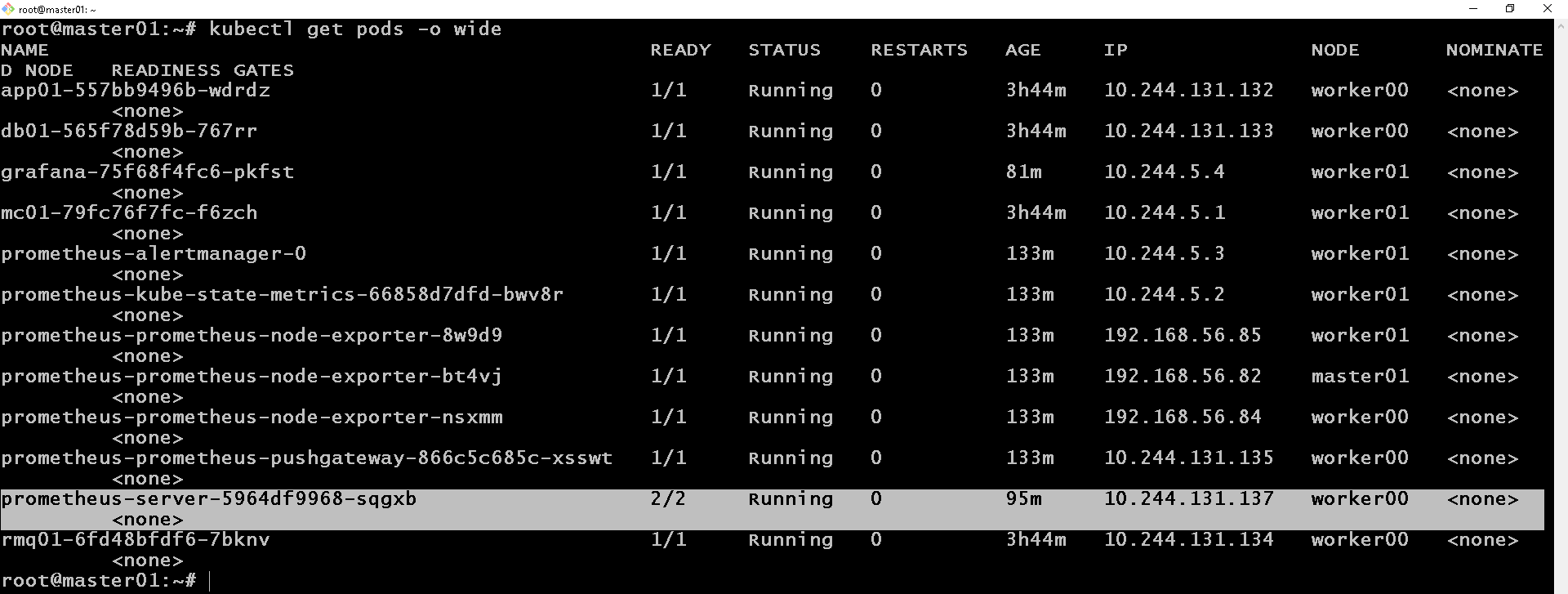


***save and quit***

Kubectl get pods



***Prometheus-server pod now running***

****

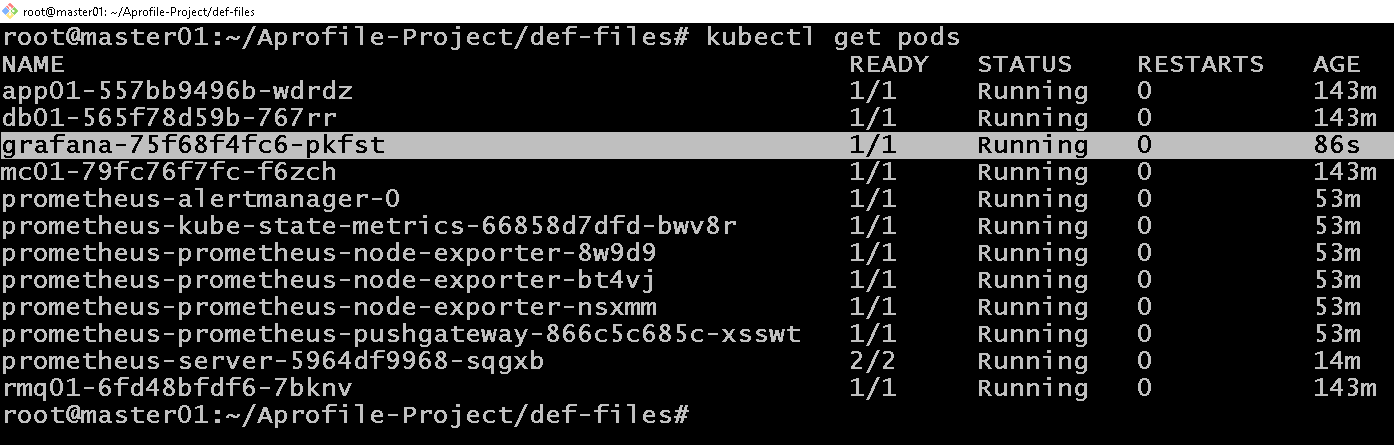
**GRAFANA**

###install and deploy grafana to the cluster

helm install grafana grafana/grafana

###verify grafana pod is running

kubectl get pods



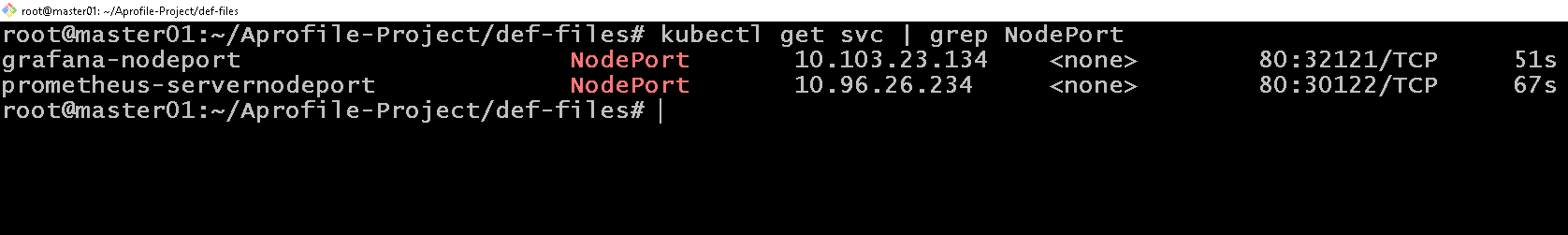
###create NodePort service for both ‘prometheus-server’ and ‘grafana’.

kubectl expose service prometheus-server --type=NodePort --target-port=9090 --name=prometheus-servernodeport

kubectl expose service grafana --type=NodePort --target-port=3000 --name=grafana-nodeport

###verify NodePort service creation

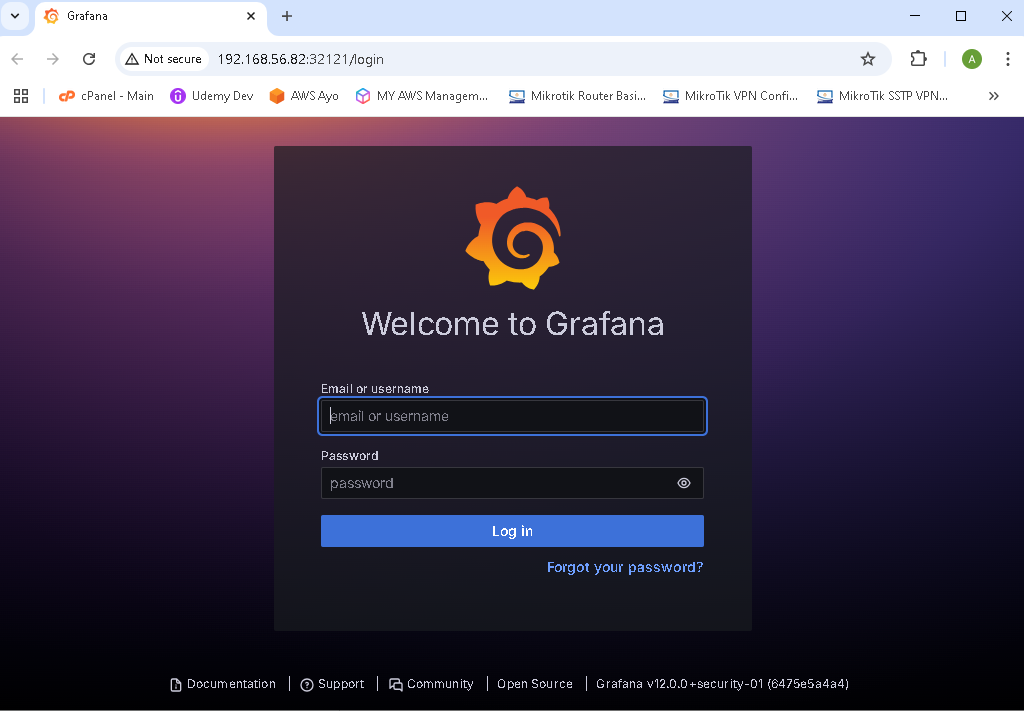
kubectl get svc | grep NodePort



###access grafana on

<http://192.168.56.82:32121>

**Note:** replace “32121” with yours



###run below command to get login password for grafana

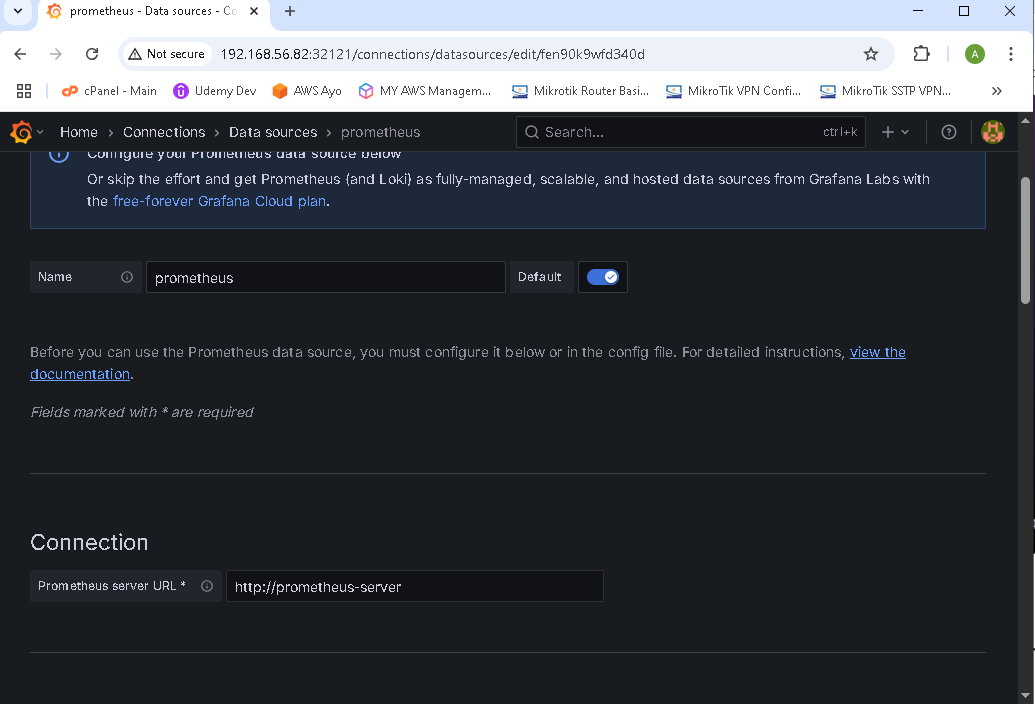
kubectl get secret --namespace default grafana -o jsonpath="{.data.admin-password}" | base64 --decode ; echo

###username = admin

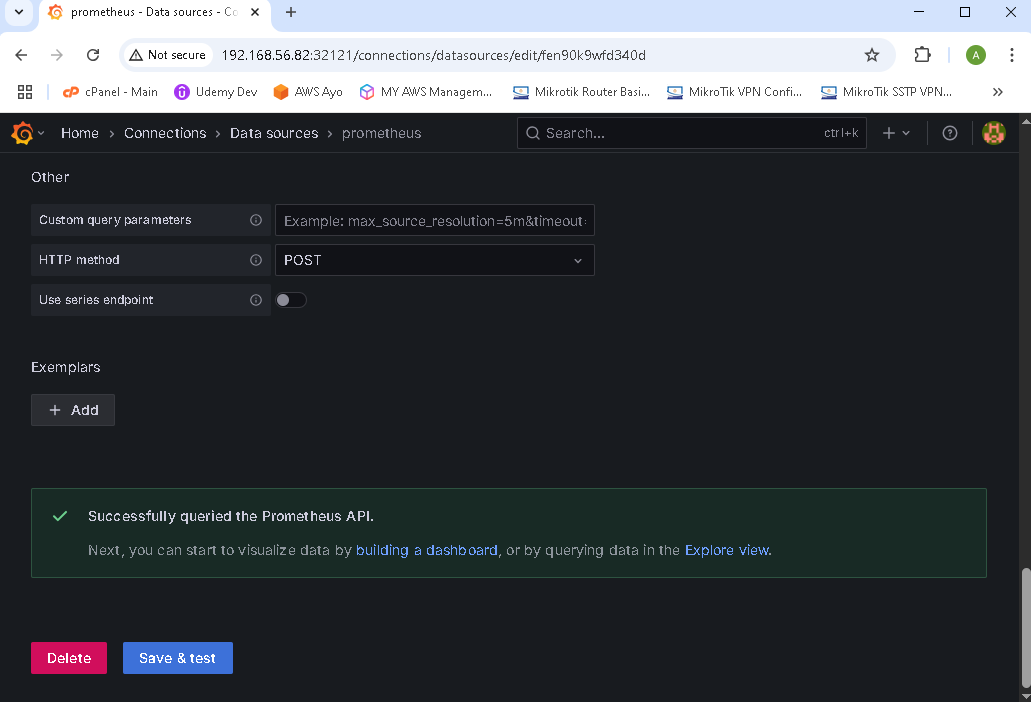
###add ‘prometheus-server as data source in grafana

-on grafana home page in the browser, goto datasource, select prometheus and add 'prometheus-server' as datasource with url 'http://prometheus-server' (prometheus-server = cluster service name for prometheus server)

-then 'save and test' (must successfully query).



***Prometheus server url:*** [***http://prometheus-server***](http://prometheus-server)



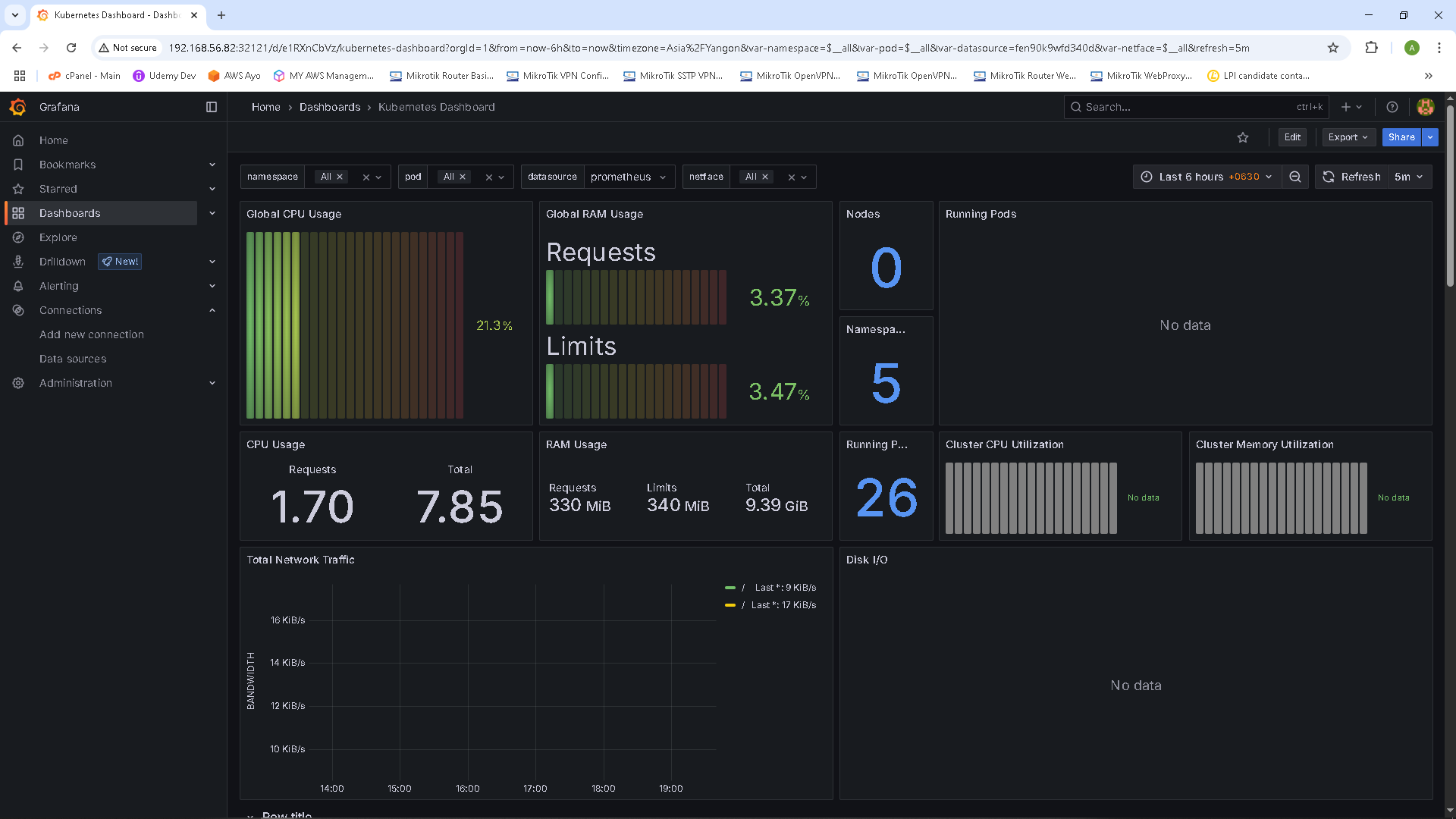
***Prometheus server successfully queried after clicking ‘save & test’.***

###Create dashboard

-goto google and search 'grafana dashboards for kubernetes'. select links from grafana.com

-copy dashboard ID, goto your grafana ui, on the home page, click on 'create your first dashboard',

-click on 'import dashboard', paste the copied id, select created data source and click on 'load'



***Grafana dashborad for kubernetes created.***