**Kubernetes Deployment with NFS Storage for MySQL and WordPress**

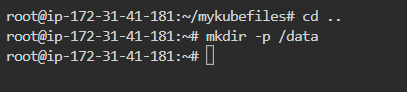
**NFS Storage and HostPath Storage with MySQL-WordPress Deployment PART OF THE PROJECT**

**Setting Up NFS Storage on the Master Node:**

1. **Create an NFS server:**

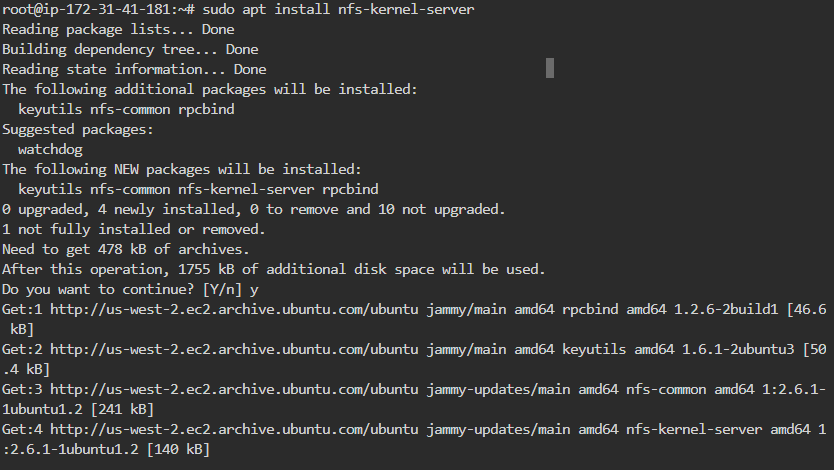
sudo su -

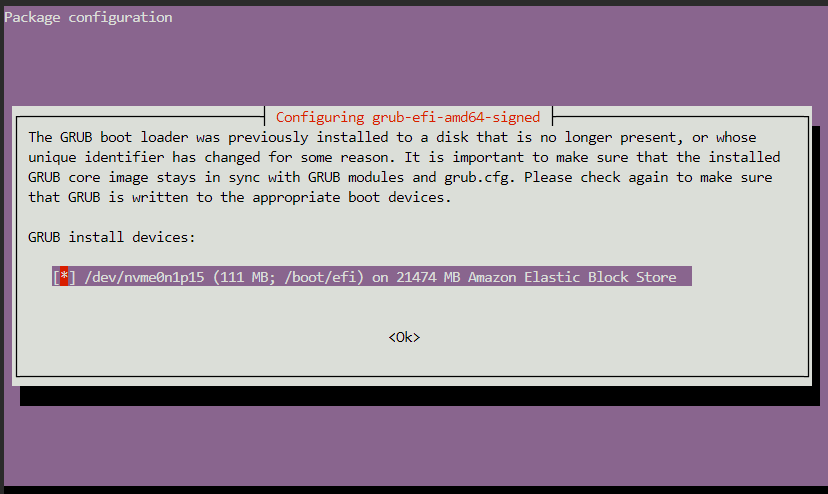
mkdir -p /data

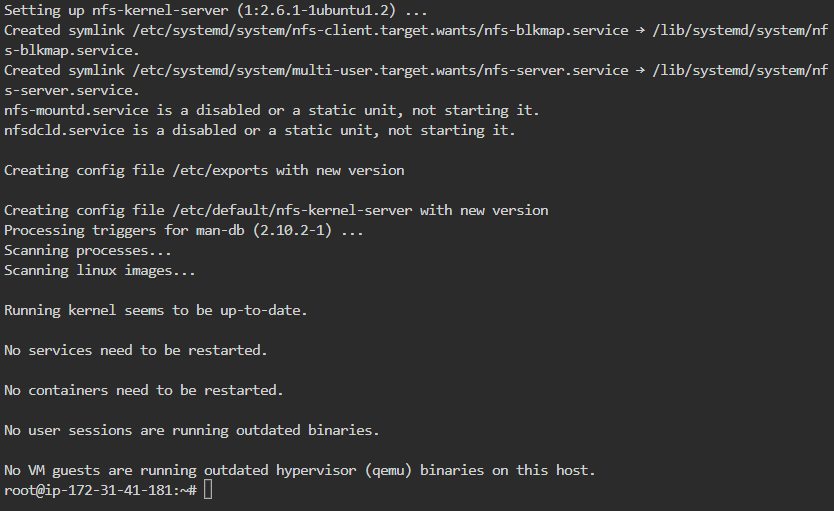


1. **Install the NFS kernel server:**

sudo apt install nfs-kernel-server



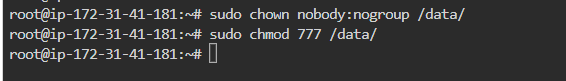




1. **Change ownership and permissions:**

sudo chown nobody:nogroup /data/

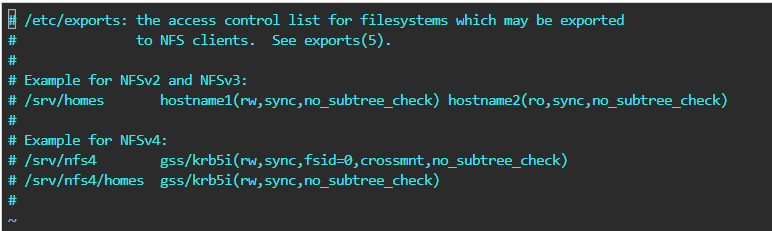
sudo chmod 777 /data/



1. **Configure access permissions:**

sudo vi /etc/exports

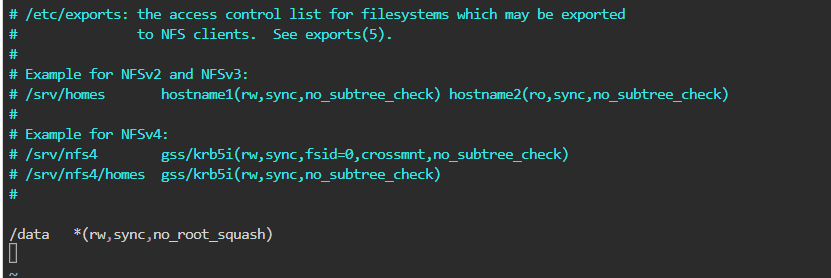




To be updated as below:

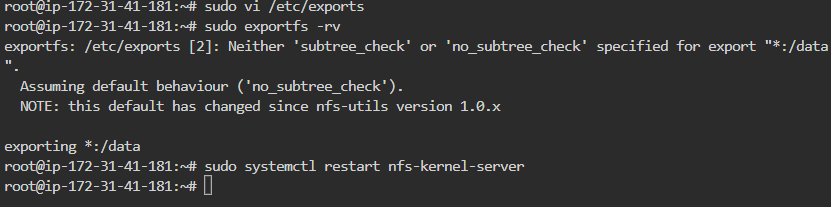
**Add the following entry:**

/data \*(rw,sync,no\_root\_squash)



sudo exportfs -rv

sudo systemctl restart nfs-kernel-server

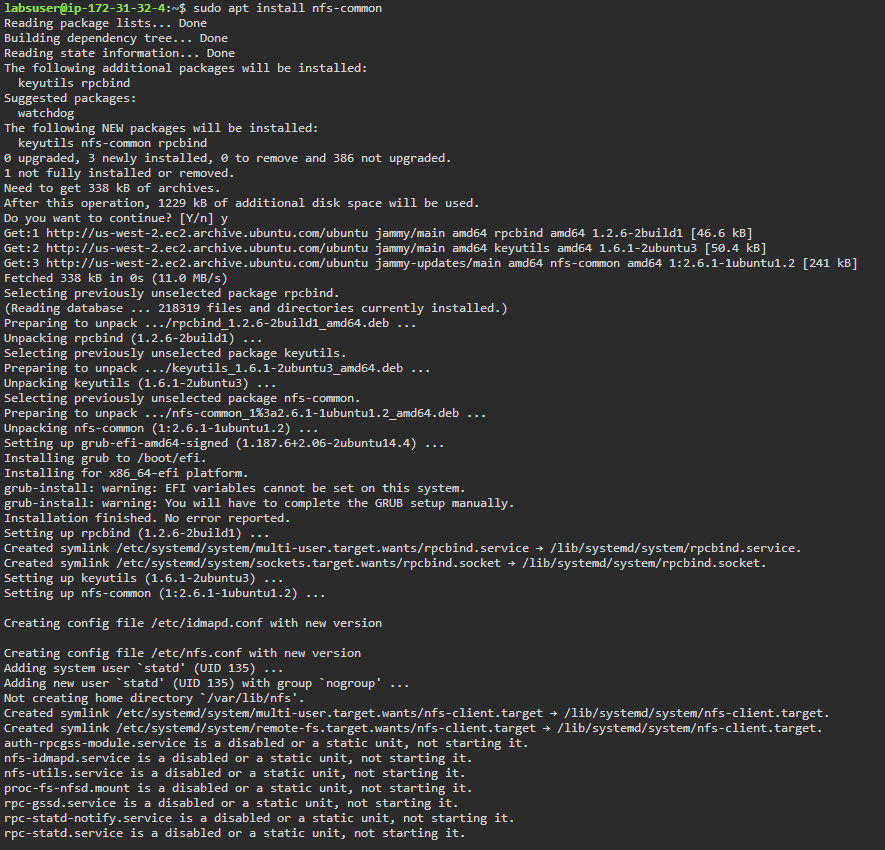


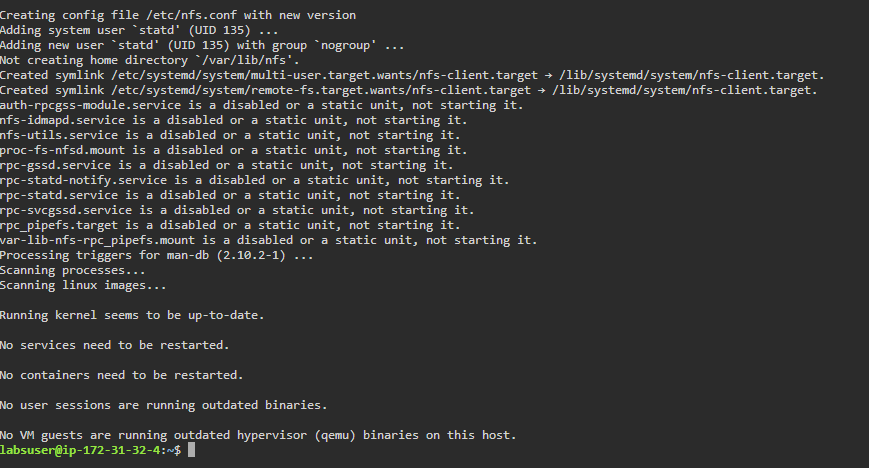
**Deploy NFS Client on Worker Nodes:**

sudo apt install nfs-common

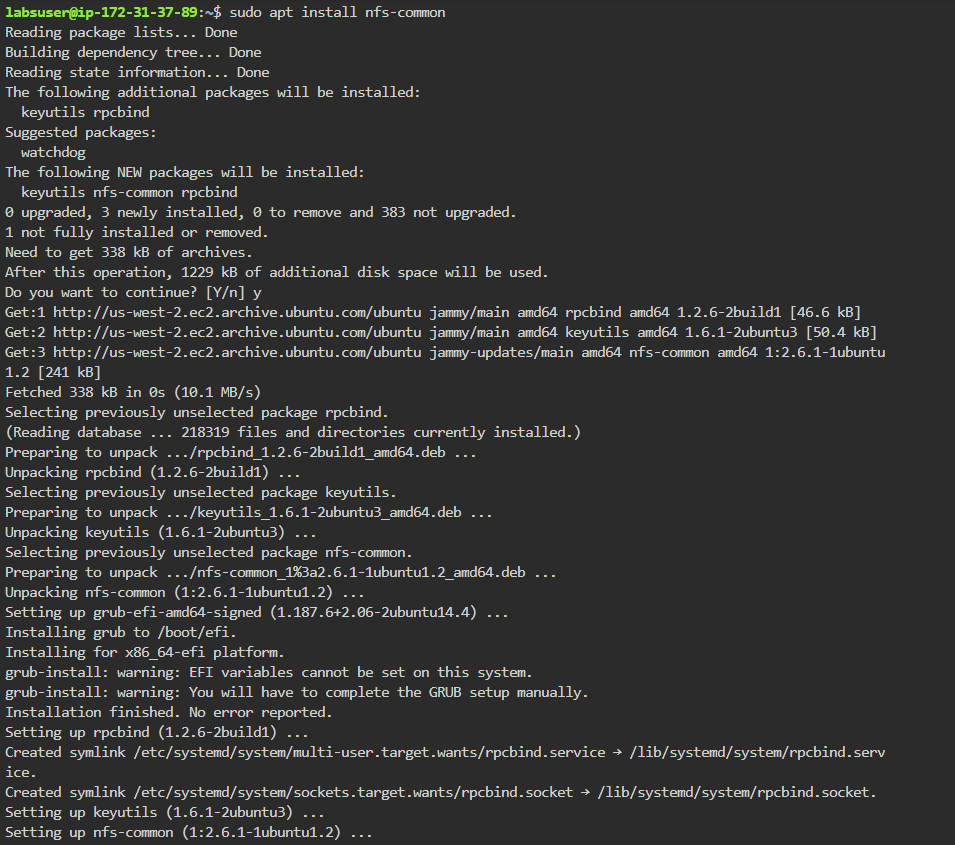
on worker1:

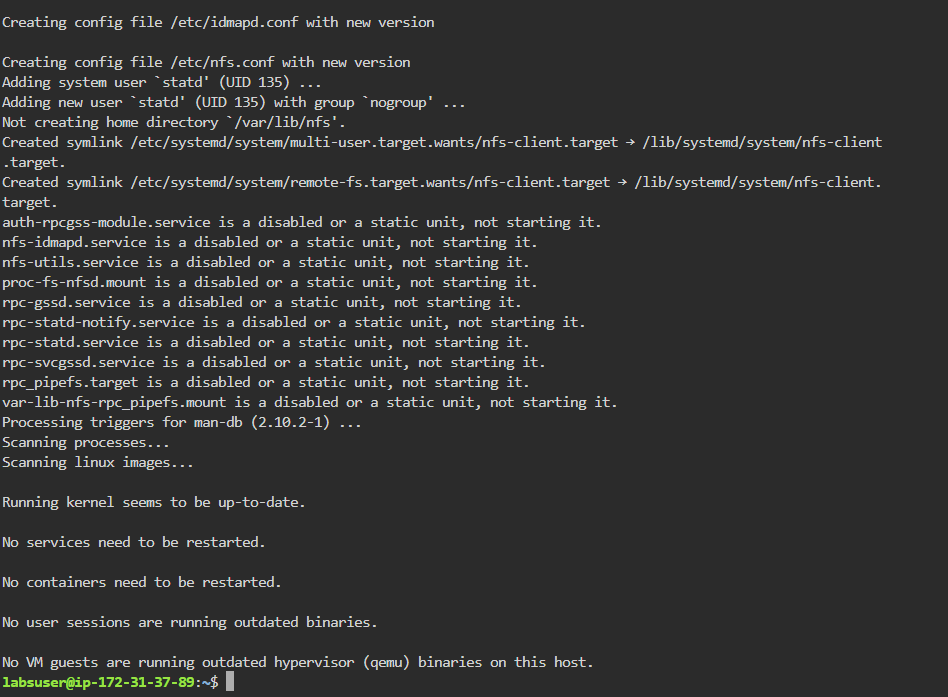






Worker 2:





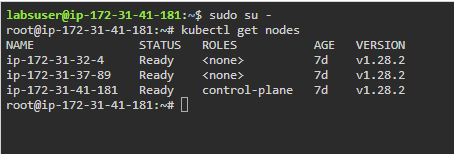
**1. Cleanup Existing Resources**

**Commands to Cleanup Resources**

bash

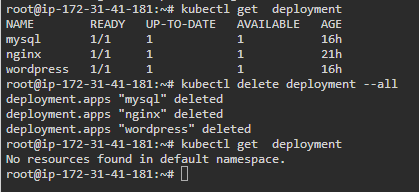
# List all nodes and ensure they are in ready status

kubectl get nodes



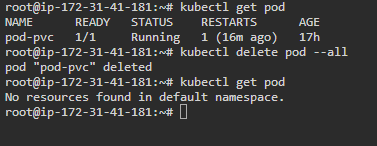
# Delete all deployments

kubectl delete deployment –all



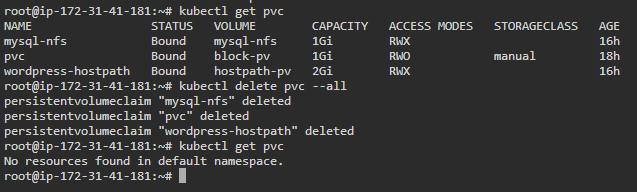
# Delete all pods

kubectl delete pod –all



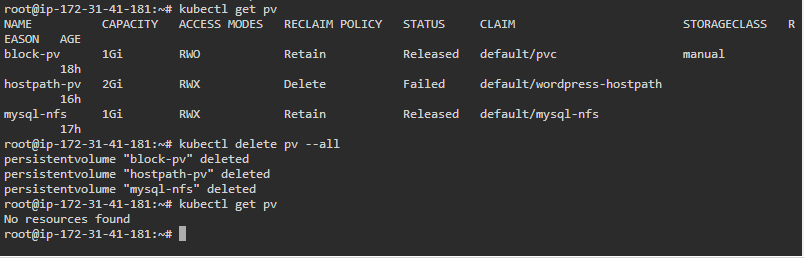
# Delete all persistent volume claims (PVCs)

kubectl delete pvc –all



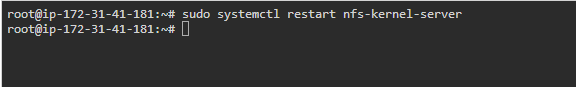
# Delete all persistent volumes (PVs)

kubectl delete pv –all



**Restart NFS Server**

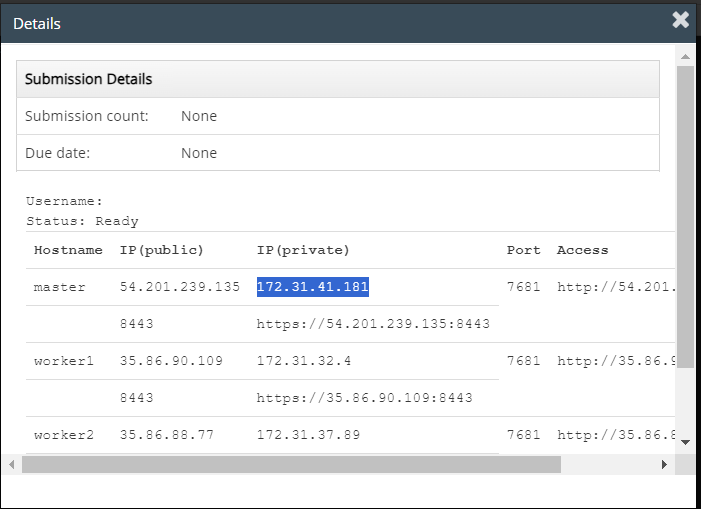
sudo systemctl restart nfs-kernel-server



**2. Persistent Volume Setup**

**Create Persistent Volume**

**Picking up private master ip**

****

<privateipaddressofMasternode>172.31.41.181

**File:** pv-mysql.yml

yaml

apiVersion: v1

kind: PersistentVolume

metadata:

name: mysql-nfs

spec:

capacity:

storage: 1Gi

accessModes:

- ReadWriteMany

mountOptions:

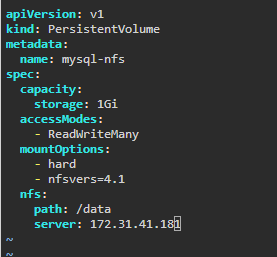
- hard

- nfsvers=4.1

nfs:

path: /data

server: <privateipaddressofMasternode>

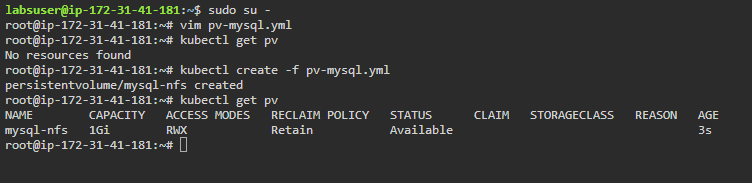


# Create Persistent Volume

kubectl create -f pv-mysql.yml

# Verify the Persistent Volume

kubectl get pv



**Create Persistent Volume Claim**

**File:** pvc-mysql.yml

yaml

kind: PersistentVolumeClaim

apiVersion: v1

metadata:

name: mysql-nfs

spec:

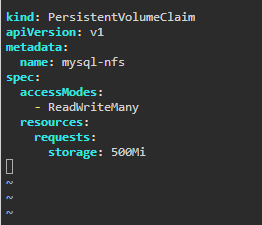
accessModes:

- ReadWriteMany

resources:

requests:

storage: 500Mi

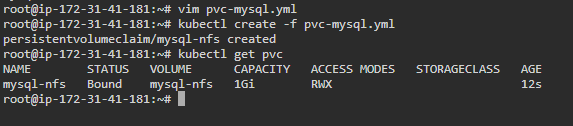


# Create Persistent Volume Claim

kubectl create -f pvc-mysql.yml

# Verify the Persistent Volume Claim

kubectl get pvc

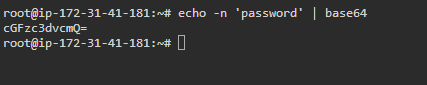


**3. Secret Management**

**Create a Secret**

**File:** mysqlsecret.yml

Encrypted password:



yaml

kind: Secret

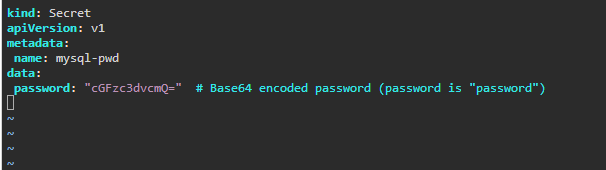
apiVersion: v1

metadata:

name: mysql-pwd

data:

password: "cGFzc3dvcmQ=" # Base64 encoded password (password is "password")

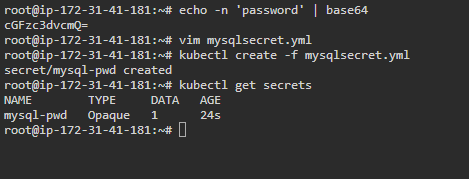


# Create the Secret

kubectl create -f mysqlsecret.yml

# Verify the Secret

kubectl get secrets



**4. ConfigMap Creation**

**Create a ConfigMap**

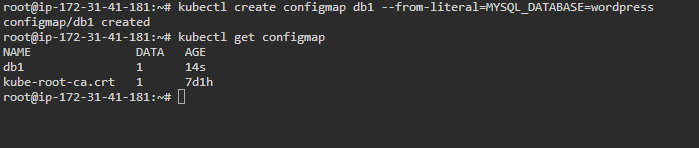
bash

# Create ConfigMap from literal value

kubectl create configmap db1 --from-literal=MYSQL\_DATABASE=wordpress

# Verify the ConfigMap

kubectl get configmap



**5. Deploy MySQL**

**Create MySQL Deployment**

**File:** mysql-deployment.yml

yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: mysql

spec:

replicas: 1

selector:

matchLabels:

app: mysql-wordpress

template:

metadata:

labels:

app: mysql-wordpress

product: mysql

spec:

containers:

- name: mysql-container

image: mysql

env:

- name: MYSQL\_ROOT\_PASSWORD

valueFrom:

secretKeyRef:

name: mysql-pwd

key: password

- name: MYSQL\_DATABASE

valueFrom:

configMapKeyRef:

name: db1

key: MYSQL\_DATABASE

volumeMounts:

- name: mysql-storage

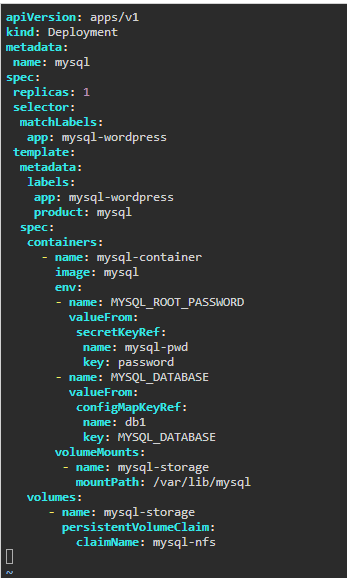
mountPath: /var/lib/mysql

volumes:

- name: mysql-storage

persistentVolumeClaim:

claimName: mysql-nfs



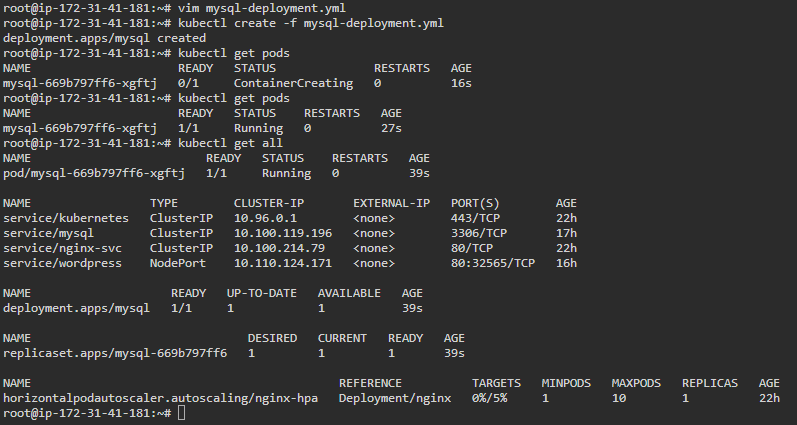
# Create MySQL Deployment

kubectl create -f mysql-deployment.yml

# Verify the Pods and all resources

kubectl get pods

kubectl get all



**Create MySQL Service**

**File:** service-mysql.yml

yaml

apiVersion: v1

kind: Service

metadata:

name: mysql

spec:

type: ClusterIP

ports:

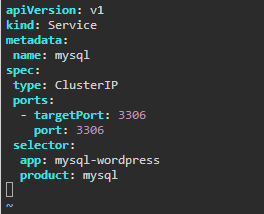
- targetPort: 3306

port: 3306

selector:

app: mysql-wordpress

product: mysql

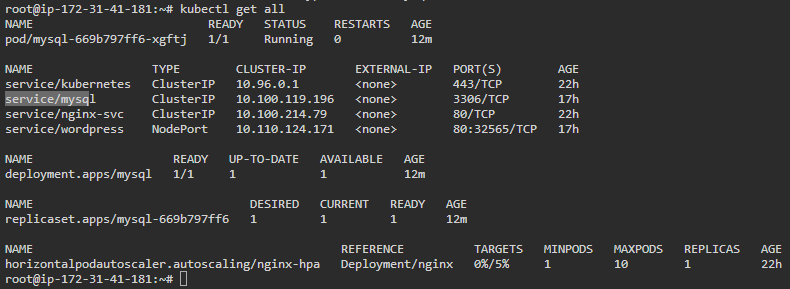


# Create MySQL Service

kubectl create -f service-mysql.yml

# Verify all resources

kubectl get all



**6. Deploy WordPress**

**Create WordPress Deployment and Service**

**File:** wordpress-deployment.yml

yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: wordpress

spec:

replicas: 1

selector:

matchLabels:

app: mysql-wordpress

tier: frontend

template:

metadata:

labels:

app: mysql-wordpress

tier: frontend

spec:

containers:

- name: wordpress-container

image: wordpress

env:

- name: WORDPRESS\_DB\_HOST

value: mysql

- name: WORDPRESS\_DB\_USER

value: root

- name: WORDPRESS\_DB\_PASSWORD

valueFrom:

secretKeyRef:

name: mysql-pwd

key: password

---

apiVersion: v1

kind: Service

metadata:

name: wordpress

spec:

type: NodePort

ports:

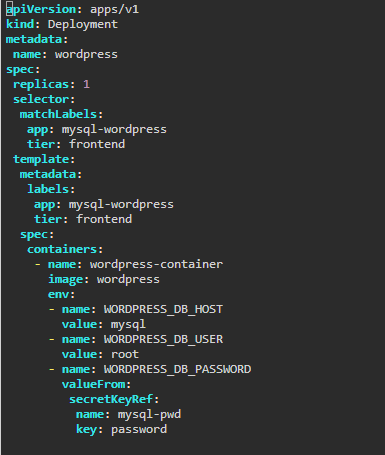
- targetPort: 80

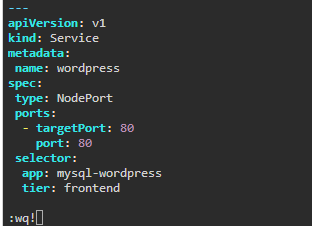
port: 80

selector:

app: mysql-wordpress

tier: frontend



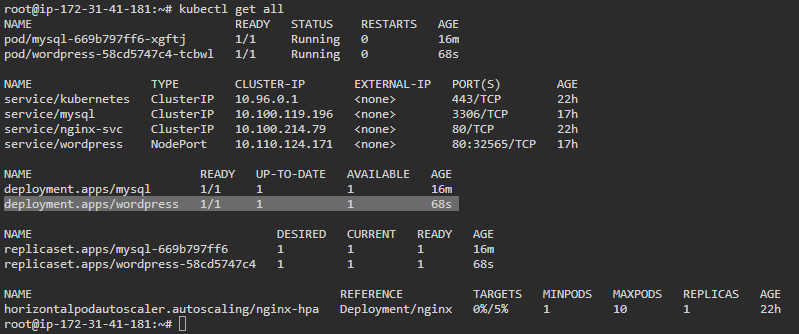


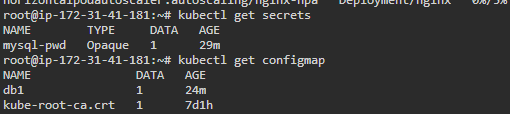
# Create WordPress Deployment and Service

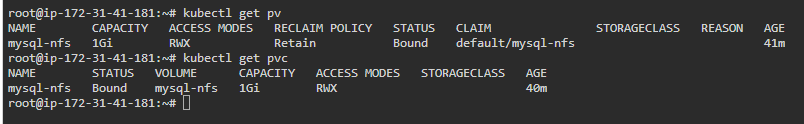
kubectl create -f wordpress-deployment.yml

# Verify all resources

kubectl get all

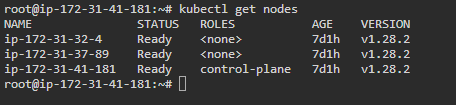






**7. Kubernetes Dashboard Setup**

**Checking Worker joined the cluster:**



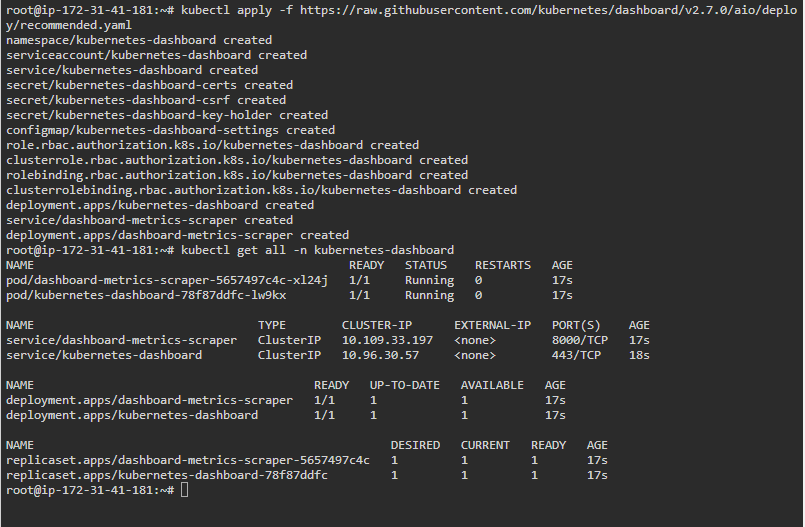
**Deploy Kubernetes Dashboard**

# Deploy Kubernetes Dashboard

kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yaml

# Verify the Dashboard deployment

kubectl get all -n kubernetes-dashboard

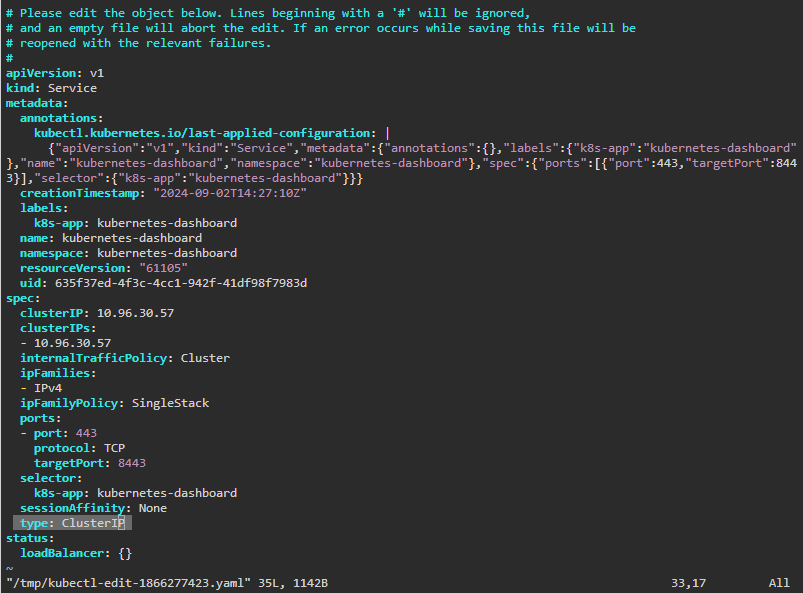


**Expose Dashboard via NodePort**

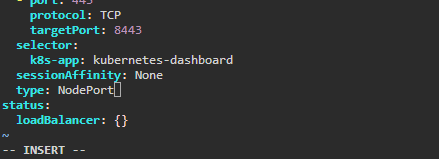
# Edit the service to change type to NodePort

kubectl edit svc -n kubernetes-dashboard kubernetes-dashboard

# Change type from ClusterIP to NodePort, save the file

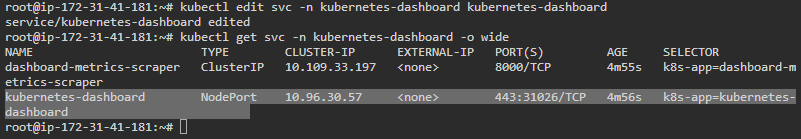


Changed to be NodePort



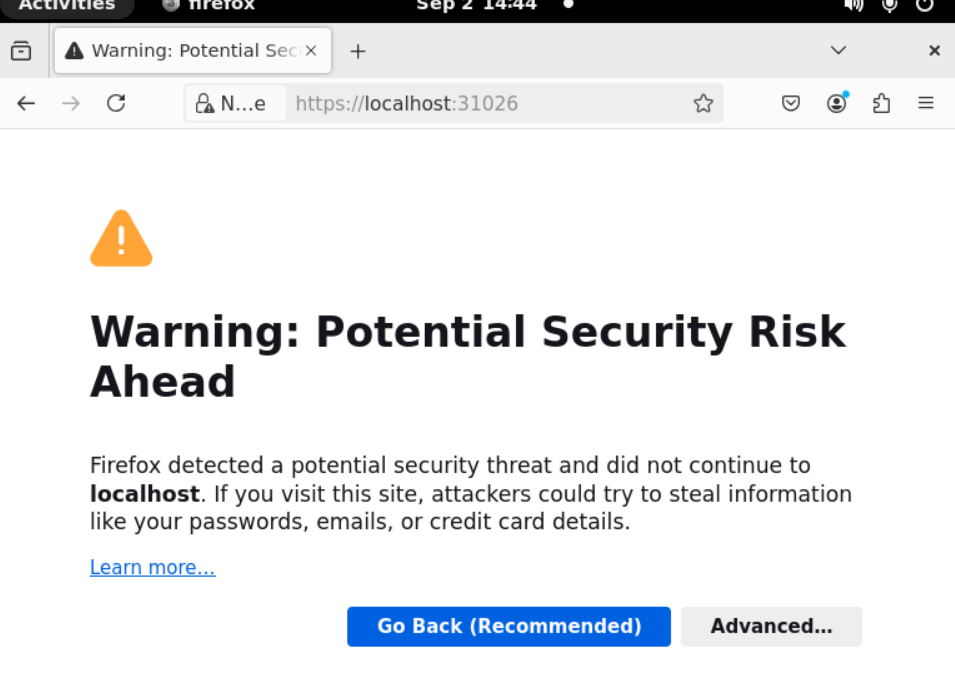
# Verify the NodePort

kubectl get svc -n kubernetes-dashboard -o wide

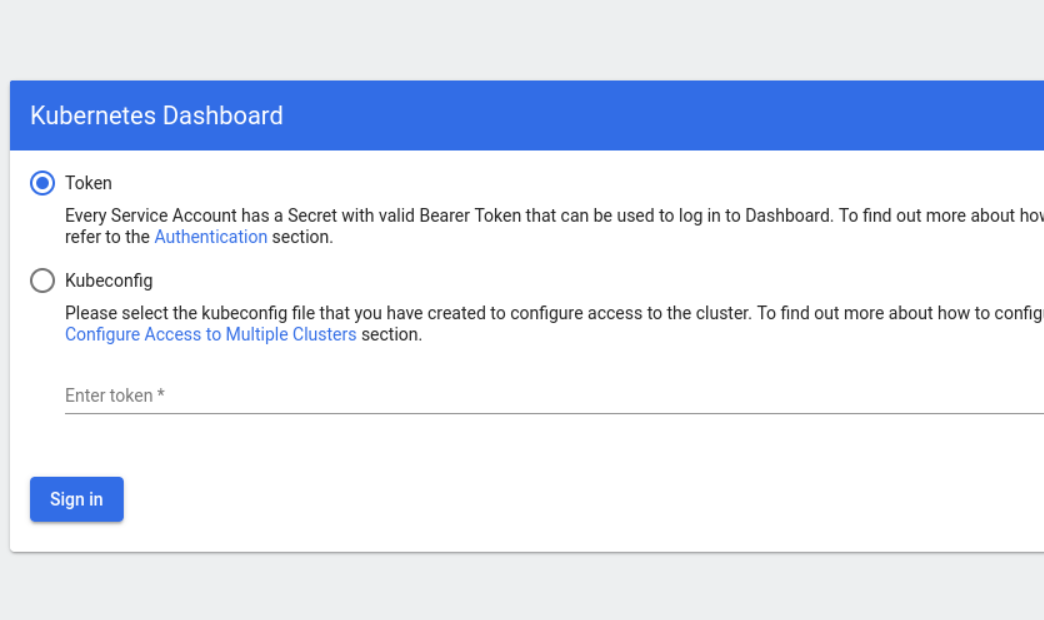


**Access Dashboard**

1. Open Firefox or any browser.
2. Go to https://localhost:<NodePort> (Replace <NodePort> with the port number obtained).



1. Click on "Advanced" -> "Accept Risk and Continue".



**8. Service Account and Token for Dashboard Access**

**Create Service Account**

**File:** serviceaccount.yml

yaml

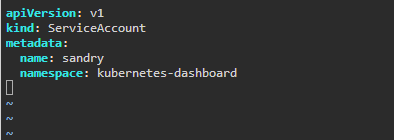
apiVersion: v1

kind: ServiceAccount

metadata:

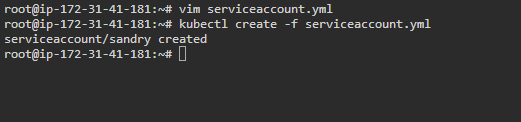
name: sandry

namespace: kubernetes-dashboard



# Create Service Account

kubectl create -f serviceaccount.yml



**Create ClusterRoleBinding**

**File:** clusterrole-sa.yml

yaml

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding

metadata:

name: admin-user

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

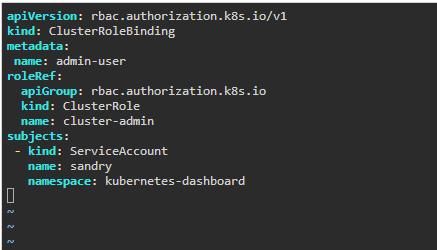
name: cluster-admin

subjects:

- kind: ServiceAccount

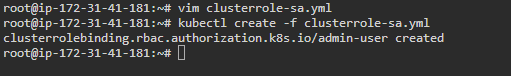
name: sandry

namespace: kubernetes-dashboard



# Create ClusterRoleBinding

kubectl create -f clusterrole-sa.yml



**Create Secret for Service Account**

**File:** secret-sa.yml

yaml

apiVersion: v1

kind: Secret

type: kubernetes.io/service-account-token

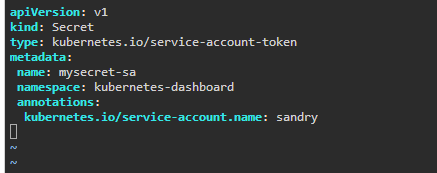
metadata:

name: mysecret-sa

namespace: kubernetes-dashboard

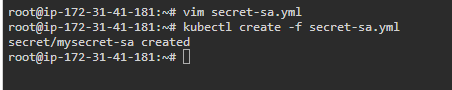
annotations:

kubernetes.io/service-account.name: sandry



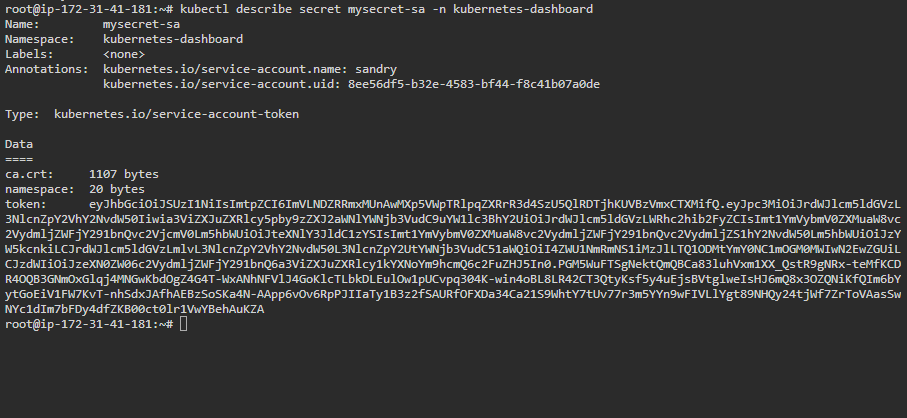
# Create Secret for Service Account

kubectl create -f secret-sa.yml

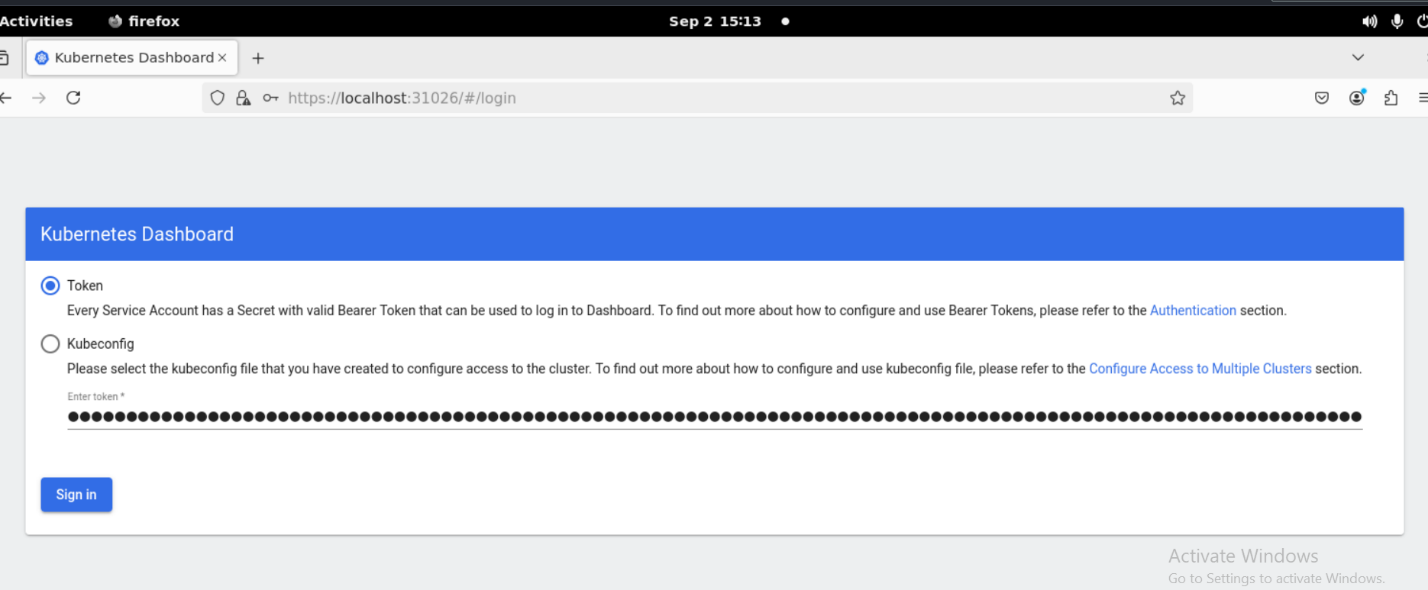


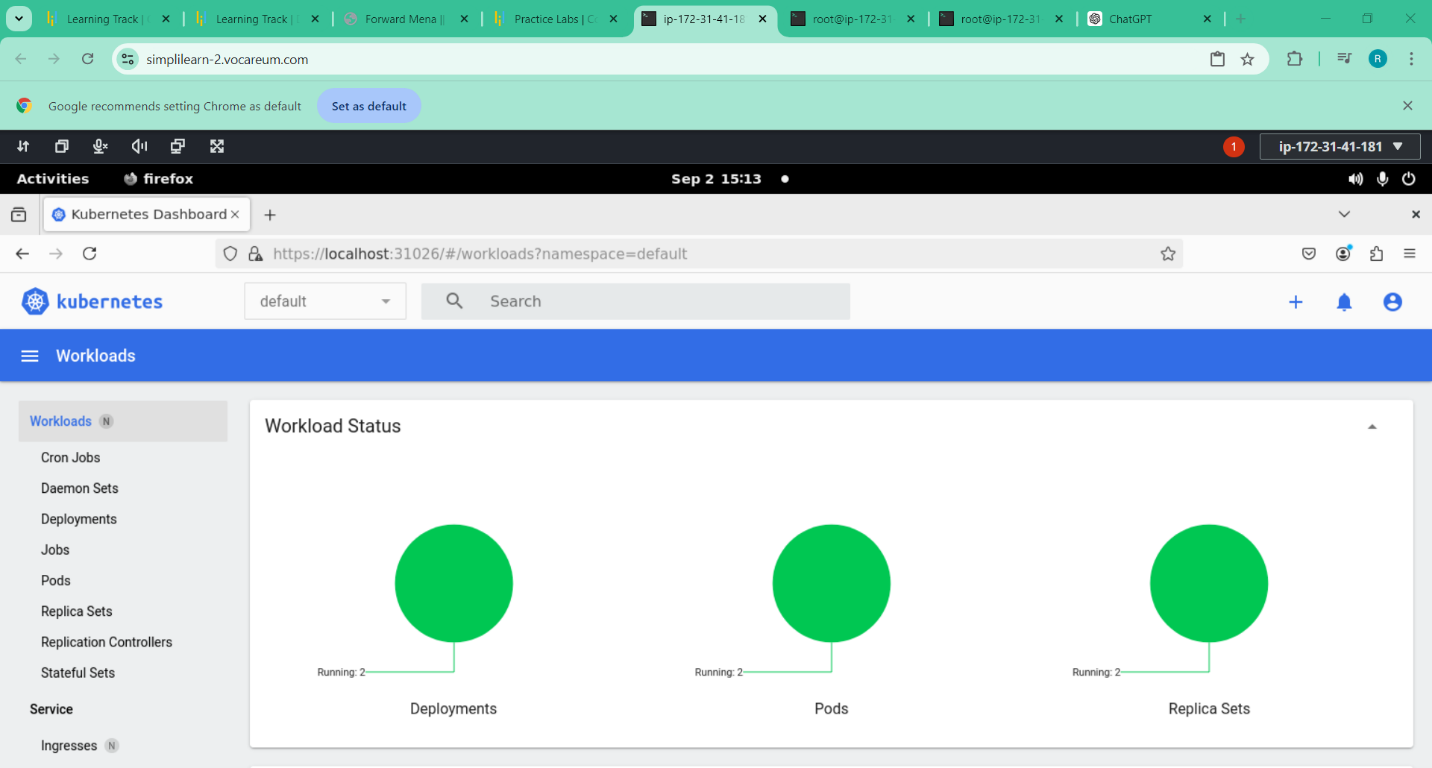
# Describe Secret to get the token

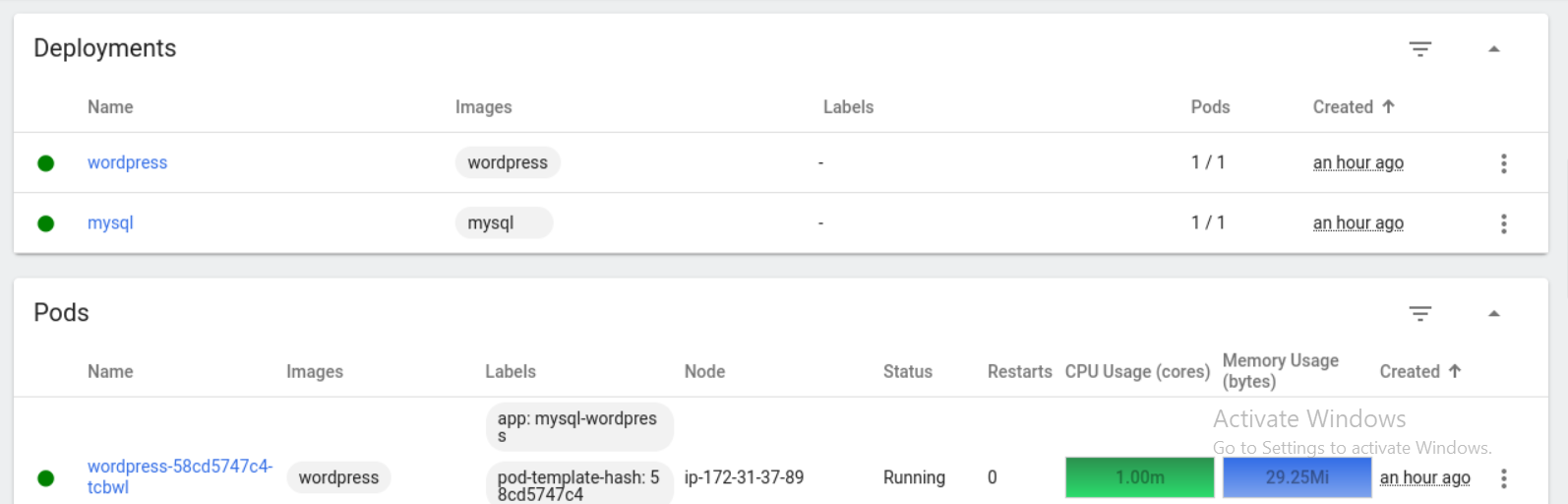
kubectl describe secret mysecret-sa -n kubernetes-dashboard

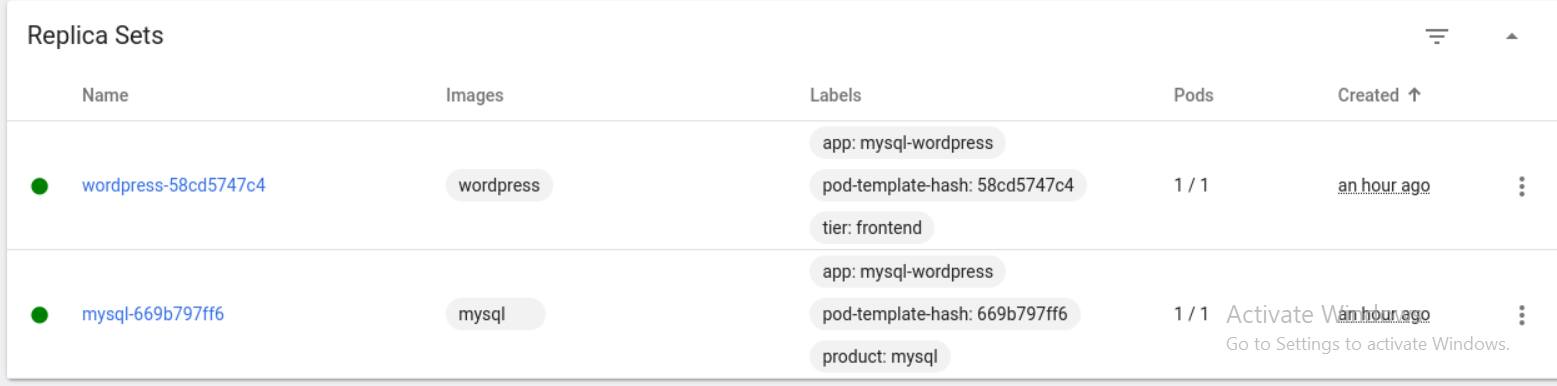


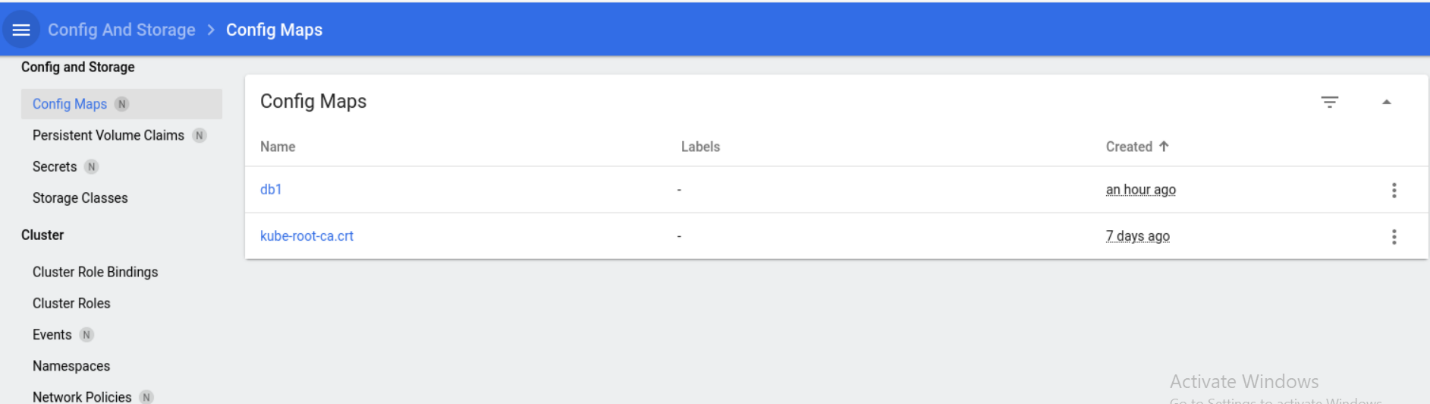
Copy the token and use it to log into the Kubernetes Dashboard.

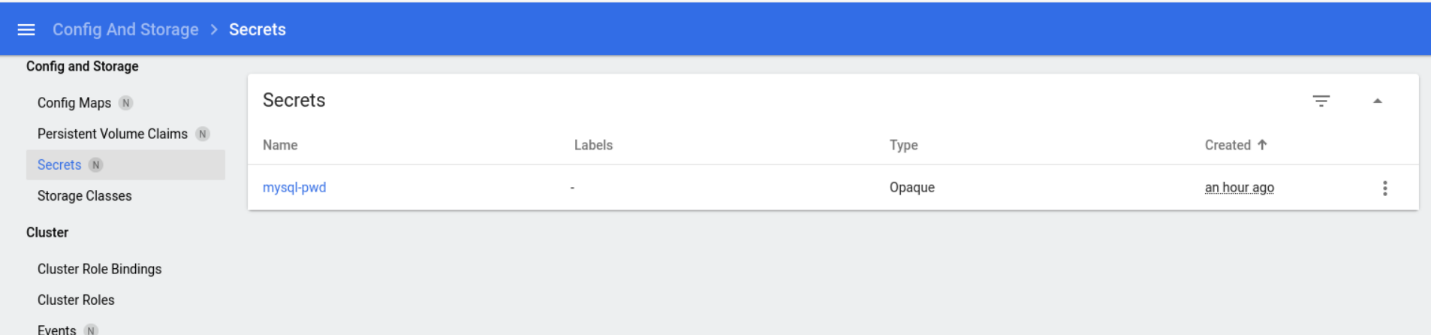


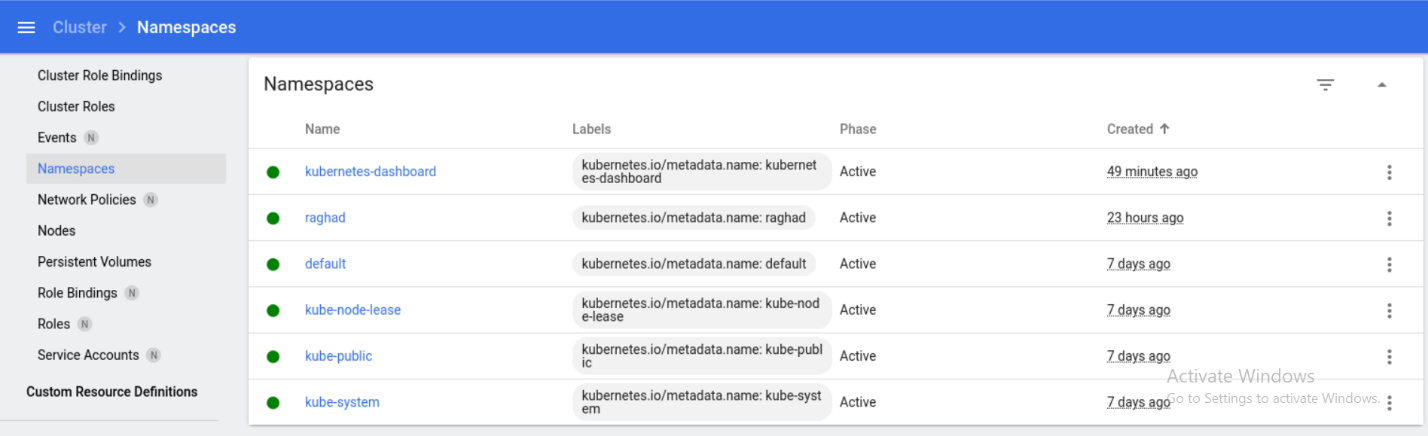


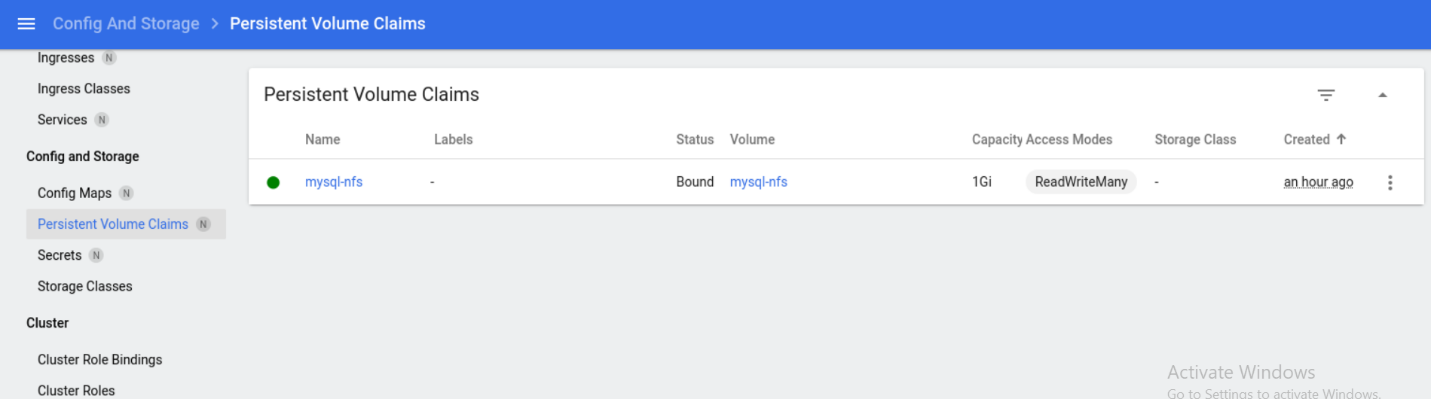


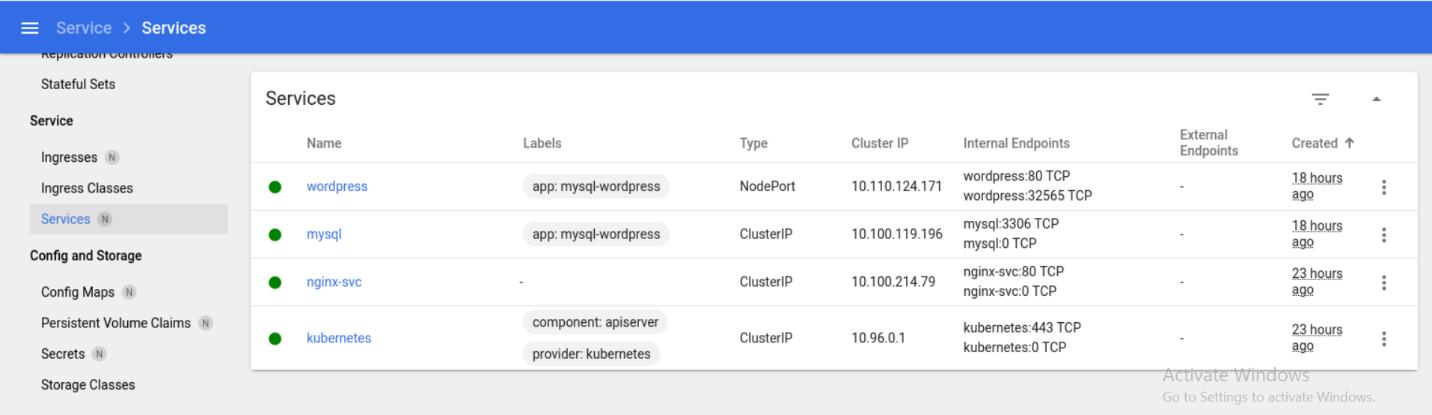


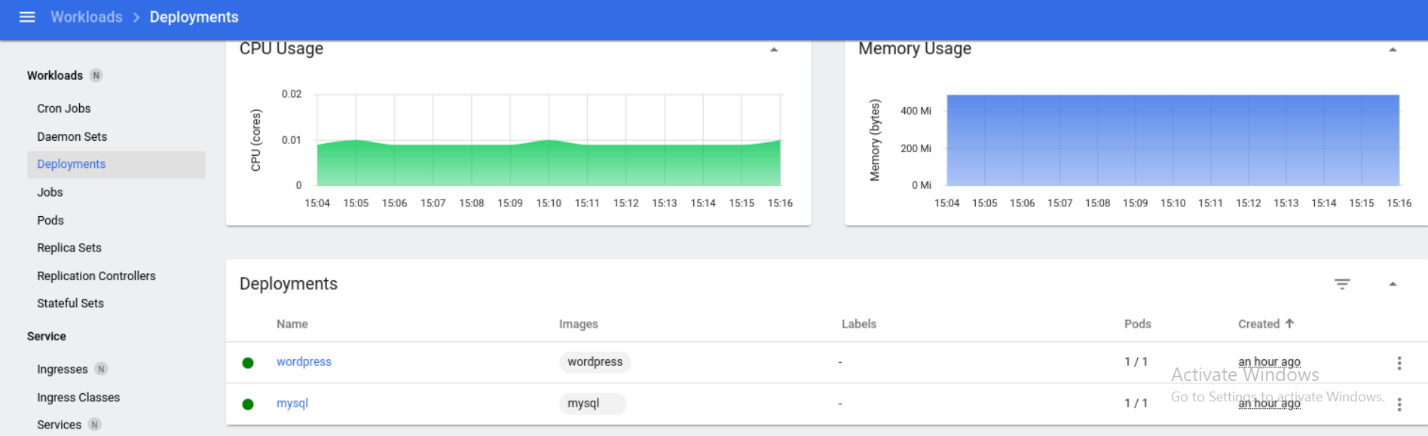












**Kubernetes Resources:**

* + kubectl get pv
  + kubectl get pvc
  + kubectl get secrets
  + kubectl get configmap
  + kubectl get all

