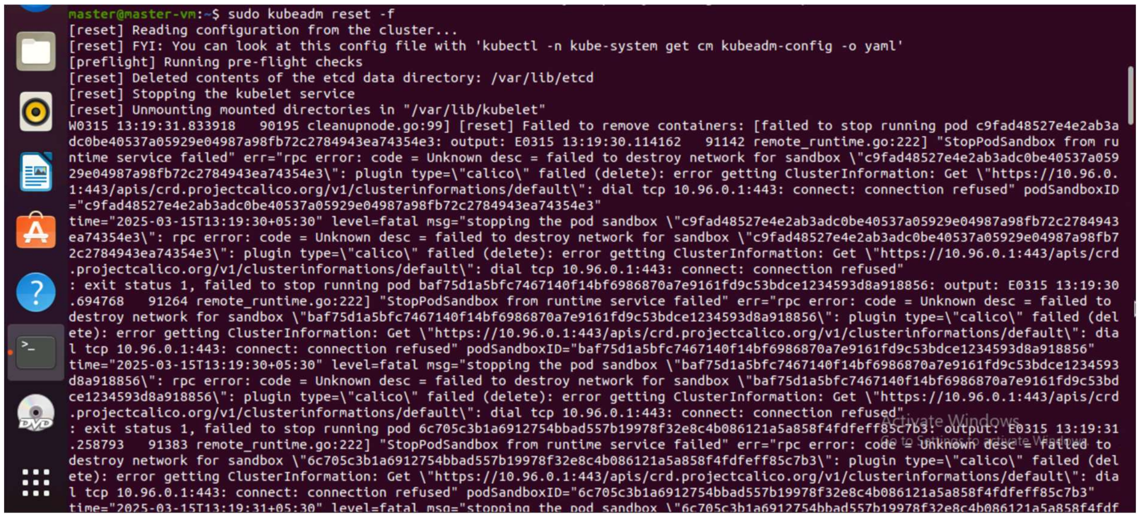
Kubernetes Project-02

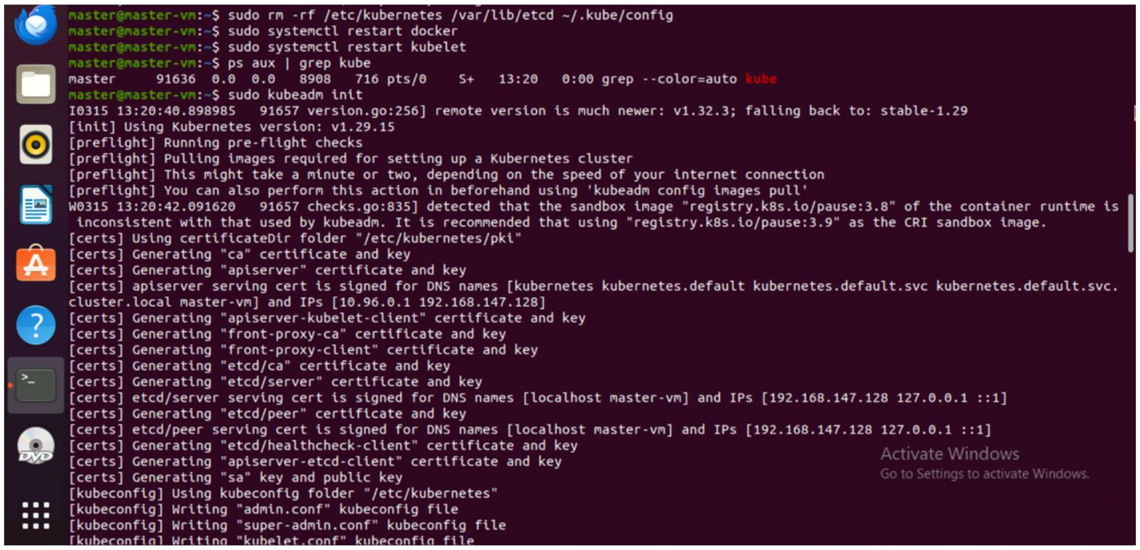
Kubernetes Multi-Tenant Project

Step 1: Check if Any Worker Node is Ready

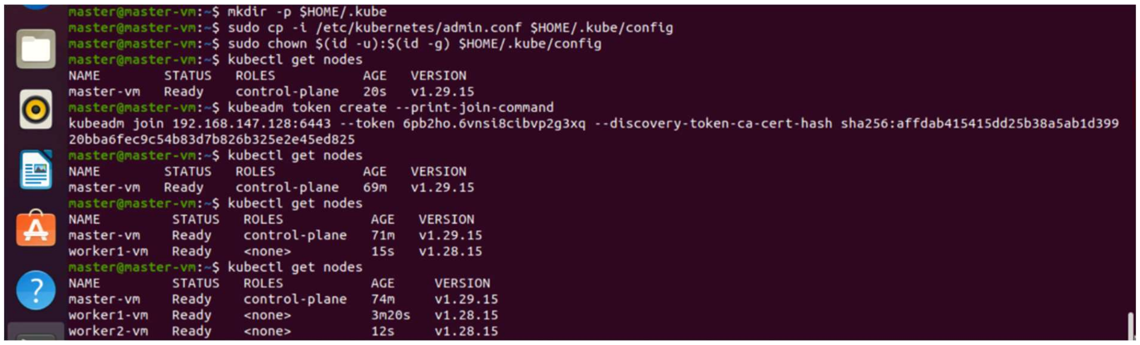
kubectl get nodes



```
master@master-vm:~$ sudo kubeadm reset -f
[reset] Reading configuration from the cluster...
[reset] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[preflight] Running pre-flight checks
[reset] Deleted contents of the etcd data directory: /var/lib/etcd
[reset] Stopping the kubelet service
[reset] Unmounting mounted directories in "/var/lib/kubelet"
W0315 13:19:31.833918   90195 cleanupnode.go:99] [reset] Failed to remove containers: [failed to stop running pod c9fad48527e4e2ab3a
dc0be40537a05929e04987a98fb72c2784943ea74354e3: output: E0315 13:19:30.114162   91142 remote_runtime.go:222] "StopPodSandbox from ru
ntime service failed" err="rpc error: code = Unknown desc = failed to destroy network for sandbox \"c9fad48527e4e2ab3adc0be40537a059
29e04987a98fb72c2784943ea74354e3\": plugin type=\"calico\" failed (delete): error getting ClusterInformation: Get \"https://10.96.0.
1:443/apis/crd.projectcalico.org/v1/clusterinformations/default\": dial tcp 10.96.0.1:443: connect: connection refused" podSandboxID
="c9fad48527e4e2ab3adc0be40537a05929e04987a98fb72c2784943ea74354e3"
time="2025-03-15T13:19:30+05:30" level=fatal msg="stopping the pod sandbox \"c9fad48527e4e2ab3adc0be40537a05929e04987a98fb72c2784943
ea74354e3\": rpc error: code = Unknown desc = failed to destroy network for sandbox \"c9fad48527e4e2ab3adc0be40537a05929e04987a98fb7
2c2784943ea74354e3\": plugin type=\"calico\" failed (delete): error getting ClusterInformation: Get \"https://10.96.0.1:443/apis/crd
.projectcalico.org/v1/clusterinformations/default\": dial tcp 10.96.0.1:443: connect: connection refused"
: exit status 1, failed to stop running pod baf75d1a5bfc7467140f14bf6986870a7e9161fd9c53bdce1234593d8a918856: output: E0315 13:19:30
.694768   91264 remote_runtime.go:222] "StopPodSandbox from runtime service failed" err="rpc error: code = Unknown desc = failed to
destroy network for sandbox \"baf75d1a5bfc7467140f14bf6986870a7e9161fd9c53bdce1234593d8a918856\": plugin type=\"calico\" failed (del
ete): error getting ClusterInformation: Get \"https://10.96.0.1:443/apis/crd.projectcalico.org/v1/clusterinformations/default\": dia
l tcp 10.96.0.1:443: connect: connection refused" podSandboxID="baf75d1a5bfc7467140f14bf6986870a7e9161fd9c53bdce1234593d8a918856"
time="2025-03-15T13:19:30+05:30" level=fatal msg="stopping the pod sandbox \"baf75d1a5bfc7467140f14bf6986870a7e9161fd9c53bdce1234593
d8a918856\": rpc error: code = Unknown desc = failed to destroy network for sandbox \"baf75d1a5bfc7467140f14bf6986870a7e9161fd9c53bd
ce1234593d8a918856\": plugin type=\"calico\" failed (delete): error getting ClusterInformation: Get \"https://10.96.0.1:443/apis/crd
.projectcalico.org/v1/clusterinformations/default\": dial tcp 10.96.0.1:443: connect: connection refused"
: exit status 1, failed to stop running pod 6c705c3b1a6912754bbad557b19978f32e8c4b086121a5a858f4fdfeff85c7b3: output: E0315 13:19:31
.258793   91383 remote_runtime.go:222] "StopPodSandbox from runtime service failed" err="rpc error: code = Unknown desc = failed to
destroy network for sandbox \"6c705c3b1a6912754bbad557b19978f32e8c4b086121a5a858f4fdfeff85c7b3\": plugin type=\"calico\" failed (del
ete): error getting ClusterInformation: Get \"https://10.96.0.1:443/apis/crd.projectcalico.org/v1/clusterinformations/default\": dia
l tcp 10.96.0.1:443: connect: connection refused" podSandboxID="6c705c3b1a6912754bbad557b19978f32e8c4b086121a5a858f4fdfeff85c7b3"
time="2025-03-15T13:19:31+05:30" level=fatal msg="stopping the pod sandbox \"6c705c3b1a6912754bbad557b19978f32e8c4b086121a5a858f4fdf
```



```
master@master-vm:~$ sudo rm -rf /etc/kubernetes /var/lib/etcd ~/.kube/config
master@master-vm:~$ sudo systemctl restart docker
master@master-vm:~$ sudo systemctl restart kubelet
master@master-vm:~$ ps aux | grep kube
master     91636  0.0  0.0   8908   716 pts/0    S+   13:20   0:00 grep --color=auto kube
master@master-vm:~$ sudo kubeadm init
I0315 13:20:40.898985   91657 version.go:256] remote version is much newer: v1.32.3; falling back to: stable-1.29
[init] Using Kubernetes version: v1.29.15
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
W0315 13:20:42.091620   91657 checks.go:835] detected that the sandbox image "registry.k8s.io/pause:3.8" of the container runtime is
 inconsistent with that used by kubeadm. It is recommended that using "registry.k8s.io/pause:3.9" as the CRI sandbox image.
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.
cluster.local master-vm] and IPs [10.96.0.1 192.168.147.128]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [localhost master-vm] and IPs [192.168.147.128 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [localhost master-vm] and IPs [192.168.147.128 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "super-admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
```

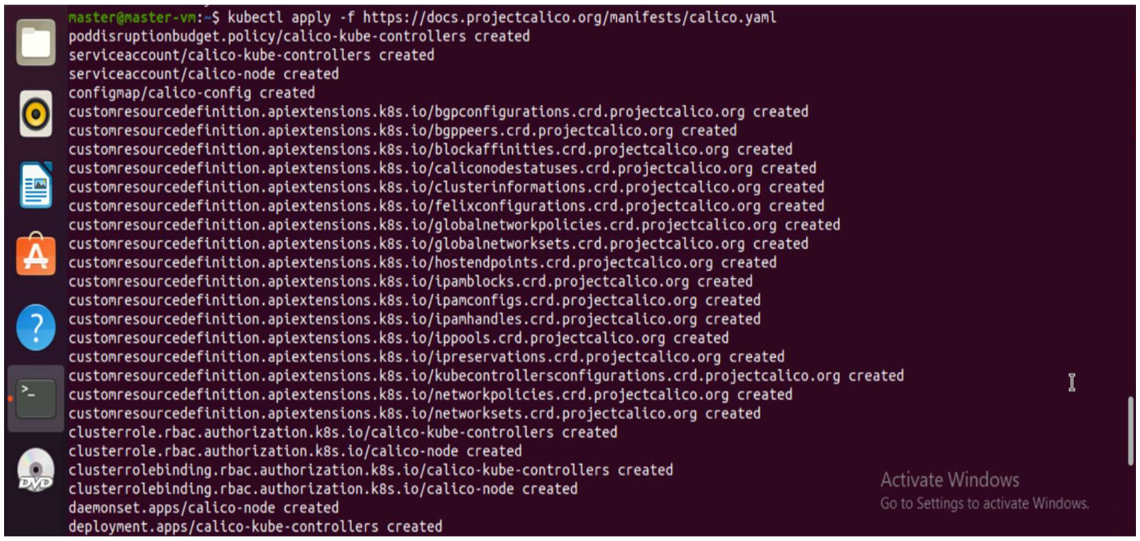


```
master@master-vm:~$ mkdir -p $HOME/.kube
master@master-vm:~$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
master@master-vm:~$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
master@master-vm:~$ kubectl get nodes
NAME        STATUS   ROLES           AGE   VERSION
master-vm   Ready    control-plane   20s   v1.29.15
master@master-vm:~$ kubeadm token create --print-join-command
kubeadm join 192.168.147.128:6443 --token 6pb2ho.6vnsi8cibvp2g3xq --discovery-token-ca-cert-hash sha256:affdab415415dd25b38a5ab1d399
20bba6fec9c54b83d7b826b325e2e45ed825
master@master-vm:~$ kubectl get nodes
NAME        STATUS   ROLES           AGE   VERSION
master-vm   Ready    control-plane   69m   v1.29.15
master@master-vm:~$ kubectl get nodes
NAME         STATUS   ROLES           AGE   VERSION
master-vm    Ready    control-plane   71m   v1.29.15
worker1-vm   Ready    <none>          15s   v1.28.15
master@master-vm:~$ kubectl get nodes
NAME         STATUS   ROLES           AGE     VERSION
master-vm    Ready    control-plane   74m     v1.29.15
worker1-vm   Ready    <none>          3m20s   v1.28.15
worker2-vm   Ready    <none>          12s     v1.28.15
```

Step 2: Install Calico for Networking

Apply the Calico manifest to enable networking:

kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml



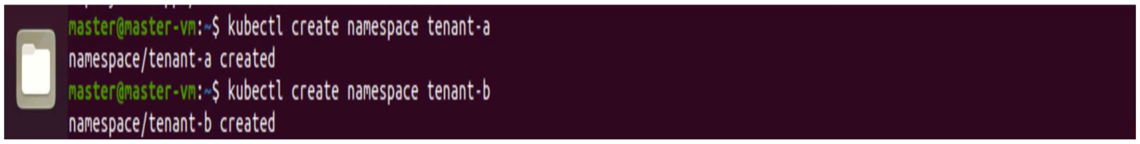
```
master@master-vm:~$ kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
poddisruptionbudget.policy/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
serviceaccount/calico-node created
configmap/calico-config created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/blockaffinities.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/caliconodestatuses.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipreservations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/kubecontrollersconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org created
clusterrole.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
daemonset.apps/calico-node created
deployment.apps/calico-kube-controllers created
```

Step 3: Create Namespaces for Tenants

To isolate tenants, create separate namespaces:

kubectl create namespace tenant-a

kubectl create namespace tenant-b



```
master@master-vm:~$ kubectl create namespace tenant-a
namespace/tenant-a created
master@master-vm:~$ kubectl create namespace tenant-b
namespace/tenant-b created
```

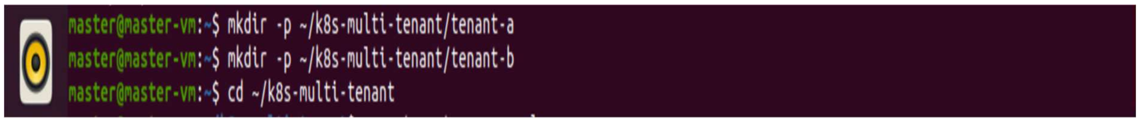
Step 4: Create Folder Structure for YAML Files

Create the folder structure to organize YAML files for each tenant:

mkdir -p ~/k8s-multi-tenant/tenant-a

mkdir -p ~/k8s-multi-tenant/tenant-b

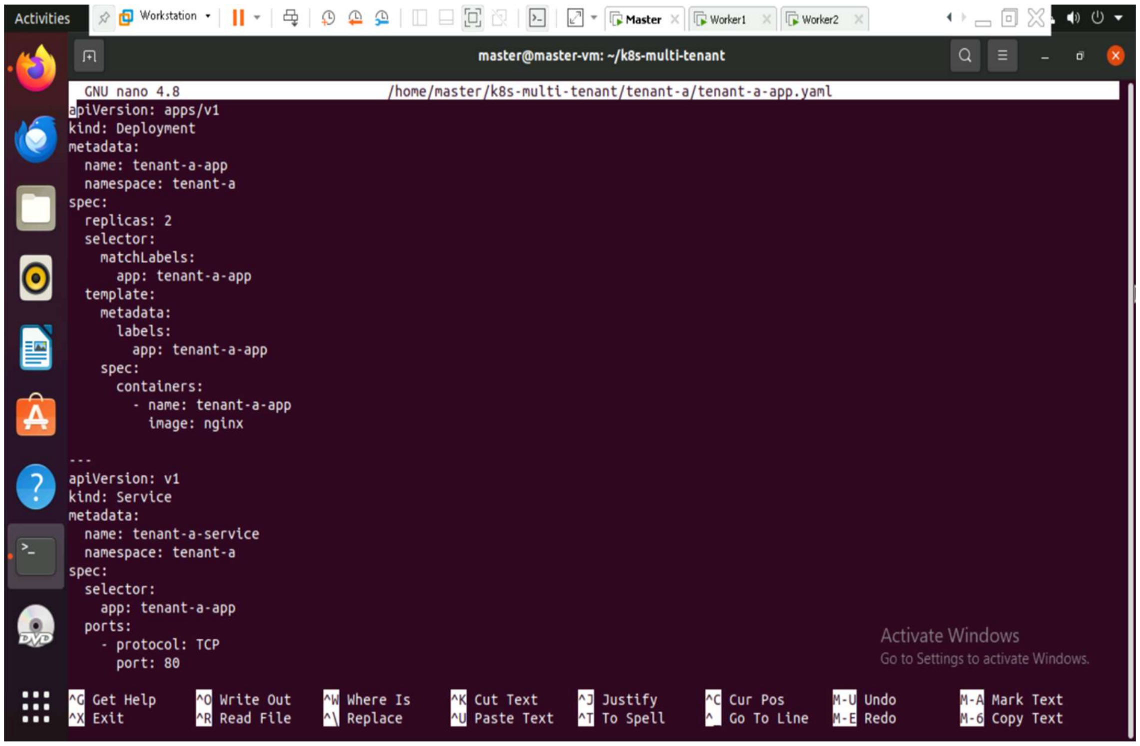
cd ~/k8s-multi-tenant



```
master@master-vm:~$ mkdir -p ~/k8s-multi-tenant/tenant-a
master@master-vm:~$ mkdir -p ~/k8s-multi-tenant/tenant-b
master@master-vm:~$ cd ~/k8s-multi-tenant
```

Step 5: Create Deployment and Service for Tenant A

nano ~/k8s-multi-tenant/tenant-a/tenant-a-app.yaml


Activities
Workstation
Master
Worker1
Worker2
master@master-vm: ~/k8s-multi-tenant
Activate Windows
Go to Settings to activate Windows.

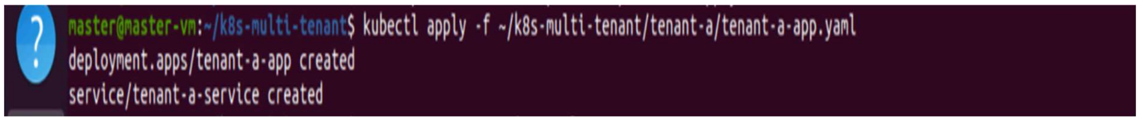
```
  GNU nano 4.8                      /home/master/k8s-multi-tenant/tenant-a/tenant-a-app.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tenant-a-app
  namespace: tenant-a
spec:
  replicas: 2
  selector:
    matchLabels:
      app: tenant-a-app
  template:
    metadata:
      labels:
        app: tenant-a-app
    spec:
      containers:
        - name: tenant-a-app
          image: nginx

---
apiVersion: v1
kind: Service
metadata:
  name: tenant-a-service
  namespace: tenant-a
spec:
  selector:
    app: tenant-a-app
  ports:
    - protocol: TCP
      port: 80

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo       M-A Mark Text
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E Redo       M-6 Copy Text
```

Apply the configuration:

kubectl apply -f ~/k8s-multi-tenant/tenant-a/tenant-a-app.yaml

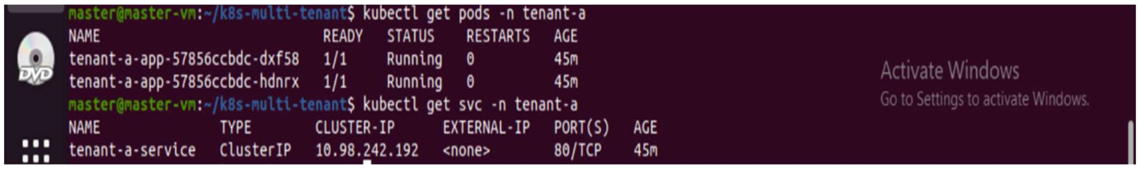


```
master@master-vm:~/k8s-multi-tenant$ kubectl apply -f ~/k8s-multi-tenant/tenant-a/tenant-a-app.yaml
deployment.apps/tenant-a-app created
service/tenant-a-service created
```

Verify the deployment:

kubectl get pods -n tenant-a

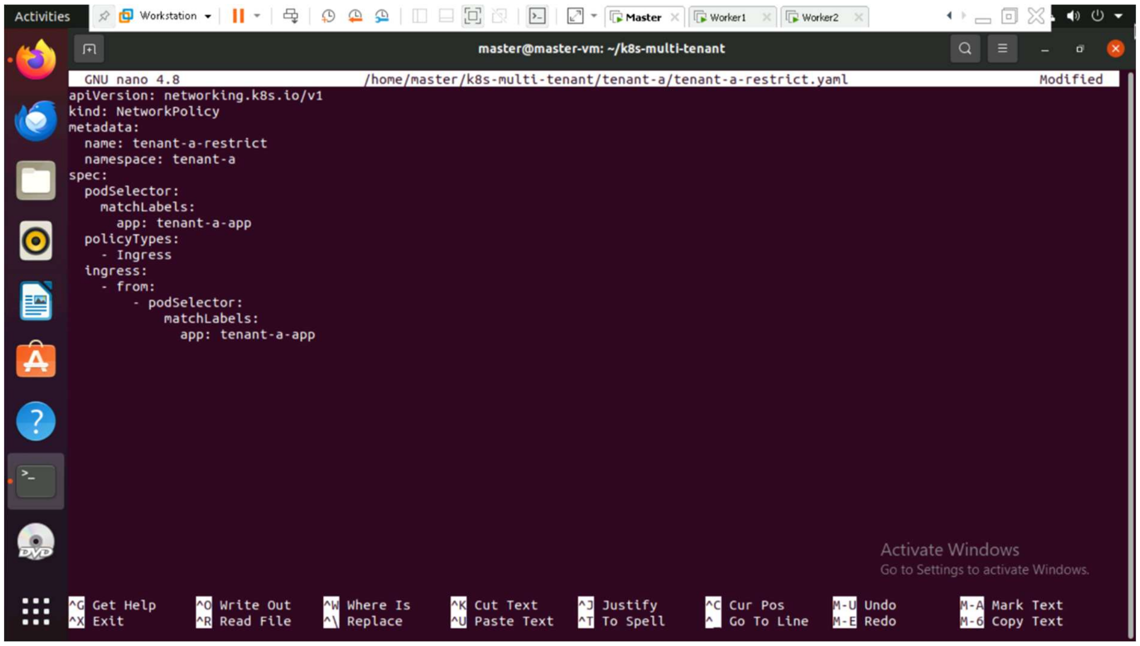
kubectl get svc -n tenant-a



```
master@master-vm:~/k8s-multi-tenant$ kubectl get pods -n tenant-a
NAME                            READY   STATUS    RESTARTS   AGE
tenant-a-app-57856ccbdc-dxf58   1/1     Running   0          45m
tenant-a-app-57856ccbdc-hdnrx   1/1     Running   0          45m
master@master-vm:~/k8s-multi-tenant$ kubectl get svc -n tenant-a
NAME               TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
tenant-a-service   ClusterIP   10.98.242.192   <none>        80/TCP    45m
```

Step 6: Restrict Network Access for Tenant A

nano ~/k8s-multi-tenant/tenant-a/tenant-a-restrict.yaml

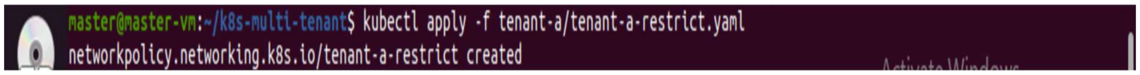


```
GNU nano 4.8          /home/master/k8s-multi-tenant/tenant-a/tenant-a-restrict.yaml          Modified
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: tenant-a-restrict
  namespace: tenant-a
spec:
  podSelector:
    matchLabels:
      app: tenant-a-app
  policyTypes:
    - Ingress
  ingress:
    - from:
        - podSelector:
            matchLabels:
              app: tenant-a-app
```

Activities
Workstation
Master
Worker1
Worker2
master@master-vm: ~/k8s-multi-tenant
Activate Windows
Go to Settings to activate Windows.
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos M-U Undo M-A Mark Text
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line M-E Redo M-6 Copy Text

Apply the network policy:

kubectl apply -f ~/k8s-multi-tenant/tenant-a/tenant-a-restrict.yaml



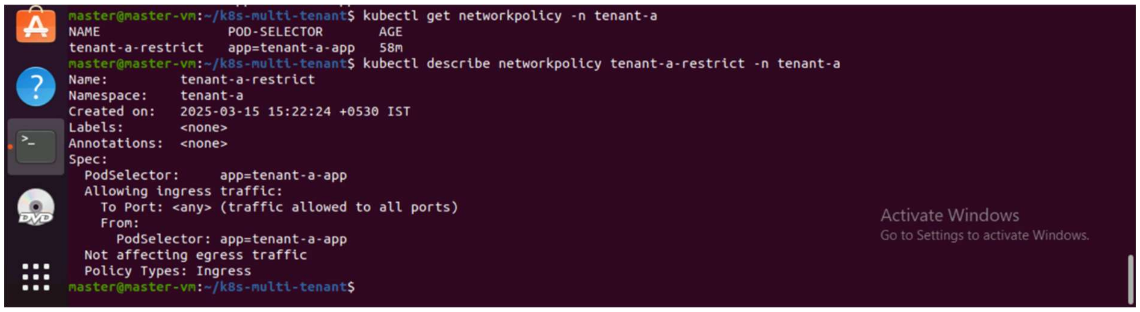
```
master@master-vm:~/k8s-multi-tenant$ kubectl apply -f tenant-a/tenant-a-restrict.yaml
networkpolicy.networking.k8s.io/tenant-a-restrict created
```

Verify Network Policy

To verify the network policy for Tenant A, run the following commands:

kubectl get networkpolicy -n tenant-a

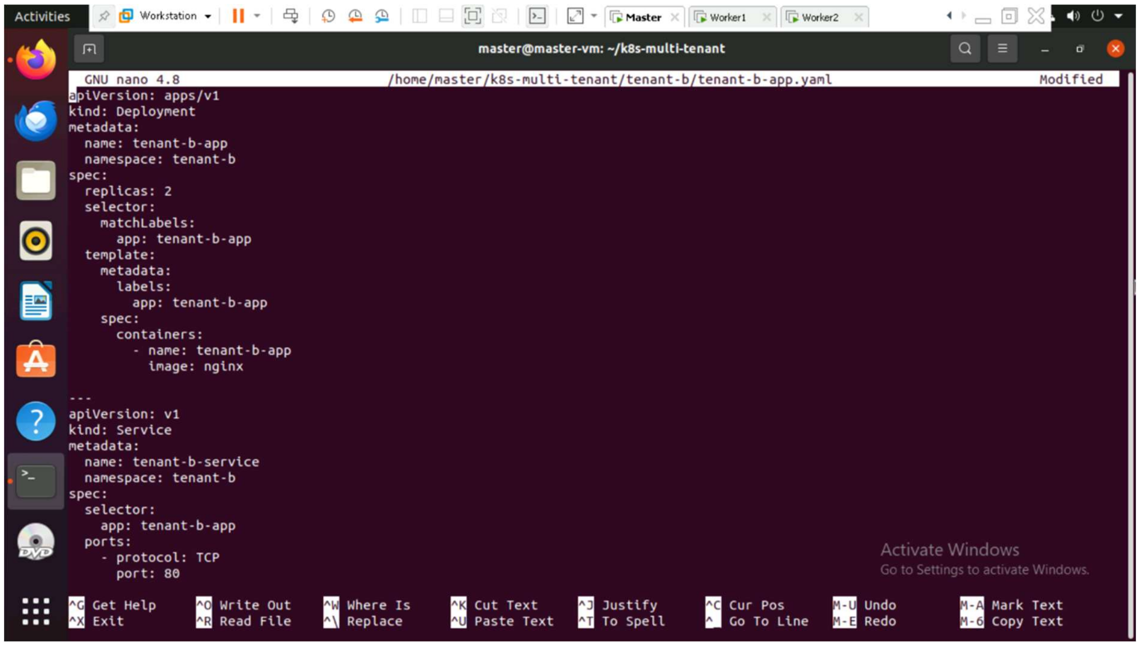
kubectl describe networkpolicy tenant-a-restrict -n tenant-a



```
master@master-vm:~/k8s-multi-tenant$ kubectl get networkpolicy -n tenant-a
NAME                POD-SELECTOR       AGE
tenant-a-restrict   app=tenant-a-app   58m
master@master-vm:~/k8s-multi-tenant$ kubectl describe networkpolicy tenant-a-restrict -n tenant-a
Name:         tenant-a-restrict
Namespace:    tenant-a
Created on:   2025-03-15 15:22:24 +0530 IST
Labels:       <none>
Annotations:  <none>
Spec:
  PodSelector:     app=tenant-a-app
  Allowing ingress traffic:
    To Port: <any> (traffic allowed to all ports)
    From:
      PodSelector: app=tenant-a-app
  Not affecting egress traffic
  Policy Types: Ingress
master@master-vm:~/k8s-multi-tenant$
```

Step 7: Create Deployment and Service for Tenant B

nano ~/k8s-multi-tenant/tenant-b/tenant-b-app.yaml



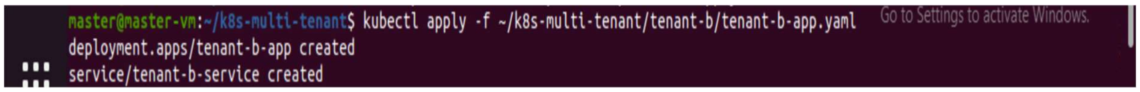
```
GNU nano 4.8          /home/master/k8s-multi-tenant/tenant-b/tenant-b-app.yaml          Modified
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tenant-b-app
  namespace: tenant-b
spec:
  replicas: 2
  selector:
    matchLabels:
      app: tenant-b-app
  template:
    metadata:
      labels:
        app: tenant-b-app
    spec:
      containers:
        - name: tenant-b-app
          image: nginx

---
apiVersion: v1
kind: Service
metadata:
  name: tenant-b-service
  namespace: tenant-b
spec:
  selector:
    app: tenant-b-app
  ports:
    - protocol: TCP
      port: 80
```

Activities
Workstation
Master
Worker1
Worker2
master@master-vm: ~/k8s-multi-tenant
Activate Windows
Go to Settings to activate Windows.
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos M-U Undo M-A Mark Text
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line M-E Redo M-6 Copy Text

Apply the configuration:

kubectl apply -f ~/k8s-multi-tenant/tenant-b/tenant-b-app.yaml

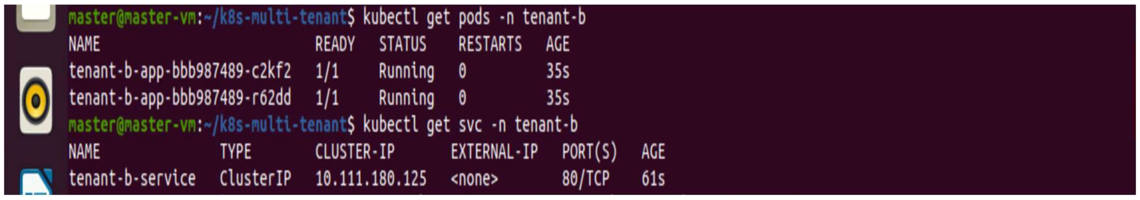


```
master@master-vm:~/k8s-multi-tenant$ kubectl apply -f ~/k8s-multi-tenant/tenant-b/tenant-b-app.yaml
deployment.apps/tenant-b-app created
service/tenant-b-service created
```

Verify the deployment:

kubectl get pods -n tenant-b

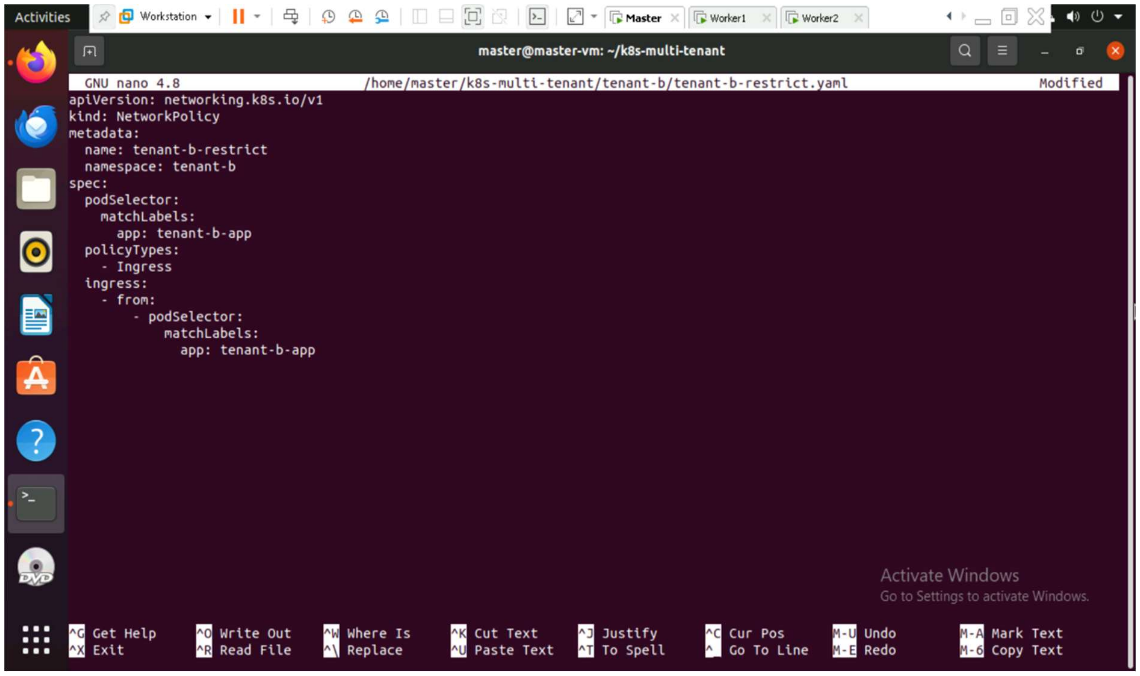
kubectl get svc -n tenant-b



```
master@master-vm:~/k8s-multi-tenant$ kubectl get pods -n tenant-b
NAME                            READY   STATUS    RESTARTS   AGE
tenant-b-app-bbb987489-c2kf2   1/1     Running   0          35s
tenant-b-app-bbb987489-r62dd   1/1     Running   0          35s
master@master-vm:~/k8s-multi-tenant$ kubectl get svc -n tenant-b
NAME               TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
tenant-b-service   ClusterIP   10.111.180.125   <none>        80/TCP    61s
```

Step 8: Restrict Network Access for Tenant A

nano ~/k8s-multi-tenant/tenant-b/tenant-b-restrict.yaml



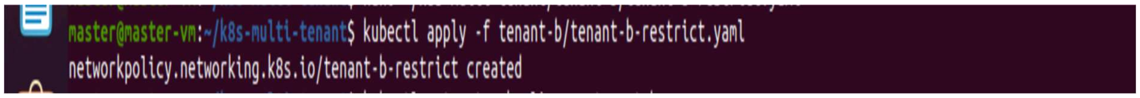
Activities
Workstation
Master
Worker1
Worker2
master@master-vm: ~/k8s-multi-tenant
Activate Windows
Go to Settings to activate Windows.

```
  GNU nano 4.8               /home/master/k8s-multi-tenant/tenant-b/tenant-b-restrict.yaml               Modified
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: tenant-b-restrict
  namespace: tenant-b
spec:
  podSelector:
    matchLabels:
      app: tenant-b-app
  policyTypes:
    - Ingress
  ingress:
    - from:
        - podSelector:
            matchLabels:
              app: tenant-b-app

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo       M-A Mark Text
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E Redo       M-6 Copy Text
```

Apply the network policy:

kubectl apply -f ~/k8s-multi-tenant/tenant-b/tenant-b-restrict.yaml



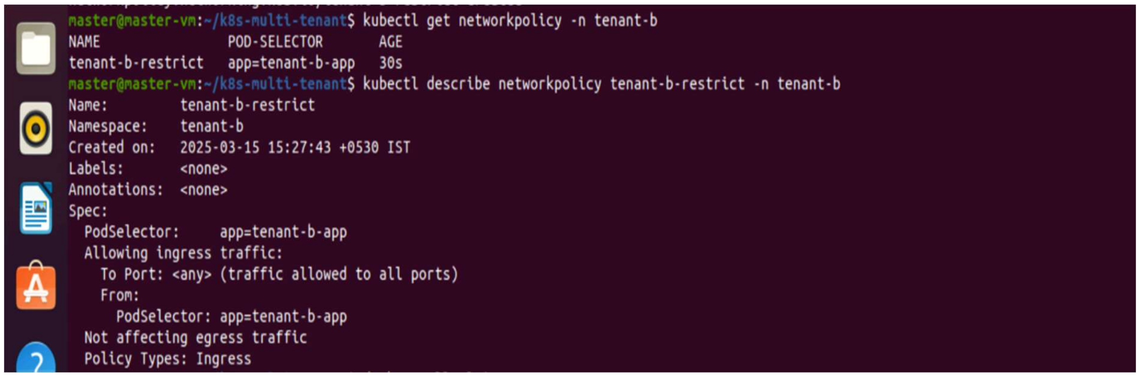
```
master@master-vm:~/k8s-multi-tenant$ kubectl apply -f tenant-b/tenant-b-restrict.yaml
networkpolicy.networking.k8s.io/tenant-b-restrict created
```

Step 9: Verify Network Policy

To verify the network policy for Tenant B, run the following commands:

kubectl get networkpolicy -n tenant-b

kubectl describe networkpolicy tenant-b-restrict -n tenant-b



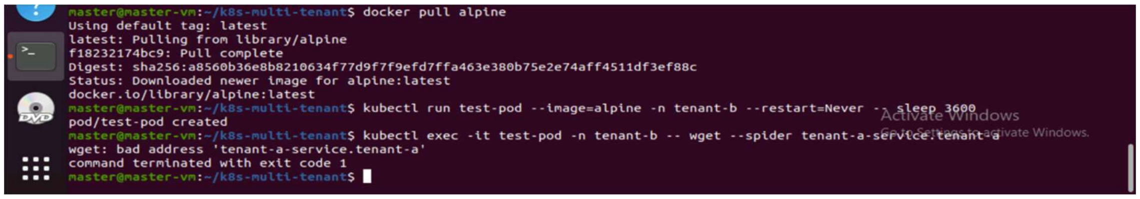
```
master@master-vm:~/k8s-multi-tenant$ kubectl get networkpolicy -n tenant-b
NAME                POD-SELECTOR       AGE
tenant-b-restrict   app=tenant-b-app   30s
master@master-vm:~/k8s-multi-tenant$ kubectl describe networkpolicy tenant-b-restrict -n tenant-b
Name:         tenant-b-restrict
Namespace:    tenant-b
Created on:   2025-03-15 15:27:43 +0530 IST
Labels:       <none>
Annotations:  <none>
Spec:
  PodSelector:     app=tenant-b-app
  Allowing ingress traffic:
    To Port: <any> (traffic allowed to all ports)
    From:
      PodSelector: app=tenant-b-app
  Not affecting egress traffic
  Policy Types: Ingress
```

Step 11: Test Tenant Isolation

Create a test pod in tenant-b and check access to tenant-a:

In worker docker run : docker pull alpine

```
kubectl run test-pod --image=alpine -n tenant-b --restart=Never -- sleep 3600
kubectl exec -it test-pod -n tenant-b -- wget --spider tenant-a-service.tenant-a
```



```
master@master-vm:~/k8s-multi-tenant$ docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
f18232174bc9: Pull complete
Digest: sha256:a8560b36e8b8210634f77d9f7f9efd7ffa463e380b75e2e74aff4511df3ef88c
Status: Downloaded newer image for alpine:latest
docker.io/library/alpine:latest
master@master-vm:~/k8s-multi-tenant$ kubectl run test-pod --image=alpine -n tenant-b --restart=Never -- sleep 3600
pod/test-pod created
master@master-vm:~/k8s-multi-tenant$ kubectl exec -it test-pod -n tenant-b -- wget --spider tenant-a-service.tenant-a
wget: bad address 'tenant-a-service.tenant-a'
command terminated with exit code 1
master@master-vm:~/k8s-multi-tenant$
```