

Kubernetes Multi-Tenant Project

Step 1: Check if Any Worker Node is Ready.

Run the following command to check the status of worker nodes:

```
master@master-vm:~$ kubectl get nodes
NAME          STATUS    ROLES    AGE   VERSION
master-vm     Ready     control-plane  17h   v1.30.11
worker1-vm    Ready     <none>    17h   v1.28.15
```

Step 2: Install Calico for Networking.

Apply the Calico manifest to enable networking:

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

Activate Windows
Go to Settings to activate

Step 3: Create Namespaces for Tenants.

```
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.

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

Step 5: Create Deployment and Service for Tenant A.

```
master@master-vm:~/k8s-multi-tenant/tenant-a$ nano tenant-a.app.yaml
```

Apply the configuration:

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

Step 6: Restrict Network Access for Tenant A.

```
master@master-vm:~/k8s-multi-tenant/tenant-a$ nano tenant-a-restrict.yaml
```

Step 7: Create Deployment and Service for Tenant B.

```
master@master-vm:~/k8s-multi-tenant/tenant-b$ nano tenant-b-restrict.yaml
```

Apply the deployment:

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

Step 8: Restrict Network Access for Tenant B.

```
master@master-vm:~/k8s-multi-tenant/tenant-b$ nano tenant-b-app.yaml
```

Apply the network policy:

```
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.

```
master@master-vm:~/k8s-multi-tenant$ kubectl get networkpolicy -n tenant-b
NAME                                POD-SELECTOR    AGE
tenant-b-restrict                  app=tenant-b-app 14s
```

Step 10: Final Folder Structure.

The final folder structure should look like this:

k8s-multi-tenant/

```
| — tenant-a/
|   | — tenant-a-app.yaml
|   | — tenant-a-restrict.yaml
| — tenant-b/
|   | — tenant-b-app.yaml
|   | — tenant-b-restrict.yaml
```

Step 11: Test Tenant Isolation.

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

In worker docker run.

```
worker1@worker1-vm:~$ sudo docker pull alpine
[sudo] password for worker1:
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 describe networkpolicy tenant-b-restrict -n tenant-b
Name:         tenant-b-restrict
Namespace:    tenant-b
Created on:    2025-03-18 10:09:34 +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
```

```
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$ kubectl get pods -n tenant-b -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP              NODE       NOMINATED NODE   READINESS GATES
tenant-b-app-79c78697f4-2kwgp       1/1     Running   0           6m25s  192.168.94.209  worker1-vm <none>           <none>
tenant-b-app-79c78697f4-pzvx5       1/1     Running   0           6m25s  192.168.94.208  worker1-vm <none>           <none>
test-pod                             1/1     Running   0           110s   192.168.94.210  worker1-vm <none>           <none>
master@master-vm:~/k8s-multi-tenant$ kubectl get pods -n tenant-a -o wide
error: unknown shorthand flag: '0' in -o
See 'kubectl get --help' for usage.
master@master-vm:~/k8s-multi-tenant$ kubectl get pods -n tenant-a -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP              NODE       NOMINATED NODE   READINESS GATES
tenant-a-app-759b876b88-8xq57       1/1     Running   0           9m4s   192.168.94.206  worker1-vm <none>           <none>
tenant-a-app-759b876b88-vz65w       1/1     Running   0           9m4s   192.168.94.207  worker1-vm <none>           <none>
```

Activate Windows
Go to Settings to activate Windows