

Kubernetes Multi-Tenant Project

Step 1: Check if Any Worker Node is Ready

- `kubectl get nodes`

```
master@master-vm:~$ kubectl get nodes
NAME          STATUS    ROLES    AGE   VERSION
master-vm     Ready     control-plane  17d   v1.28.15
worker1-vm    Ready     <none>      17d   v1.28.15
worker2-vm    NotReady  <none>      17d   v1.28.15
```

Step 2: Install Calico for Networking

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

```
master@master-vm:~/k8s-multi-tenant/tenant-a$ 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/bgpconfigurations.crd.projectcalico.org configured
customresourcedefinition.apiextensions.k8s.io/bgppeers.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/l2pools.crd.projectcalico.org configured
customresourcedefinition.apiextensions.k8s.io/l2preservations.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
```

Step 3: Create Namespaces for Tenants

- `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

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

Step 5: Create Deployment and Service for Tenant A and apply the configuration.

- `kubectl apply -f tenant-a/tenant-a-app.yaml`

```
master@master-vm:~/k8s-multi-tenant/tenant-a$ nano tenant-a-app.yaml
master@master-vm:~/k8s-multi-tenant/tenant-a$ kubectl apply -f tenant-a/tenant-a-app.yaml
error: the path "tenant-a/tenant-a-app.yaml" does not exist
master@master-vm:~/k8s-multi-tenant/tenant-a$ ls -l ~/k8s-multi-tenant/tenant-a/
total 4
-rw-rw-r-- 1 master master 506 Mar 15 15:10 tenant-a-app.yaml
master@master-vm:~/k8s-multi-tenant/tenant-a$ kubectl apply -f ~/k8s-multi-tenant/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 and apply the network policy.

- kubectl apply -f tenant-a/tenant-a-restrict.yaml

```
service/tenant-a-service created
master@master-vm:~/k8s-multi-tenant/tenant-a$ nano tenant-a-restrict.yaml
master@master-vm:~/k8s-multi-tenant/tenant-a$ kubectl apply -f tenant-a/tenant-a-restrict.yaml
error: the path "tenant-a/tenant-a-restrict.yaml" does not exist
master@master-vm:~/k8s-multi-tenant/tenant-a$ kubectl apply -f ~/k8s-multi-tenant/tenant-a/tenant-a-restrict.yaml
networkpolicy.networking.k8s.io/tenant-a-restrict created
```

Step 7: Create Deployment and Service for Tenant B and apply the deployment.

- kubectl apply -f tenant-b/tenant-b-app.yaml

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

```
master@master-vm:~/k8s-multi-tenant/tenant-b$ kubectl get pods -n tenant-b
NAME                                READY STATUS RESTARTS AGE
tenant-b-app-bbb987489-5hhf6        1/1   Running 0       25s
tenant-b-app-bbb987489-n44qk        1/1   Running 0       25s
```

```
master@master-vm:~/k8s-multi-tenant/tenant-b$ kubectl get svc -n tenant-b
NAME             TYPE        CLUSTER-IP    EXTERNAL-IP  PORT(S)    AGE
tenant-b-service ClusterIP   10.99.147.97  <none>       80/TCP     68s
```

Step 8: Restrict Network Access for Tenant B and Apply the network policy.

- kubectl apply -f tenant-b/tenant-b-restrict.yaml

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

Step 9: Verify Network Policy

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

```
master@master-vm:~/k8s-multi-tenant/tenant-b$ kubectl get networkpolicy -n tenant-b
NAME                POD-SELECTOR  AGE
tenant-b-restrict   app=tenant-b-app  26s
master@master-vm:~/k8s-multi-tenant/tenant-b$ kubectl describe networkpolicy tenant-b-restrict -n tenant-b
Name:                 tenant-b-restrict
Namespace:             tenant-b
Created on:            2025-03-15 15:39:38 +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/tenant-b$
```

Step 10: Test Tenant Isolation

- docker pull alpine

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

- 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/tenant-b$ kubectl run test-pod --image=alpine -n tenant-b --restart=Never -- sleep 3600
pod/test-pod created
master@master-vm:~/k8s-multi-tenant/tenant-b$ 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/tenant-b$
```