

KUBERNETES PROJECT - 2

Multi-Tenant Project

Step 1: Check if Any Worker Node is Ready

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

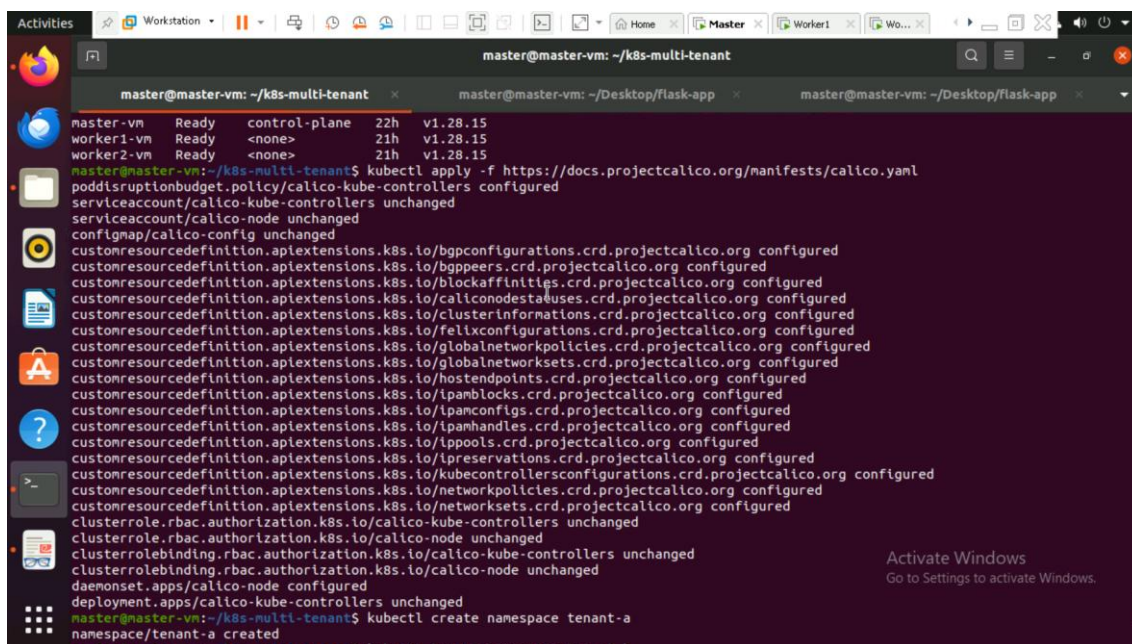
kubectl get nodes

```
master@master-vm:~/k8s-multi-tenant$ kubectl get nodes
NAME              STATUS    ROLES    AGE   VERSION
master-vm         Ready    control-plane   22h   v1.28.15
worker1-vm        Ready    <none>         21h   v1.28.15
worker2-vm        Ready    <none>         21h   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>



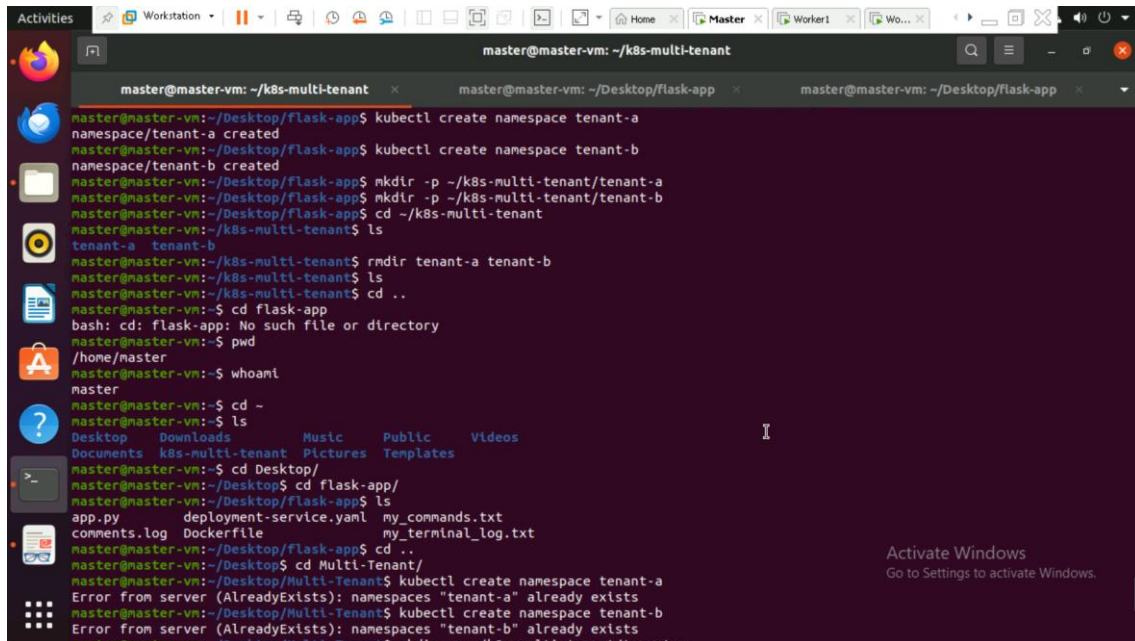
```
master@master-vm:~/k8s-multi-tenant$ kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
poddisruptionbudget.policy/calico-kube-controllers configured
serviceaccount/calico-kube-controllers unchanged
serviceaccount/calico-node unchanged
configmap/calico-config unchanged
customresourcedefinition.apiextensions.k8s.io/bgppolicies.crd.projectcalico.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
master@master-vm:~/k8s-multi-tenant$ kubectl create namespace tenant-a
namespace/tenant-a created
```

Step 3: Create Namespaces for Tenants

To isolate tenants, create separate namespaces:

kubectl create namespace tenant-a kubectl

create namespace tenant-b



The screenshot shows a terminal window with the following commands and output:

```
master@master-vm: ~/k8s-multi-tenant
master@master-vm:~/Desktop/flask-app$ kubectl create namespace tenant-a
namespace/tenant-a created
master@master-vm:~/Desktop/flask-app$ kubectl create namespace tenant-b
namespace/tenant-b created
master@master-vm:~/Desktop/flask-app$ mkdir -p ~/k8s-multi-tenant/tenant-a
master@master-vm:~/Desktop/flask-app$ mkdir -p ~/k8s-multi-tenant/tenant-b
master@master-vm:~/Desktop/flask-app$ cd ~/k8s-multi-tenant
master@master-vm:~/k8s-multi-tenant$ ls
tenant-a  tenant-b
master@master-vm:~/k8s-multi-tenant$ rmdir tenant-a tenant-b
master@master-vm:~/k8s-multi-tenant$ ls
master@master-vm:~/k8s-multi-tenant$ cd ..
master@master-vm:~/Desktop/flask-app$ cd flask-app
bash: cd: flask-app: No such file or directory
master@master-vm:~/Desktop/flask-app$ pwd
/home/master
master@master-vm:~/Desktop/flask-app$ whoami
master
master@master-vm:~/Desktop/flask-app$ cd -
master@master-vm:~/Desktop/flask-app$ ls
Desktop  Downloads  Music  Public  Videos
Documents  k8s-multi-tenant  Pictures  Templates
master@master-vm:~/Desktop/flask-app$ cd Desktop/
master@master-vm:~/Desktop$ cd flask-app/
master@master-vm:~/Desktop/flask-app$ ls
app.py  deployment-service.yaml  my_commands.txt
comments.log  Dockerfile  my_terminal_log.txt
master@master-vm:~/Desktop/flask-app$ cd ..
master@master-vm:~/Desktop$ cd Multi-Tenant/
master@master-vm:~/Desktop/Multi-Tenant$ kubectl create namespace tenant-a
Error from server (AlreadyExists): namespaces "tenant-a" already exists
master@master-vm:~/Desktop/Multi-Tenant$ kubectl create namespace tenant-b
Error from server (AlreadyExists): namespaces "tenant-b" already exists
```

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
```

Step 5: Create Deployment and Service for Tenant A

Create tenant-a-app.yaml in the tenant-a/ directory with the following contents:

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

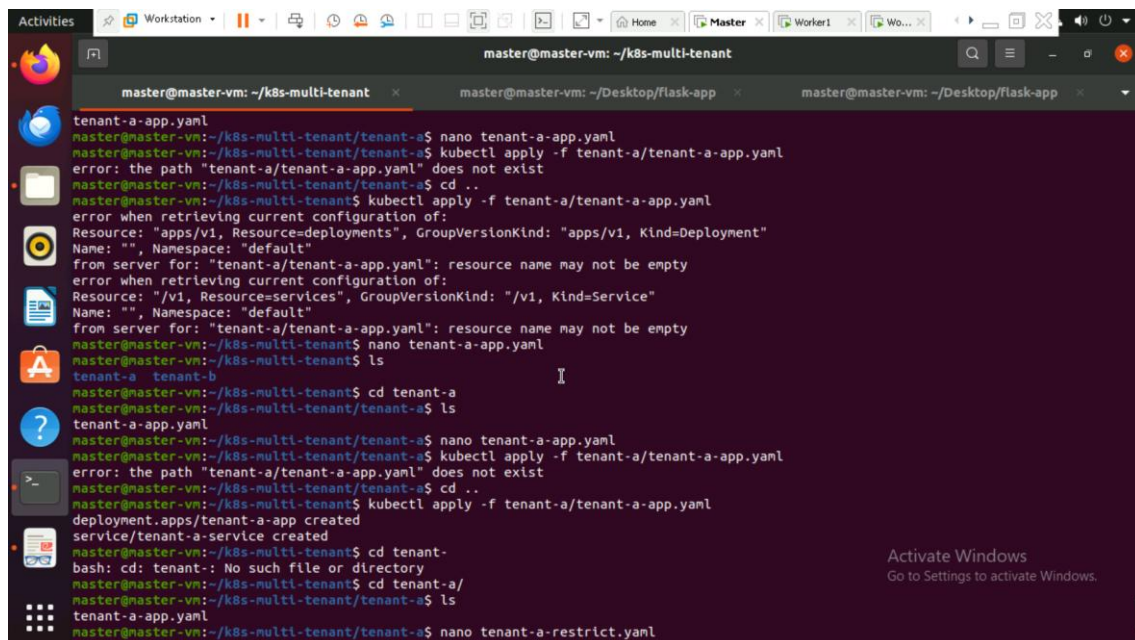
ports: - protocol:

TCP port: 80

targetPort: 80

Apply the configuration:

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



```
master@master-vm: ~/k8s-multi-tenant
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$ cd ..
master@master-vm: ~/k8s-multi-tenant$ kubectl apply -f tenant-a/tenant-a-app.yaml
error when retrieving current configuration of:
Resource: "apps/v1, Resource=deployments", GroupVersionKind: "apps/v1, Kind=Deployment"
Name: "", Namespace: "default"
from server for: "tenant-a/tenant-a-app.yaml": resource name may not be empty
error when retrieving current configuration of:
Resource: "/v1, Resource=services", GroupVersionKind: "/v1, Kind=Service"
Name: "", Namespace: "default"
from server for: "tenant-a/tenant-a-app.yaml": resource name may not be empty
master@master-vm: ~/k8s-multi-tenant$ nano tenant-a-app.yaml
master@master-vm: ~/k8s-multi-tenant$ ls
tenant-a  tenant-b
master@master-vm: ~/k8s-multi-tenant$ cd tenant-a
master@master-vm: ~/k8s-multi-tenant/tenant-a$ ls
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$ cd ..
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
master@master-vm: ~/k8s-multi-tenant$ cd tenant-a
bash: cd: tenant-a: No such file or directory
master@master-vm: ~/k8s-multi-tenant$ cd tenant-a/
master@master-vm: ~/k8s-multi-tenant/tenant-a$ ls
tenant-a-app.yaml
master@master-vm: ~/k8s-multi-tenant/tenant-a$ nano tenant-a-restrict.yaml
```

Step 6: Restrict Network Access for Tenant A

Create tenant-a-restrict.yaml in the tenant-a/ directory with the following contents:

```
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 Apply
```

the network policy:

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

Step 7: Create Deployment and Service for Tenant B

Create tenant-b-app.yaml in the tenant-b/ directory with the following contents:

```
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

targetPort: 80

Apply the deployment:

kubectl apply -f tenant-b/tenant-b-app.yaml Verify the deployment:

kubectl get pods -n tenant-b kubectl get svc -n tenant-b

Step 8: Restrict Network Access for Tenant B

Create tenant-b-restrict.yaml in the tenant-b/ directory with the following contents:

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 Apply

the network policy:

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

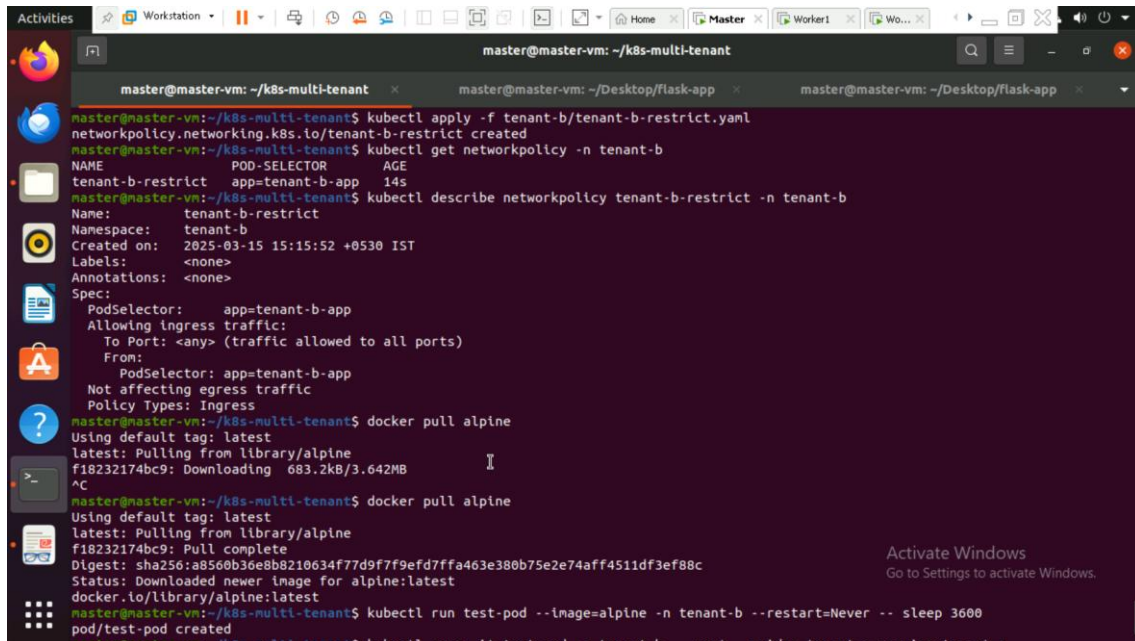
```
master@master-vm: ~/k8s-multi-tenant
master@master-vm:~/k8s-multi-tenant/tenant-a$ nano tenant-a-restrict.yaml
master@master-vm:~/k8s-multi-tenant/tenant-a$ cd ..
master@master-vm:~/k8s-multi-tenant$ kubectl apply -f tenant-a/tenant-a-restrict.yaml
networkpolicy.networking.k8s.io/tenant-a-restrict created
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 tenant-b/tenant-b-app.yaml
error: the path "tenant-b/tenant-b-app.yaml" does not exist
master@master-vm:~/k8s-multi-tenant/tenant-b$ cd ..
master@master-vm:~/k8s-multi-tenant$ kubectl apply -f tenant-b/tenant-b-app.yaml
deployment.apps/tenant-b-app created
service/tenant-b-service created
master@master-vm:~/k8s-multi-tenant$ kubectl get pods -n tenant-b
NAME                                READY   STATUS    RESTARTS   AGE
tenant-b-app-bbb987489-9db97        1/1     Running   0           19s
tenant-b-app-bbb987489-v2bh6        0/1     ContainerCreating   0           19s
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   10.96.82.226  <none>       80/TCP     34s
master@master-vm:~/k8s-multi-tenant$ cd tenant-b
master@master-vm:~/k8s-multi-tenant/tenant-b$ ls
tenant-b-app.yaml
master@master-vm:~/k8s-multi-tenant/tenant-b$ nano tenant-b-restrict.yaml
master@master-vm:~/k8s-multi-tenant/tenant-b$ cd ..
master@master-vm:~/k8s-multi-tenant$ kubectl apply -f tenant-b/tenant-b-restrict.yaml
networkpolicy.networking.k8s.io/tenant-b-restrict created
master@master-vm:~/k8s-multi-tenant$ kubectl get networkpolicy -n tenant-b
NAME                POD-SELECTOR   AGE
tenant-b-restrict   app=tenant-b-app   14s
master@master-vm:~/k8s-multi-tenant$ kubectl describe networkpolicy tenant-b-restrict -n tenant-b
Name:
Namespace: tenant-b
Created on: 2025-03-15 15:15:52 +0530 IST
```

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
master@master-vm:~/k8s-multi-tenant$ kubectl apply -f tenant-b/tenant-b-restrict.yaml
networkpolicy.networking.k8s.io/tenant-b-restrict created
master@master-vm:~/k8s-multi-tenant$ kubectl get networkpolicy -n tenant-b
NAME                                POD-SELECTOR  AGE
tenant-b-restrict                  app=tenant-b-app  14s
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 15:15:52 +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$ docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
f18232174bc9: Downloading  683.2kB/3.642MB
^C
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
```

Step 10: Final Folder Structure

The final folder structure should look like this:

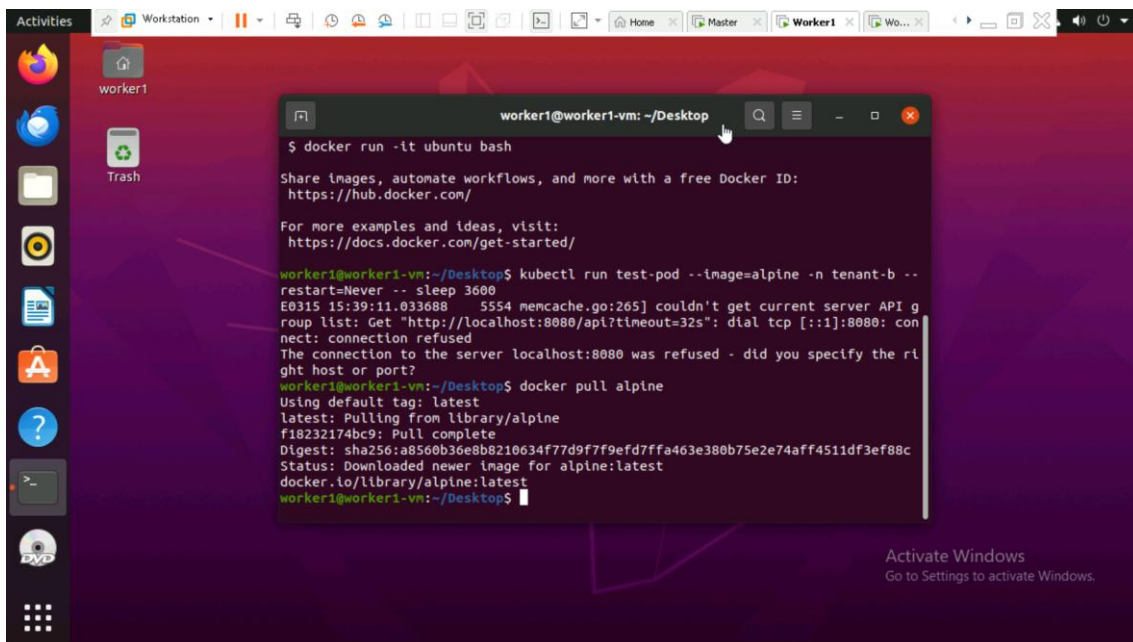
k8s-multi-tenant/

```
|— 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 docker pull alpine



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

