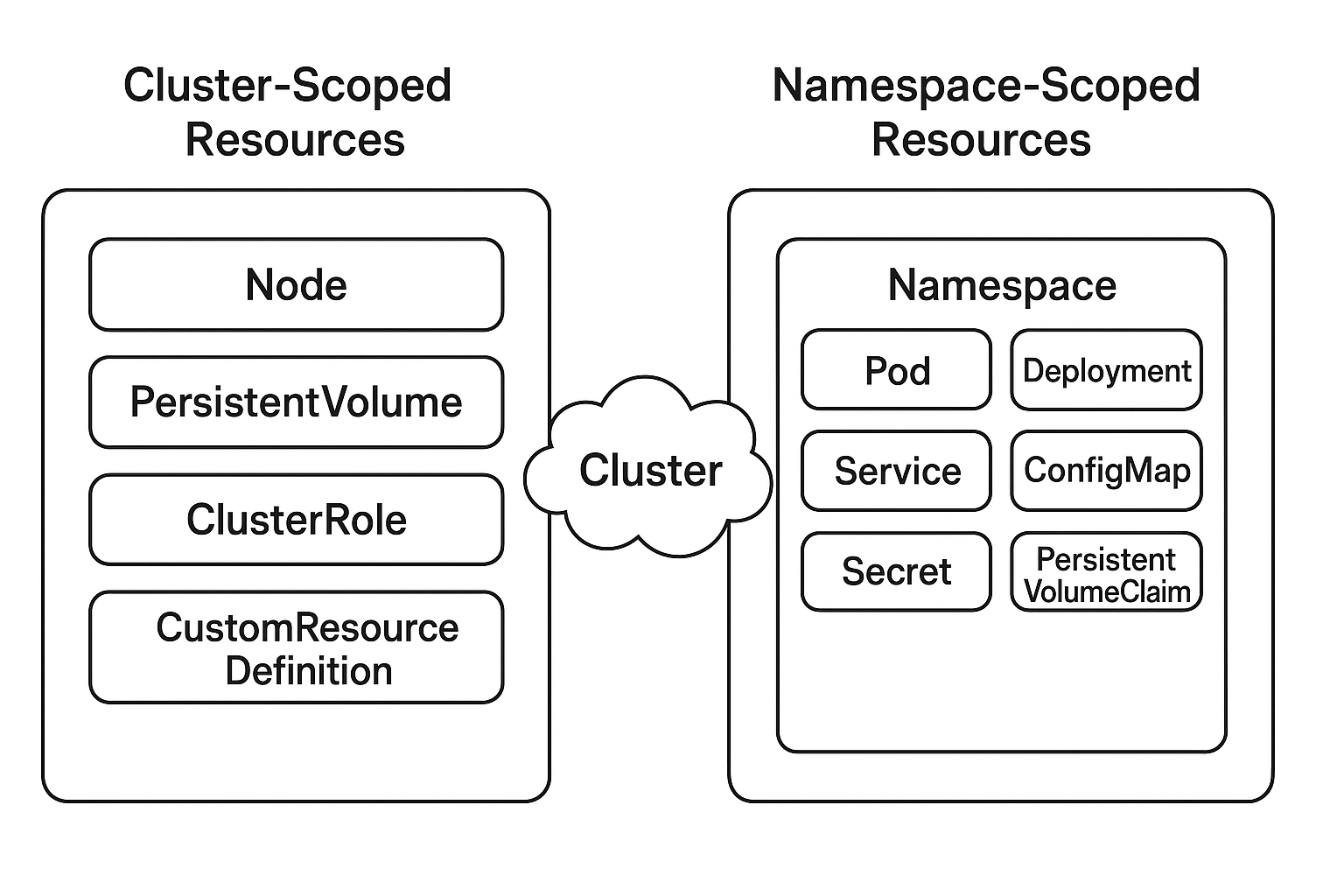
Cluster & Namespace Resources Overview

Cluster Scop resources & Namespace Scope Resources



# Cluster Scoped Resources

**1. Cluster-Scoped Resources**

**What it is:**

Cluster-scoped resources exist at the **cluster level**, meaning they are **not confined to any particular namespace**. They can be accessed or modified from anywhere in the cluster and apply globally.

**Why it exists / why we use it:**

They are used for configurations or entities that are **shared across all namespaces** or affect the entire cluster.

| **Resource** | **What it is** | **Why it exists / use case** |
| --- | --- | --- |
| Node | Represents a physical or virtual machine in the cluster | To manage cluster capacity and scheduling of workloads across nodes |
| PersistentVolume (PV) | Represents storage at the cluster level | Provides storage that can be claimed by any namespace via PersistentVolumeClaims (PVCs) |
| ClusterRole | Defines permissions at cluster level | To give access rights across all namespaces (global RBAC control) |
| ClusterRoleBinding | Binds a ClusterRole to a user or group | To assign cluster-wide permissions to users/groups |
| CustomResourceDefinition (CRD) | Defines a new type of resource globally | To extend Kubernetes with custom resources available cluster-wide |
| StorageClass | Defines storage policies at cluster level | To standardize storage provision for PVs across all namespaces |
| Namespace | Logical partition of the cluster (but itself is cluster-scoped) | To create isolated environments for workloads |

**2. Namespace-Scoped Resources**

**What it is:**

**Namespace-scoped resources exist inside a specific namespace**, meaning their scope and visibility are **limited to that namespace**.

**Why it exists / why we use it:**

They are used to **organize resources logically**, provide **multi-tenancy**, and **limit the impact of resources** to a single namespace.

| **Resource** | **What it is** | **Why it exists / use case** |
| --- | --- | --- |
| Pod | The smallest deployable unit in Kubernetes | Runs containerized applications within a namespace |
| Deployment | Manages a set of identical Pods | Provides scaling, updates, and rollback for applications in a namespace |
| Service | Exposes Pods internally or externally | Allows communication within or outside the namespace |
| ConfigMap | Stores configuration data | Decouples configuration from containers for apps in the namespace |
| Secret | Stores sensitive data | Keeps passwords, keys, tokens securely for apps in the namespace |
| PersistentVolumeClaim (PVC) | Requests storage from PVs | Lets applications in the namespace claim storage |
| Role | Defines permissions within a namespace | Provides namespace-level access control |
| RoleBinding | Binds a Role to a user or group | Assigns namespace-scoped permissions to users/groups |

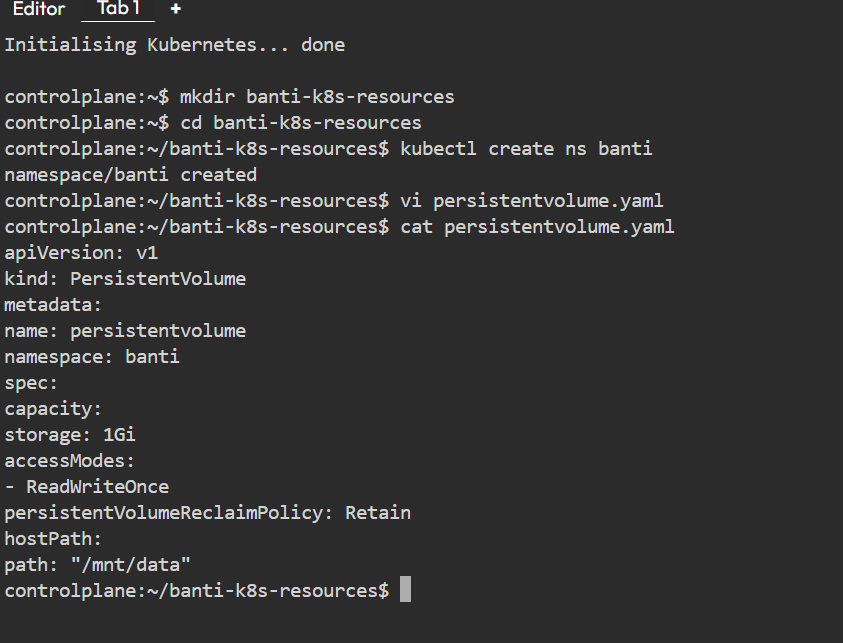
## K8s Ops to do

Make a folder banti-k8s-resources

Change the folder to get into that

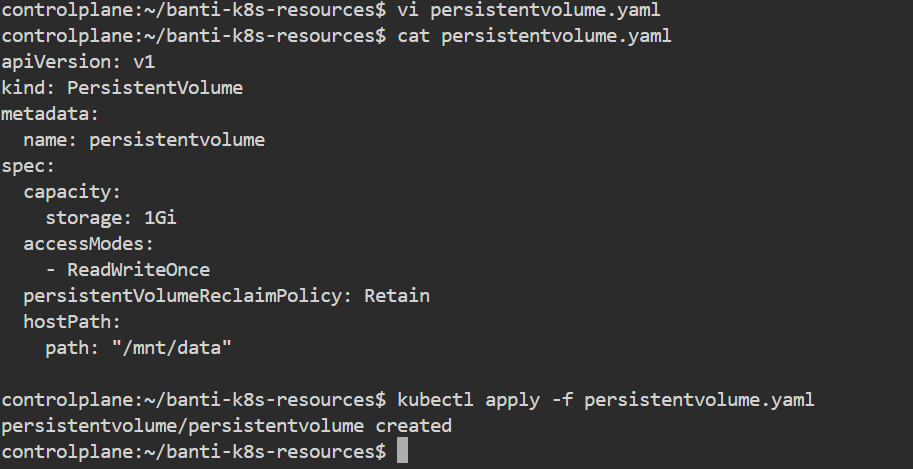
Create a namespace with my name – banti

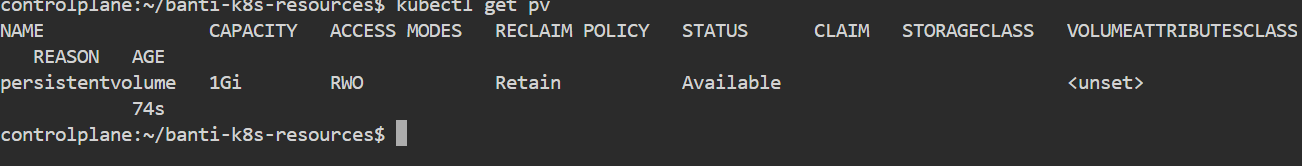
persistentvolume.yaml



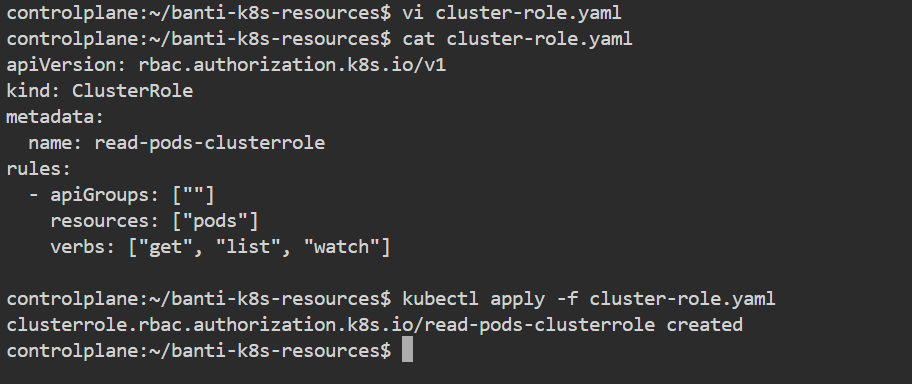
**Commands:**

**kubectl apply -f persistentvolume.yaml**

  
**kubectl get pv**



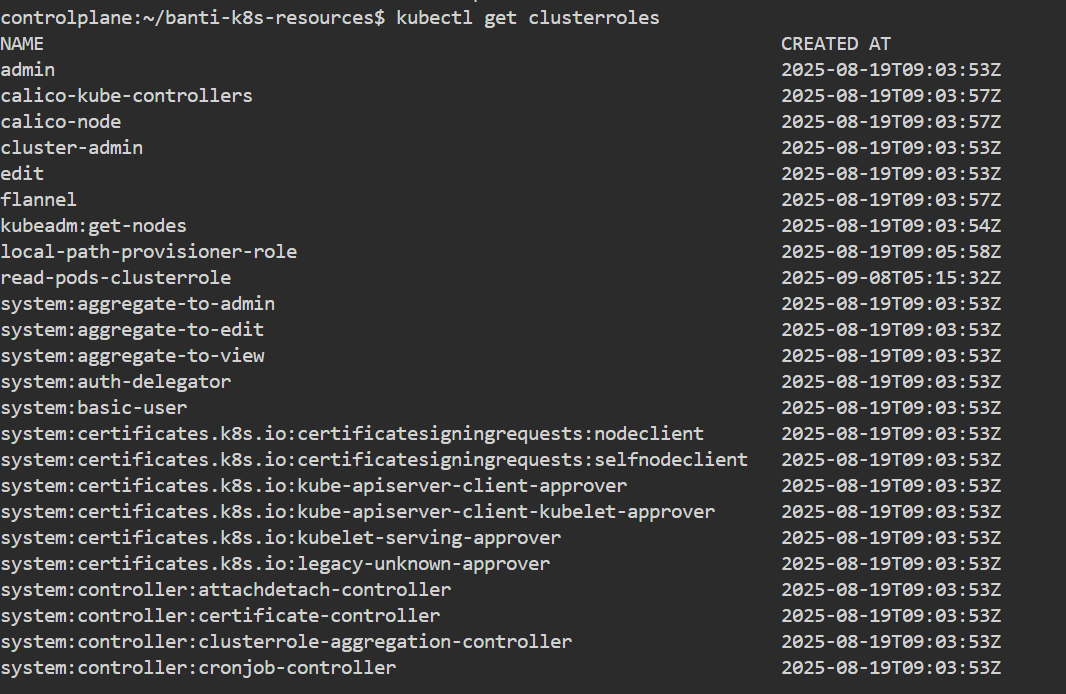
## ClusterRole



**Commands:**

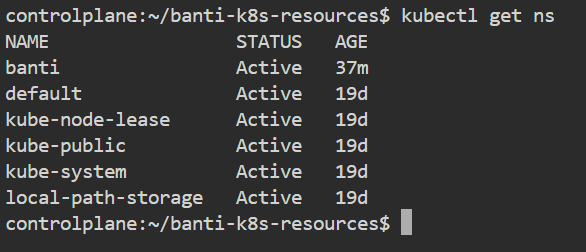
**kubectl apply -f cluster-role.yaml**

**kubectl get clusterroles**

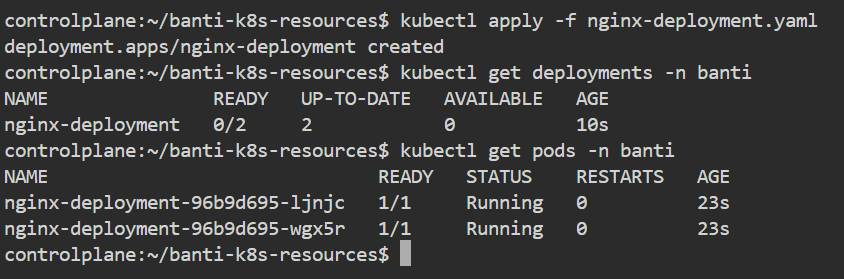


# Namespace Scoped Resources (Namespace = banti)

## 1. Namespace

  
**kubectl get ns**

## 2. Deployment

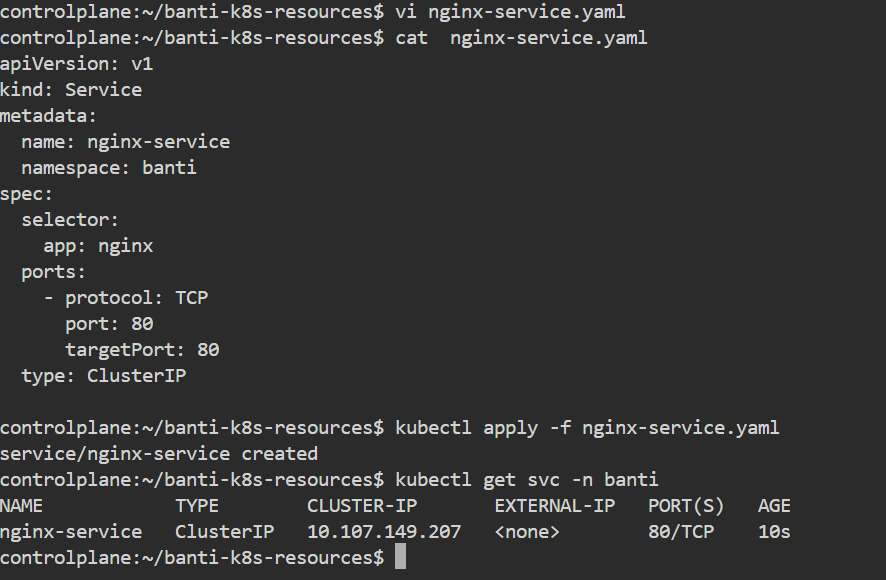
File: nginx-deployment.yaml

**Commands:**

**kubectl apply -f nginx-deployment.yaml  
kubectl get deployments -n banti  
kubectl get pods -n banti**

## 3. Service

File: nginx-service.yaml

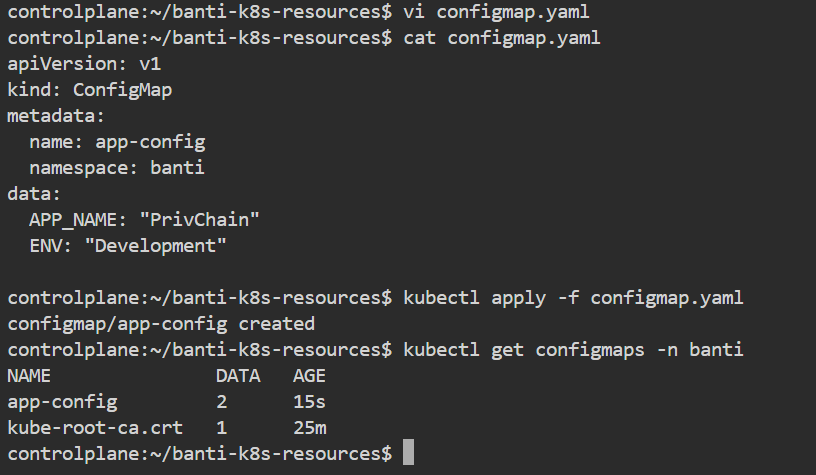


**Commands:**

**kubectl apply -f nginx-service.yaml  
kubectl get svc -n banti**

## 4. ConfigMap

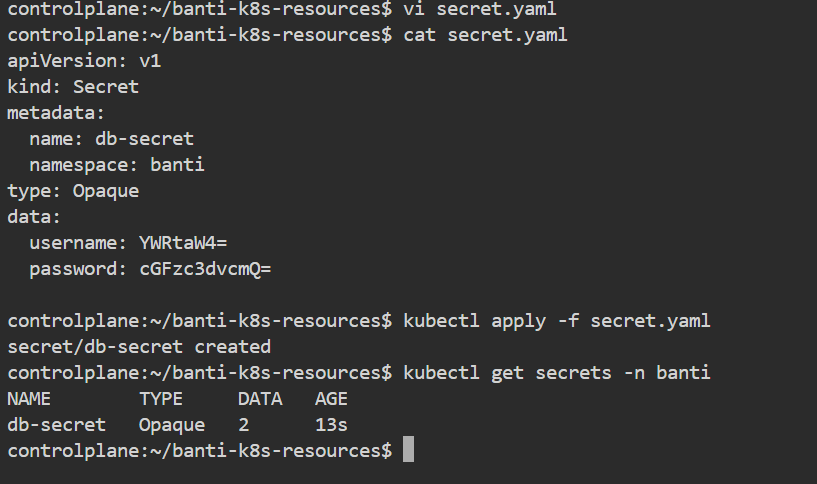
File: configmap.yaml



**Commands:**

**kubectl apply -f configmap.yaml  
kubectl get configmaps -n banti**

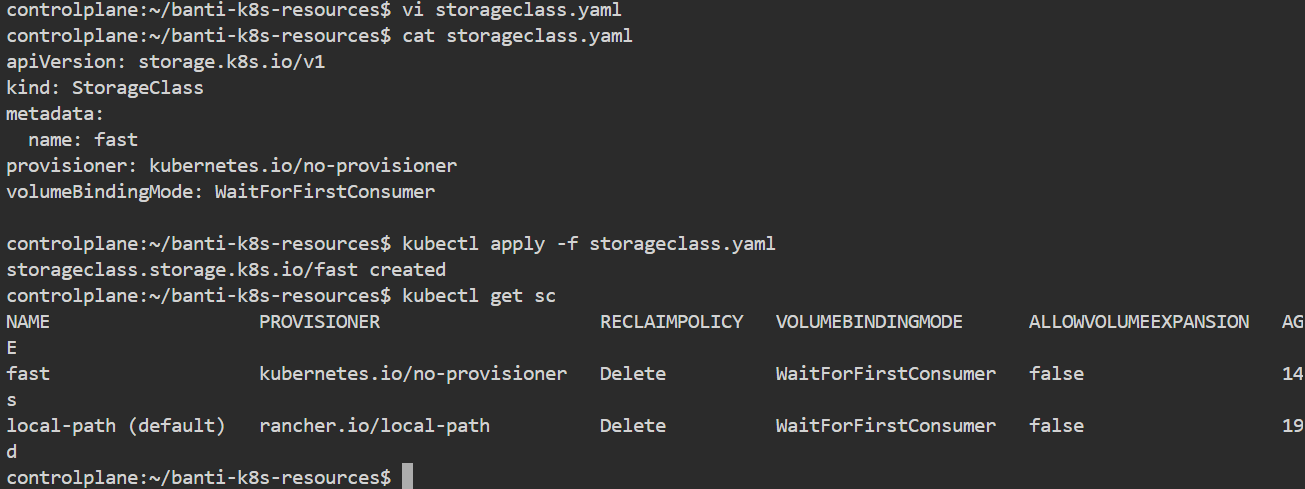
## 5. Secret

File: secret.yaml

**Commands**:

**kubectl apply -f secret.yaml  
kubectl get secrets -n banti**

## 5. Storageclass



**Commands :**

**vi storageclass.yaml**

**cat storageclass.yaml**

**kubectl apply -f storageclass.yaml**

**kubectl get sc**