Cluster-Scoped Resources & Namespace-Scoped Resources

* **Cluster-Scoped Resources**

Cluster-scoped resources are not bound to any namespace. They exist at the cluster level and are used for managing infrastructure, storage, security, and API extensions across the entire cluster.

Key Points:

 They define **infrastructure-level settings.**

 Managed by **cluster administrators** (not app developers).

 Example: Nodes, PersistentVolumes, ClusterRoles, StorageClasses.

 When you run kubectl get <resource> -n <namespace>, it will fail for these resources.

**1. Node**

What it is: A Node is a physical or virtual machine in the cluster where Pods run.

Why it is : Nodes provide the compute power for running applications and are managed by the control plane.

**2. PersistentVolume (PV)**

What it is: A PersistentVolume is a cluster-wide storage resource provisioned by an administrator.

Why it is used: It allows storage to be shared and reused across multiple namespaces through PersistentVolumeClaims.

**3.ClusterRole**

What it is: A ClusterRole defines a set of permissions that apply across the entire cluster.

Why it is : It is needed when users, groups, or service accounts require access to resources in multiple namespaces.

**4.ClusterRoleBinding**

What it is: A ClusterRoleBinding connects a ClusterRole to a user, group, or service account.

Why it is : This allows cluster-wide permissions to be granted to entities.

**5.CustomResourceDefinition (CRD)**

What it is: A CRD extends the Kubernetes API by introducing new resource types.

Why it is : It allows developers to define and manage custom resources, enabling operators and advanced controllers.

**6.StorageClass**

What it is: A StorageClass defines how dynamic storage should be provisioned.

Why it is : It allows applications to request storage with specific characteristics (e.g., SSD, HDD, cloud disk).

**7.Namespace**

What it is: A Namespace is a logical partition within the cluster. It is itself a cluster-scoped resource.

Why it is : Namespaces organize workloads and isolate applications for different teams or environments.

**8.APIService**

What it is: An APIService registers a new API into the Kubernetes API server.

Why it is : It allows Kubernetes to integrate with extension APIs and aggregated services.

* **Namespace-Scoped Resources**

Namespace-scoped resources exist inside a specific namespace. They are mainly used to manage applications, workloads, and configurations for isolated environments within the same cluster.

Key Points:

 They define **application-level settings.**

 Usually managed by **developers and DevOps engineers.**

 Example: Pods, Deployments, Services, ConfigMaps, Secrets.

 You can have multiple Deployments with the same name in different namespaces (but not in the same one).

**1.Pod**

What it is: The smallest deployable unit in Kubernetes, consisting of one or more containers.

Why it is used: Pods run application workloads within a namespace.

**2.ReplicaSet**

What it is: A ReplicaSet ensures a defined number of identical Pods are running at all times.

Why it is used: It provides self-healing by replacing failed Pods.

**3.Deployment**

What it is: A Deployment manages ReplicaSets and provides declarative updates for applications.

Why it is used: It simplifies application rollout, scaling, and rollback.

**4.StatefulSet**

What it is: A StatefulSet manages stateful applications such as databases.

Why it is used: It provides stable network identities and persistent storage for each Pod.

**5.DaemonSet**

What it is: A DaemonSet ensures that one Pod runs on every Node in the cluster.

Why it is used: It is commonly used for logging, monitoring, and security agents.

**6.Job**

What it is: A Job runs Pods until a task is completed successfully.

Why it is used: It is useful for batch processing tasks like data processing or backups.

**7.CronJob**

What it is: A CronJob runs Jobs on a predefined schedule.

Why it is used: It automates scheduled tasks such as report generation or database backups.

**8.Service**

What it is: A Service provides a stable network endpoint to expose Pods.

Why it is used: It ensures reliable communication between applications, even if Pods are replaced.

**9.Ingress**

What it is: An Ingress manages external HTTP/HTTPS access to Services.

Why it is used: It provides routing, load balancing, and SSL termination for applications.

**10.ConfigMap**

What it is: A ConfigMap stores non-sensitive configuration data as key-value pairs.

Why it is used: It allows applications to be configured without rebuilding images.

**11.Secret**

What it is: A Secret stores sensitive data like passwords or tokens.

Why it is used: It secures confidential information and injects it into Pods safely.

**12.ServiceAccount**

What it is: A ServiceAccount provides an identity for Pods to interact with the Kubernetes API.

Why it is used: It manages Pod authentication and authorization securely.

**13.Role**

What it is: A Role defines a set of permissions within a single namespace.

Why it is used: It restricts access to resources only inside one namespace.

**14.RoleBinding**

What it is: A RoleBinding connects a Role to a user, group, or service account in a namespace.

Why it is : It enforces namespace-specific RBAC permissions.