* Scope of Resources in Kubernetes: Cluster vs Namespace

Cluster-scoped resources in Kubernetes exist and operate across the entire cluster, while namespace-scoped resources are confined and managed within individual namespaces, supporting isolation and multi-tenancy. The distinction is fundamental to Kubernetes architecture and affects resource access, control, and management.

1. **Cluster-scoped Resources**

Cluster-scoped resources are accessible cluster-wide, independent of namespaces. Common examples include:

* + - **Nodes:** Physical or virtual machines forming the cluster; they run application workloads. Nodes exist at the infrastructure level and have global identity because scheduling, monitoring, and control must happen across the whole cluster.
    - **PersistentVolumes (PV):** These are storage units provisioned for the entire cluster, which can then be claimed by pods in any namespace. PVs are scoped to the whole cluster so they can be shared and allocated dynamically wherever needed.
    - **Namespaces:** Although used to partition cluster resources, the namespace object itself is managed at cluster scope for isolation.
    - **ClusterRoles and ClusterRoleBindings (RBAC):** Define permissions for actions across the whole cluster and are critical for global security policies.
    - **StorageClasses:** Describe storage types available for use anywhere in the cluster.
    - **CustomResourceDefinitions (CRDs):** Extend Kubernetes by defining new resource types available cluster-wide.

These resources support the fundamental architecture by providing global infrastructure, security, and extensibility. They enable cluster-wide scheduling, shared storage, control policies, and API extension for custom platforms and tools

1. **Namespace-scoped Resources**

Namespace-scoped resources are isolated within a namespace and can be duplicated without conflict across the cluster. Examples include:

* + - **Pods:** The smallest compute unit that runs application containers. Pods within a namespace share names and configurations unique to that namespace.
    - **Deployments, ReplicaSets:** Manage lifecycle, scalability, and availability of pods within a namespace for microservices and logical environment separation.
    - **Services:** Network endpoints for pods, scoped to offer connectivity inside the namespace or exposed externally.
    - **Ingress, ConfigMaps, Secrets:** Configuration and access objects, remaining separate in each namespace to prevent configuration mixing and security leaks.
    - **PersistentVolumeClaim (PVC):** Claims for storage by pods; lives in a namespace and maps to cluster-scoped PVs.

Namespaces exist to:

* + - Isolate teams, environments (dev/prod/test), and applications.
    - Prevent naming conflicts and access leaks between workloads.
    - Enable resource quotas, limits, and fine-grained RBAC, restricting who can access and manage resources based on namespace boundaries.
    - Facilitate multi-tenancy, clean organization, and structured deployment patterns.

|  |  |  |
| --- | --- | --- |
| **Resource Type** | **Scope** | **Why It Is/Architecture Impact** |
| Nodes | Cluster | Schedules pods, manages workloads cluster-wide |
| PersistentVolumes | Cluster | Provides storage shared across namespaces |
| ClusterRoles/RBindings | Cluster | Controls authorization globally |
| Namespaces | Cluster | Enables resource partitioning |
| CRDs | Cluster | Custom API extension for all workloads |
| Pods | Namespace | Compute unit, isolated per tenant/team |
| Deployments/ReplicaSets | Namespace | Manages pod lifecycles within environment |
| Services | Namespace | Networking endpoints, isolated by environment |
| Ingress, ConfigMap, Secret | Namespace | Configuration & security, isolated by namespace |
| PVCs | Namespace | Storage claims, mapped to cluster-wide PVs |

* **Definition and Key Differences**
* **Cluster-scoped resource:** Exists outside of any namespace and affects the entire Kubernetes cluster. Example resources include Nodes, PersistentVolumes, ClusterRoles, StorageClasses, Namespaces themselves, and CustomResourceDefinitions (CRDs).
* **Namespaced resource:** Exists within a particular namespace, and multiple instances of the same resource can exist across different namespaces (e.g., Deployments, Services, Secrets, ConfigMaps).
* **Why Use Cluster-Scoped vs Namespaced**
* **Cluster-scoped resources** are used for infrastructure-wide components and configurations. They define or control aspects of the whole cluster, such as storage options, roles for global permissions, and machine management.
* **Namespaced resources** are designed for logical isolation, allowing multiple teams or projects to share a cluster while keeping their workloads separate. This prevents naming conflicts and enables fine-grained resource quotas and security policies for each team or environment.
* **Example Usage in Architecture**
* **Cluster-scoped:** Creating a PersistentVolume or ClusterRole means that resource can be referenced across namespaces and supports cluster-wide infrastructure decisions.
* **Namespaced:** Deploying an application in one namespace does not affect resources or applications in another namespace, ideal for multi-tenant or multi-project environments.

