**Namespace-scoped Resources**

These resources are tied to **one application or environment** (like dev, test, prod).  
They live inside namespaces so that different teams/apps don’t interfere with each other.

**1. Pod**

* **Role:** Smallest building block in Kubernetes. Runs one or more containers (like nginx or MongoDB).
* **Why namespace-scoped?**
  + Pods usually belong to one application.
  + Example: A nginx-pod in *dev* shouldn’t clash with a nginx-pod in *prod*.
  + Namespaces give isolation.

**2. ReplicaSet (RS)**

* **Role:** Makes sure a fixed number of Pods are always running (e.g., 3 replicas of nginx).
* **Why namespace-scoped?**
  + It only manages Pods of one app.
  + A ReplicaSet in *dev* shouldn’t touch Pods in *prod*.

**3. Deployment**

* **Role:** Manages ReplicaSets + Pods. Helps with rolling updates, rollbacks.
* **Why namespace-scoped?**
  + Each environment may have its own Deployment.
  + Example: frontend-deployment in *dev* may use a test image, while in *prod* it uses the stable image.

**4. Service (Svc)**

* **Role:** Provides a stable network identity (IP/DNS) to access Pods.
* **Why namespace-scoped?**
  + A frontend-service in *dev* may point to test Pods.
  + A frontend-service in *prod* points to production Pods.
  + Keeping them separate avoids conflicts.

**5. Ingress**

* **Role:** Manages external traffic into cluster (routes /api → backend-service).
* **Why namespace-scoped?**
  + Each app/team can define its own rules inside its namespace.
  + But the **IngressClass** (controller type) is cluster-scoped because the controller serves all namespaces.

**6. ConfigMap**

* **Role:** Stores app configuration (like database host = db.dev.local).
* **Why namespace-scoped?**
  + Configurations are different for each environment.
  + Example: dev DB vs prod DB.

**7. Secret**

* **Role:** Stores sensitive data (passwords, API keys, tokens).
* **Why namespace-scoped?**
  + Secrets belong to apps.
  + You don’t want the **prod DB password** to be visible in *dev*.

**Cluster-scoped Resources**

These resources belong to the **whole cluster**.  
They’re not tied to a single namespace, because multiple apps/namespaces may share them.

**8. Node**

* **Role:** A worker machine (VM/physical) where Pods run.
* **Why cluster-scoped?**
  + Pods from many namespaces can run on the same Node.
  + A Node is shared infrastructure.

**9. Namespace**

* **Role:** A virtual space inside the cluster to separate apps.
* **Why cluster-scoped?**
  + Namespaces are like “folders” in the cluster.
  + You can’t have a namespace inside another namespace.

**10. PersistentVolume (PV)**

* **Role:** Provides storage resources (EBS, NFS, Ceph).
* **Why cluster-scoped?**
  + PVs represent actual storage in the cluster.
  + Apps (via PVCs in namespaces) can claim storage from the shared pool.

**11. StorageClass**

* **Role:** Defines *how* storage should be created (e.g., AWS gp2 SSD).
* **Why cluster-scoped?**
  + All namespaces can request storage from the same StorageClass.
  + It defines cluster-wide storage behavior.

**12. CSIDriver**

* **Role:** Defines a Container Storage Interface driver (AWS EBS, Ceph, GCP PD).
* **Why cluster-scoped?**
  + A storage driver serves the entire cluster.
  + Example: AWS EBS driver must be available for all namespaces.

**13. IngressClass**

* **Role:** Defines which Ingress controller to use (NGINX, Traefik, etc.).
* **Why cluster-scoped?**
  + The controller works across the whole cluster.
  + You may have one for *external* traffic and another for *internal* traffic.

**Easy Rule to Remember**

* **Namespace-scoped → App-level resources** (Pods, Deployments, Services, ConfigMaps, Secrets).
* **Cluster-scoped → Infrastructure-level resources** (Nodes, Namespaces, Storage, Controllers).

REAL TIME ANALOGY  
**cluster = Home**

* The **cluster** is like your **entire house**.
* Things like **walls, electricity, water supply, main gate** belong to **everyone in the house**.
* These are like **cluster-scoped resources** (Nodes, StorageClasses, Namespaces, IngressClass).
* Nobody "owns" them individually — they’re shared infrastructure.

**Namespace = Private Room**

* A **namespace** is like a **room in the house**.
* Each room can be decorated differently, have its own bed, closet, study table.
* These are like **namespace-scoped resources** (Pods, Deployments, Services, ConfigMaps, Secrets).
* What happens in your room doesn’t disturb other rooms.

**Example**

* **Home electricity meter (cluster resource)** → shared for the entire house.
* **Your room’s fan & light (namespace resource)** → private to your room.
* **Storage tank (PersistentVolume, cluster-scoped)** → provides water to the whole house.
* **Your personal water bottle (PVC, namespace-scoped)** → only you use it in your room.