Namespaces

In Kubernetes, *namespaces* provides a mechanism for isolating groups of resources within a single cluster. Names of resources need to be unique within a namespace, but not across namespaces. Namespace-based scoping is applicable only for namespaced objects *(e.g. Deployments, Services, etc)* and not for cluster-wide objects *(e.g. StorageClass, Nodes, PersistentVolumes, etc)*.

When to Use Multiple Namespaces

Namespaces are intended for use in environments with many users spread across multiple teams, or projects. For clusters with a few to tens of users, you should not need to create or think about namespaces at all. Start using namespaces when you need the features they provide.

Namespaces provide a scope for names. Names of resources need to be unique within a namespace, but not across namespaces. Namespaces cannot be nested inside one another and each Kubernetes resource can only be in one namespace.

Namespaces are a way to divide cluster resources between multiple users (via resource quota).

It is not necessary to use multiple namespaces to separate slightly different resources, such as different versions of the same software: use labels to distinguish resources within the same namespace.

> **Note:** For a production cluster, consider *not* using the `default` namespace. Instead, make other namespaces and use those.

Initial namespaces

Kubernetes starts with four initial namespaces:

`default`

Kubernetes includes this namespace so that you can start using your new cluster without first creating a namespace.

`kube-node-lease`

This namespace holds Lease objects associated with each node. Node leases allow the kubelet to send heartbeats so that the control plane can detect node failure.

`kube-public`

This namespace is readable by *all* clients (including those not authenticated). This namespace is mostly reserved for cluster usage, in case that some resources should be visible and readable publicly throughout the whole cluster. The public aspect of this namespace is only a convention, not a requirement.

`kube-system`

The namespace for objects created by the Kubernetes system.

Working with Namespaces

Creation and deletion of namespaces are described in the Admin Guide documentation for namespaces.

> **Note:** Avoid creating namespaces with the prefix `kube-`, since it is reserved for Kubernetes system namespaces.

Viewing namespaces

You can list the current namespaces in a cluster using:

```
kubectl get namespace
```

```
NAME              STATUS   AGE
default           Active   1d
kube-node-lease   Active   1d
kube-public       Active   1d
kube-system       Active   1d
```

Setting the namespace for a request

To set the namespace for a current request, use the `--namespace` flag.

For example:

```
kubectl run nginx --image=nginx --namespace=<insert-namespace-name-here>
kubectl get pods --namespace=<insert-namespace-name-here>
```

Setting the namespace preference

You can permanently save the namespace for all subsequent kubectl commands in that context.

```
kubectl config set-context --current --namespace=<insert-namespace-name-here>
# Validate it
kubectl config view --minify | grep namespace:
```

Namespaces and DNS

When you create a Service, it creates a corresponding DNS entry. This entry is of the form `<service-name>.<namespace-name>.svc.cluster.local`, which means that if a container only uses `<service-name>`, it will resolve to the service which is local to a namespace. This is useful for using the same configuration across multiple namespaces such as Development, Staging and Production. If you want to reach across namespaces, you need to use the fully qualified domain name (FQDN).

As a result, all namespace names must be valid RFC 1123 DNS labels.

Warning:
By creating namespaces with the same name as public top-level domains, Services in these namespaces can have short DNS names that overlap with public DNS records. Workloads from any namespace performing a DNS lookup without a trailing dot will be redirected to those services, taking precedence over public DNS.

To mitigate this, limit privileges for creating namespaces to trusted users. If required, you could additionally configure third-party security controls, such as admission webhooks, to block creating any namespace with the name of public TLDs.

Not all objects are in a namespace

Most Kubernetes resources (e.g. pods, services, replication controllers, and others) are in some namespaces. However namespace resources are not themselves in a namespace. And low-level resources, such as nodes and persistentVolumes, are not in any namespace.

To see which Kubernetes resources are and aren't in a namespace:

```
# In a namespace
kubectl api-resources --namespaced=true

# Not in a namespace
kubectl api-resources --namespaced=false
```

Automatic labelling

FEATURE STATE: `Kubernetes 1.22 [stable]`

The Kubernetes control plane sets an immutable label `kubernetes.io/metadata.name` on all namespaces. The value of the label is the namespace name.

What's next

- Learn more about creating a new namespace.
- Learn more about deleting a namespace.

Feedback

Was this page helpful?

Yes No

Last modified June 29, 2023 at 12:14 PM PST: Update content/en/docs/concepts/overview/working-with-objects/namespaces.md (10e15641bd)