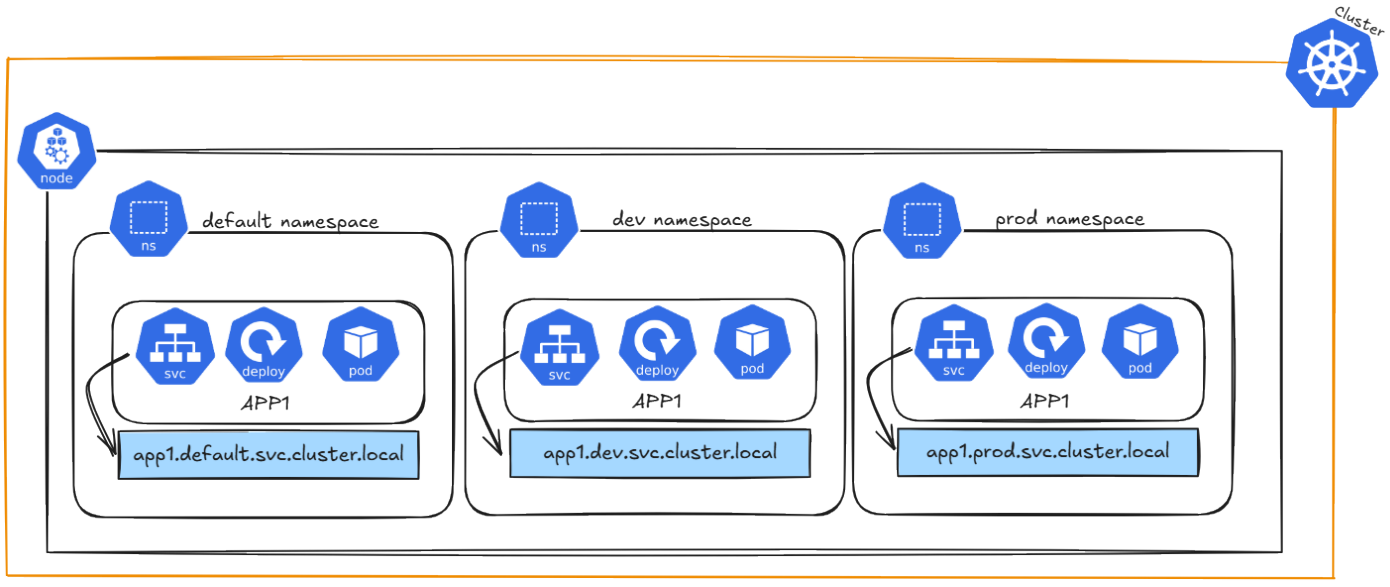
Namespace

Useful Links

- Kubernetes Official Documentation
- Namespaces Walkthrough

Architecture



Detailed Description

Kubernetes **namespaces** are like **virtual environments** within your Kubernetes cluster. They help organize and divide resources, making it easier to manage large environments.

For example, you can use namespaces to separate development, staging, and production environments within the same cluster.

- **Isolation**: Namespaces isolate resources like Pods, Services, and Deployments, so they don't interfere with each other.
- **Resource Management**: You can apply resource limits (CPU, memory) and access control to namespaces.
- **Organization**: They help group related resources, making it easier to manage them.
- **Access Control**: Namespaces can limit who can access certain resources through Kubernetes RBAC (Role-Based Access Control).
- **Complexity**: Having too many namespaces can complicate management, especially with network policies or cross-namespace communication.

Command Reference Guide

> Remeber to use dry-run and tee to check the configuration of each command first.
> `--dry-run=client -o yaml | tee nginx-deployment.yaml`

Create a Namespace using a YAML file (declarative method)

namespace.yaml

```
apiVersion: v1
kind: Namespace
metadata:
  name: dev
  labels:
    name: dev
---
apiVersion: v1
kind: Namespace
metadata:
  name: prod
  labels:
    name: prod
```

```
# Create namespaces
kubectl create -f namespace.yaml

# Get namespaces
kubectl get namespaces --show-labels

# Add context spaces (First get user and clustername)
kubectl config view
CLUSTER_NAME=$(kubectl config view --raw -o jsonpath='{.clusters[0].name}')
USER_NAME=$(kubectl config view --raw -o jsonpath='{.users[0].name}')
kubectl config set-context dev --namespace=dev --cluster=$CLUSTER_NAME --user=$USER_NAME
kubectl config set-context prod --namespace=prod --cluster=$CLUSTER_NAME --user=$USER_NAME
# We added  two new request contexts (dev and prod)
kubectl config view

# Switch context
kubectl config use-context dev

# Check current context
kubectl config current-context

# Create deployment in context dev and check pods
kubectl create deployment nginx-deployment --image=nginxdemos/hello --port=80
kubectl get pods

# Switch context and check pods
kubectl config use-context prod
kubectl get pods

# Delete context
kubectl config use-context default
kubectl config delete-context dev
kubectl config delete-context prod
```

Revision #5
Created 24 November 2024 15:57:37 by Admin
Updated 25 November 2024 15:57:31 by Admin