Deploying PostgreSQL Databases Across Multiple Namespaces in a Single Kubernetes Cluster

Deploying multiple PostgreSQL databases in Kubernetes within a single Kubernetes cluster while using multiple namespaces efficiently manages resources while maintaining isolation between different applications, such as the DICOM and Keycloak databases.

Implementation based on info found at this url: <https://www.enterprisedb.com/blog/how-deploy-pgadmin-kubernetes>

Why Use Multiple Namespaces in One Cluster?

1. Isolation of Resources: Using separate namespaces, such as blackford-dicom-ns and blackford-keycloak-ns, provides a clear separation of resources like services, pods, secrets, and configurations for each database. This isolation simplifies resource management while still taking advantage of shared infrastructure.

2. Efficient Resource Usage: A single Kubernetes cluster allows for shared compute resources, reducing the overhead compared to managing separate clusters for each application and is appropriate in a development environment.

3. Simplified Operations: Managing one Kubernetes cluster with multiple namespaces minimizes operational complexity. There is only one CloudNativePG operator and Kubernetes control plane to manage, which simplifies deploying, monitoring, and maintaining multiple databases.

Method of Deployment

The deployment process is automated using PowerShell and Shell scripts to set up the namespaces, the CloudNativePG operator, and the PostgreSQL databases.

Where there is a \*.ps1 script there is also a corresponding \*.sh script.

1. Namespace Creation: .\deploy.ps1

Namespaces for each environment are defined and applied using the following commands:

kubectl apply -f ./namespace.yaml

kubectl apply -f ./base/dicom/namespace.yaml

kubectl apply -f ./base/keycloak/namespace.yaml

2. Operator Deployment: also in .\deploy.ps1

The CloudNativePG operator is deployed using Helm and Kustomize:

kustomize build --enable-helm ./operator | kubectl --server-side --force-conflicts --validate=false apply -f -

3. Database Deployment:

Each PostgreSQL database is deployed using its own script and kustomize folder structure:

- DICOM PostgreSQL Deployment (deploy-dicom.ps1):

kustomize build --enable-helm ./base/dicom | kubectl apply --server-side --force-conflicts -f -

- Keycloak PostgreSQL Deployment (deploy-keycloak.ps1):

kustomize build --enable-helm ./base/keycloak | kubectl apply --server-side --force-conflicts -f -

4. pgAdmin Setup: Optional

A separate namespace and service are created for pgAdmin, allowing it to manage both PostgreSQL databases. pgAdmin is exposed via a NodePort service for easy external access.

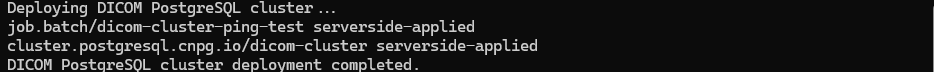
Example

.\deploy.ps1

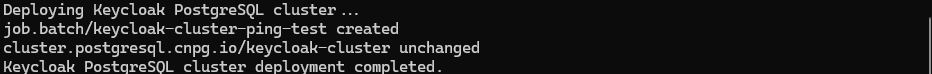
A screen shot of a computer

Description automatically generated

.\deploy-dicom.ps1



.\deploy-keycloak.ps1



pgAdmin

.\apply.ps1



kubectl port-forward svc/pgadmin-service 8080:80 -n blackford-platform-ns



See the commands.txt

kubectl get pods -n blackford-platform-ns