**Sealed Secrets in Kubernetes**

Sealed Secrets is an **open-source tool developed by Bitnami** that allows you to manage sensitive data securely in Kubernetes and GitOps workflows. Instead of storing raw Kubernetes Secrets directly in your Git repository (which is unsafe), you can use Sealed Secrets to encrypt them into *"SealedSecrets"*. These encrypted resources are safe to commit to version control, as only the target Kubernetes cluster can decrypt them.

**How Sealed Secrets Work**

Sealed Secrets rely on three main components:

* **SealedSecret (CRD):** A custom Kubernetes resource representing your encrypted Secret.
* **Sealed Secrets Controller:** Runs inside the cluster and decrypts SealedSecrets into native Kubernetes Secrets at runtime.
* **kubeseal CLI:** A client-side tool used to encrypt Secret manifests before committing them to Git.

**Protection Workflow**

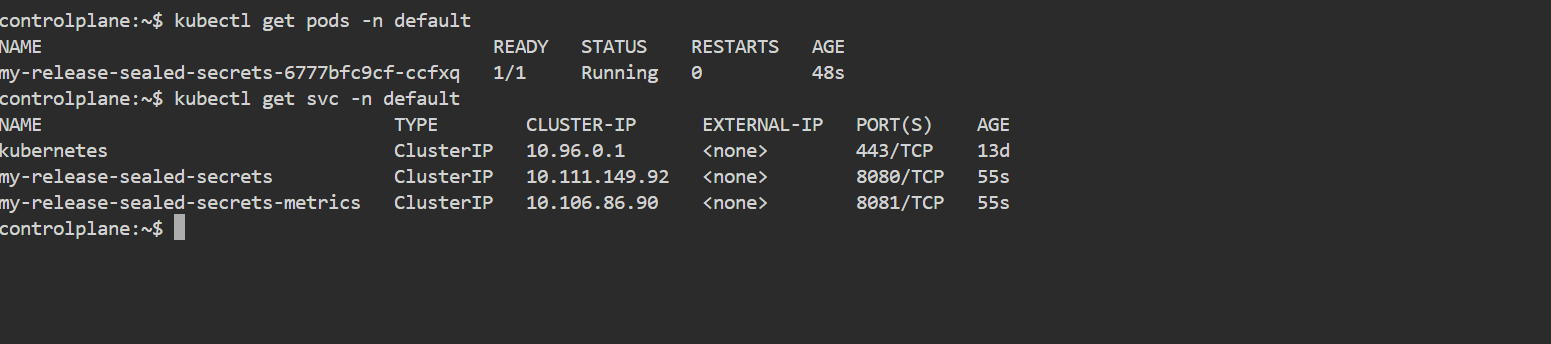
1. Install the **Sealed Secrets controller** in your Kubernetes cluster.
2. Use the **kubeseal CLI** to encrypt your sensitive data into a SealedSecret.
3. Commit and push the SealedSecret manifest to Git.
4. When applied to the cluster, the controller decrypts it into a regular **Kubernetes Secret**.

This ensures that your credentials are safe in Git repositories and can only be decrypted by the cluster that owns the matching key.

**Installation**

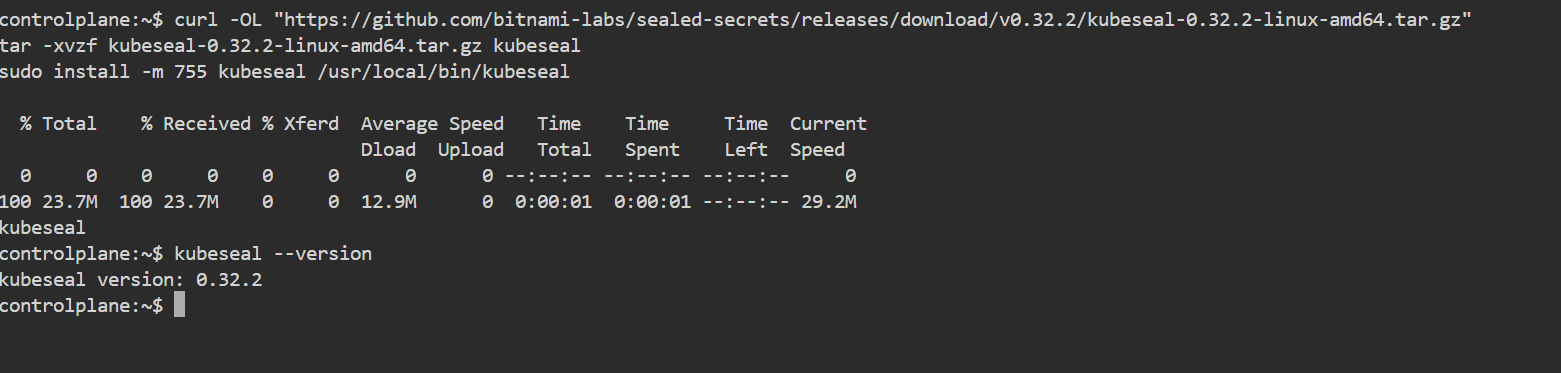
**Install the Sealed Secrets Controller**

helm repo add sealed-secrets https://bitnami-labs.github.io/sealed-secrets  
helm install my-release sealed-secrets/sealed-secrets



**Install kubeseal (Linux x86\_64)**

curl -OL "https://github.com/bitnami-labs/sealed-secrets/releases/download/v0.32.2/kubeseal-0.32.2-linux-amd64.tar.gz"  
tar -xvzf kubeseal-0.32.2-linux-amd64.tar.gz kubeseal  
sudo install -m 755 kubeseal /usr/local/bin/kubeseal



**Usage**

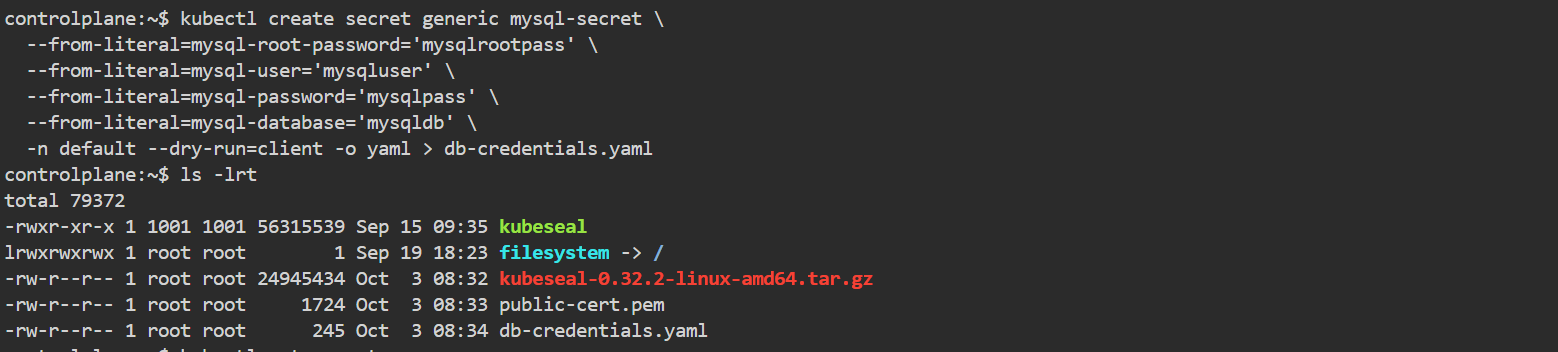
**Fetch the Controller’s Public Key**

kubeseal --fetch-cert \  
 --controller-name=my-release-sealed-secrets \  
 --controller-namespace=default > public-cert.pem



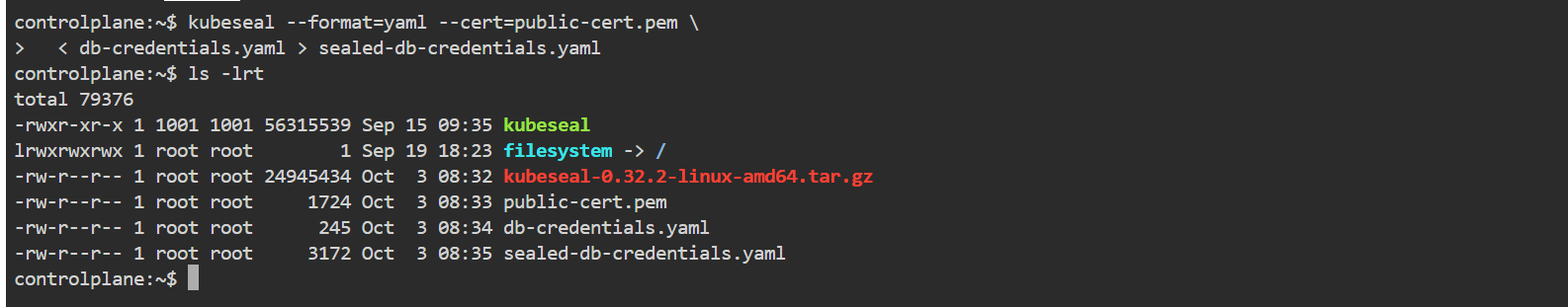
**Create a Kubernetes Secret (Dry Run in local)**

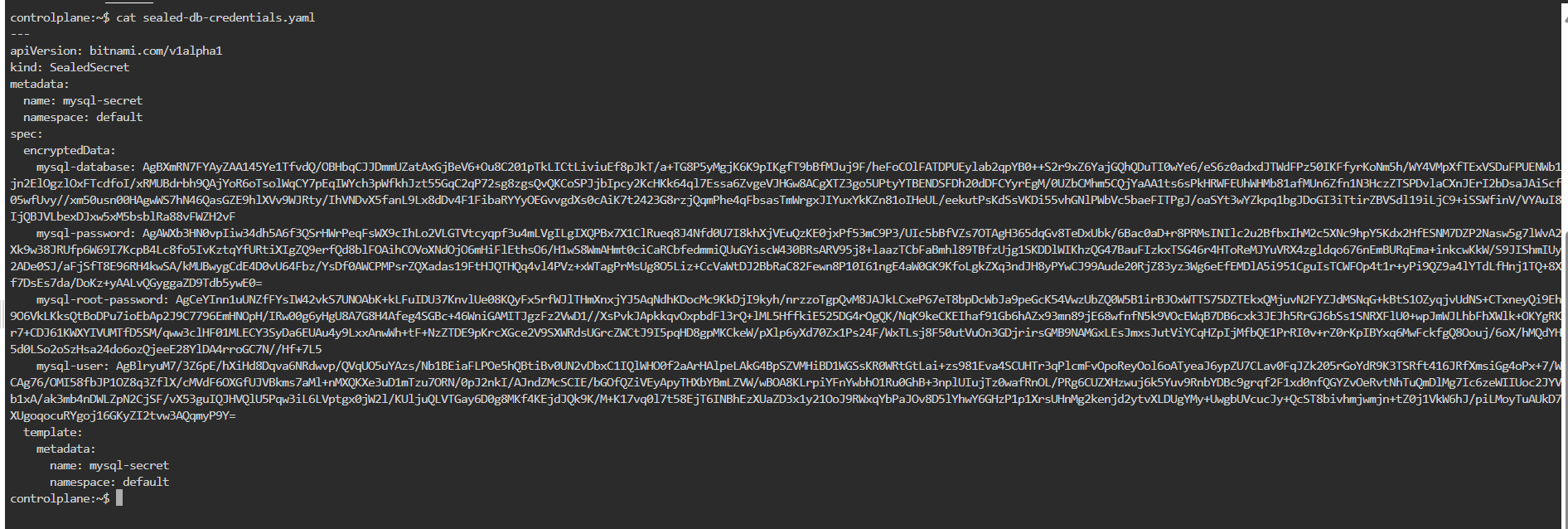
kubectl create secret generic mysql-secret \  
 --from-literal=mysql-root-password='mysqlrootpass' \  
 --from-literal=mysql-user='mysqluser' \  
 --from-literal=mysql-password='mysqlpass' \  
 --from-literal=mysql-database='mysqldb' \  
 -n default --dry-run=client -o yaml > db-credentials.yaml



**Seal the Secret**

kubeseal --format=yaml --cert=public-cert.pem \  
 < db-credentials.yaml > sealed-db-credentials.yaml





**Save and Commit**

Commit the SealedSecret file (sealed-db-credentials.yaml) to your Git repository safely.

**Apply to Cluster**

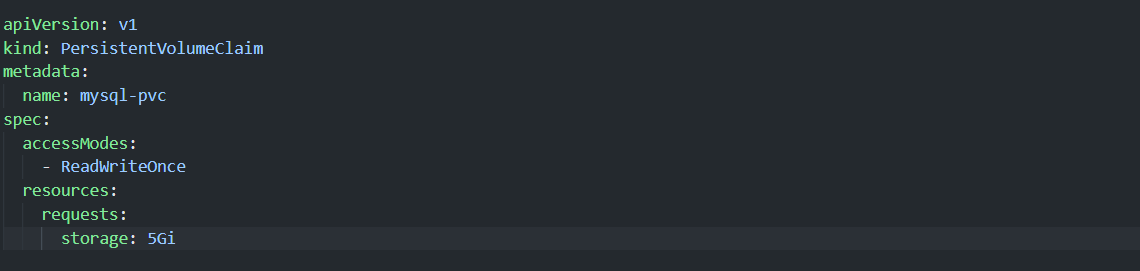
kubectl apply -f sealed-db-credentials.yaml



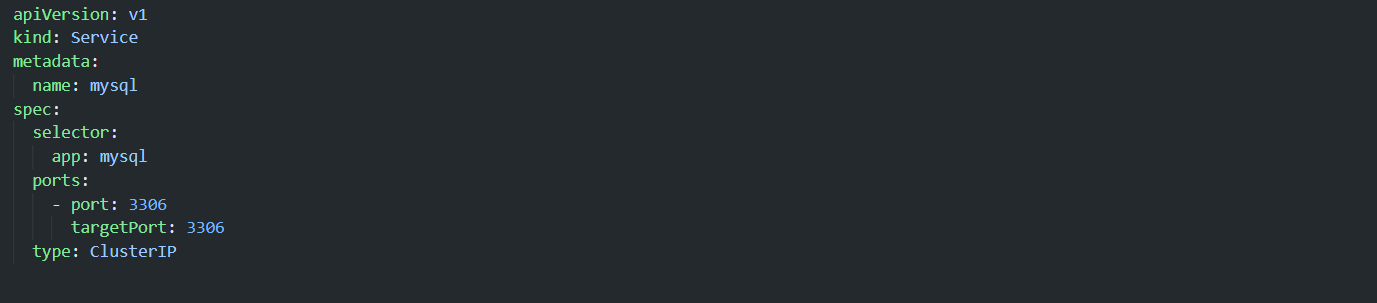
The Sealed Secrets Controller will automatically decrypt the sealed data and create a Kubernetes Secret inside the cluster.

**Created PVC and MySQL Deployment:**

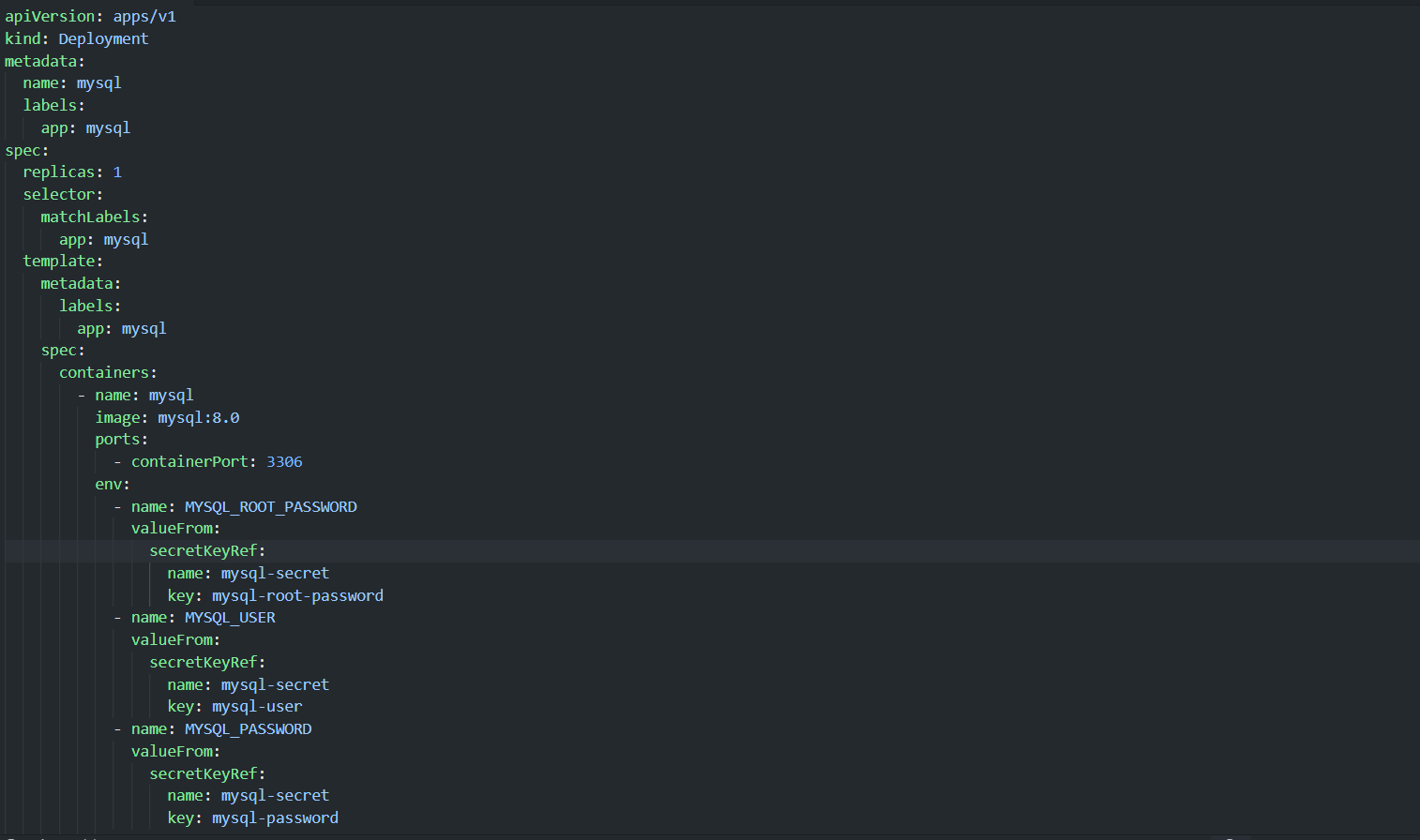
kubectl apply -f pvc.yaml

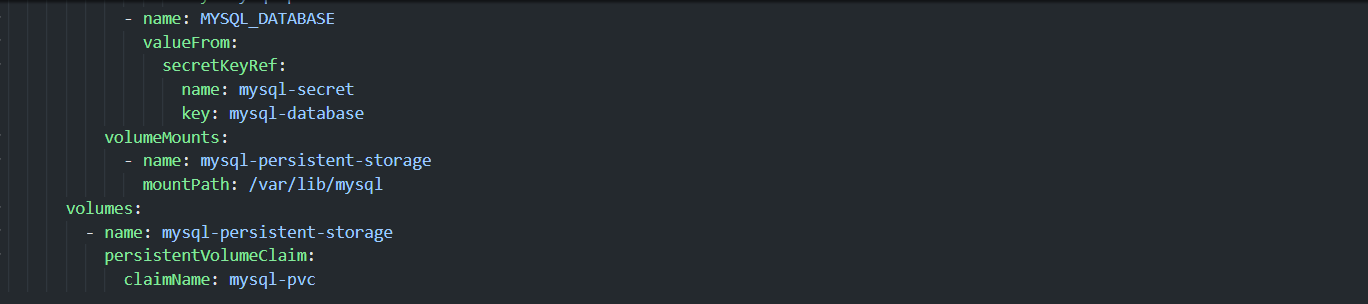


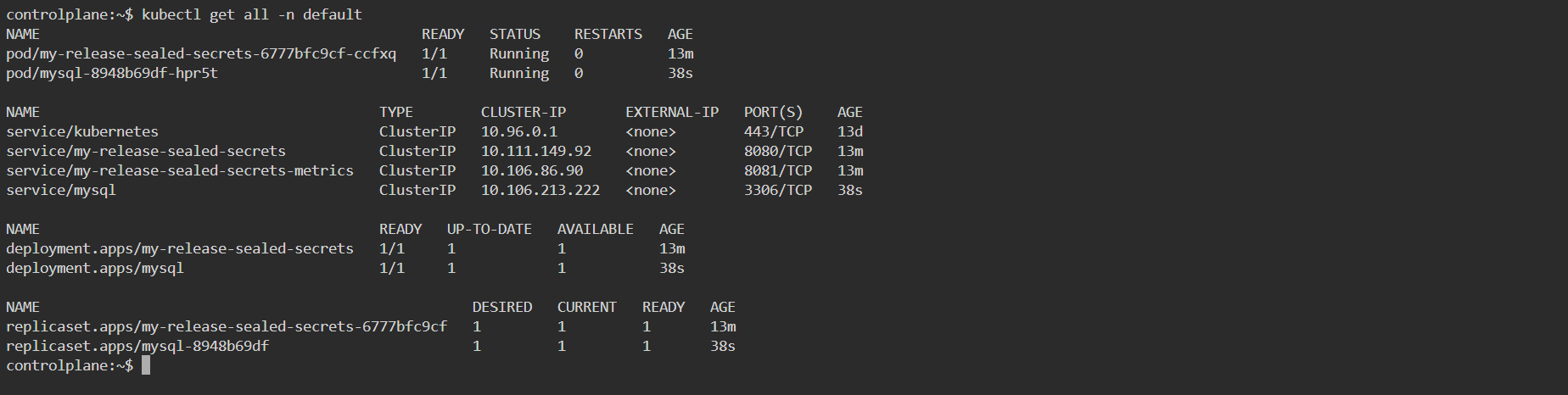
kubectl apply -f svc.yaml



kubectl apply -f deployment.yaml



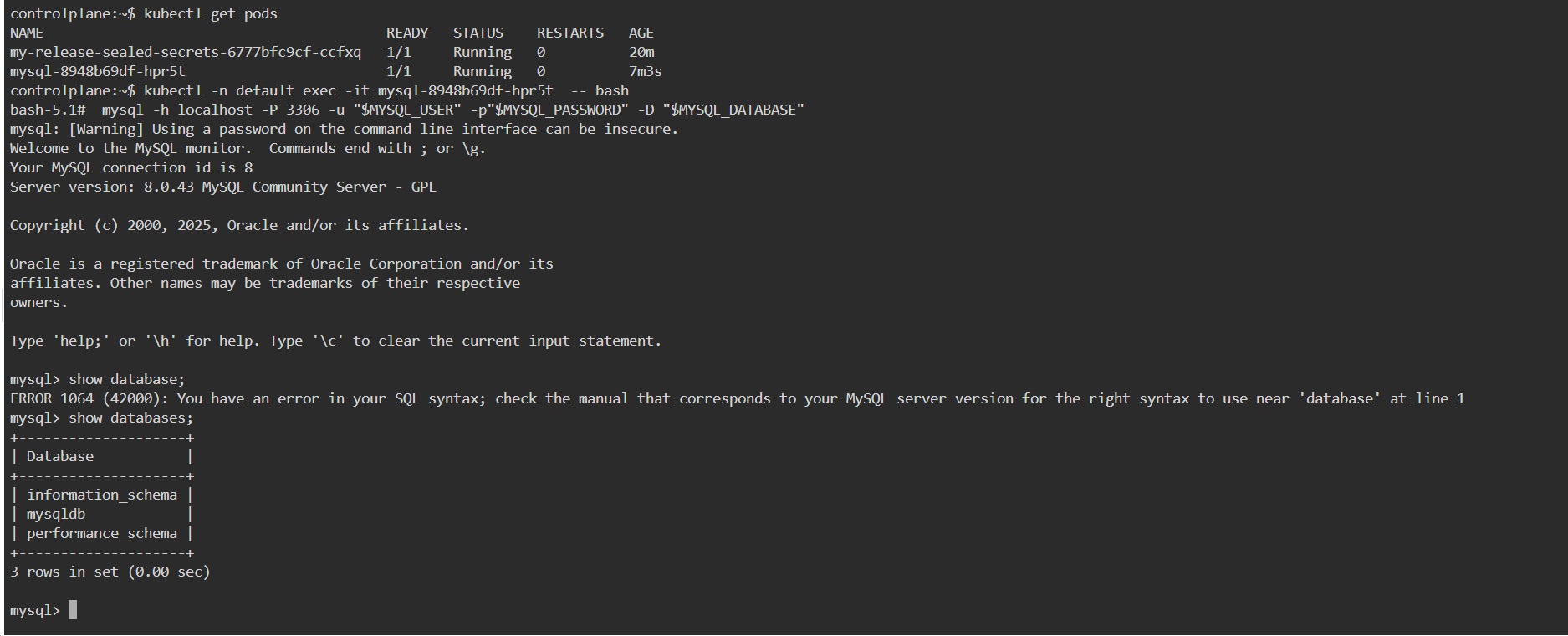




kubectl get pods

kubectl -n default exec -it mysql-8948b69df-hpr5t -- bash

mysql -h localhost -P 3306 -u "$MYSQL\_USER" -p"$MYSQL\_PASSWORD" -D "$MYSQL\_DATABASE"



**Benefits**

* **GitOps friendly:** You can store encrypted Secrets in Git without risk.
* **Cluster-specific decryption:** Only the intended cluster can decrypt SealedSecrets.
* **Easy integration:** Works seamlessly with Helm, ArgoCD, Flux, and Terraform.
* **Automation:** Secret rotation and management can be automated securely.