## **MySQL Deployment on Kubernetes**

To successfully deploy a MySQL instance on Kubernetes, create a series of [YAML files](https://phoenixnap.com/blog/what-is-yaml-with-examples" \t "https://phoenixnap.com/kb/_blank) that you will use to define the following Kubernetes objects:

* A [Kubernetes secret](https://phoenixnap.com/kb/kubernetes-secrets" \t "https://phoenixnap.com/kb/_blank) for storing the database password.
* A [Persistent Volume](https://phoenixnap.com/kb/kubernetes-persistent-volumes" \t "https://phoenixnap.com/kb/_blank) (PV) to allocate storage space for the database.
* A Persistent Volume Claim (PVC) that will claim the PV for the deployment .
* The deployment itself.
* The Kubernetes Service.

### **Step 1: Create Kubernetes Secret**

Use a [text editor such as Nano](https://phoenixnap.com/kb/use-nano-text-editor-commands-linux" \t "https://phoenixnap.com/kb/_blank) to create the Secret file.

nano mysql-secret.yaml

The file defines the secret. Enter the password for the root MySQL account in the **stringData** section of the YAML.

apiVersion: v1

kind: Secret

metadata:

name: mysql-secret

type: kubernetes.io/basic-auth

stringData:

password: test1234

Save the file and exit. [Use kubectl](https://phoenixnap.com/kb/kubectl-commands-cheat-sheet" \t "https://phoenixnap.com/kb/_blank) to apply the changes to the cluster.

kubectl apply -f mysql-secret.yaml

The system confirms the successful creation of the secret:

### **Step 2: Create Persistent Volume and Volume Claim**

Create the storage configuration file:

nano mysql-storage.yaml

This file consists of two parts:

* The first part defines the Persistent Volume. Customize the amount of allocated storage in **spec.capacity.storage**. In **spec.hostPath** specify the volume mount point.
* The second part of the file defines the PVC.

apiVersion: v1

kind: PersistentVolume

metadata:

name: mysql-pv-volume

labels:

type: local

spec:

storageClassName: manual

capacity:

storage: 20Gi

accessModes:

- ReadWriteOnce

hostPath:

path: "/mnt/data"

---

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: mysql-pv-claim

spec:

storageClassName: manual

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 20Gi

Save the file and exit.

Then, apply the storage configuration with **kubectl**.

kubectl apply -f mysql-storage.yaml

The system confirms the creation of the PV and the PVC.

### **Step 3: Create MySQL Deployment**

1. Create the deployment file. The deployment file defines the resources the MySQL deployment will use.

nano mysql-deployment.yaml

2. In the **spec.template.spec.containers** section, specify the MySQL image:

containers:

- image: mysql:5.6

name: mysql

3. Assign the value of the **MYSQL\_ROOT\_PASSWORD** environment variable to the password you specified in the Secret from ****Step 1****.

env:

- name: MYSQL\_ROOT\_PASSWORD

valueFrom:

secretKeyRef:

name: mysql-secret

key: password

4. Connect the PVC from ****Step 2**** to the deployment.

volumes:

- name: mysql-persistent-storage

persistentVolumeClaim:

claimName: mysql-pv-claim

5. In the separate section of the file, define the service name and port.

The entire YAML should look like in the example below:

apiVersion: apps/v1

kind: Deployment

metadata:

name: mysql

spec:

selector:

matchLabels:

app: mysql

strategy:

type: Recreate

template:

metadata:

labels:

app: mysql

spec:

containers:

- image: mysql:5.6

name: mysql

env:

- name: MYSQL\_ROOT\_PASSWORD

valueFrom:

secretKeyRef:

name: mysql-secret

key: password

ports:

- containerPort: 3306

name: mysql

volumeMounts:

- name: mysql-persistent-storage

mountPath: /var/lib/mysql

volumes:

- name: mysql-persistent-storage

persistentVolumeClaim:

claimName: mysql-pv-claim

---

apiVersion: v1

kind: Service

metadata:

name: mysql

spec:

ports:

- port: 3306

selector:

app: mysql

Save the file and exit. Create the deployment by applying the file with **kubectl**:

kubectl apply -f mysql-deployment.yaml

The system confirms the successful creation of both the deployment and the service.

## **Access Your MySQL Instance**

To access the MySQL instance, access the pod created by the deployment.

1. List the pods:

kubectl get pod

2. Find the MySQL pod and copy its name by selecting it and pressing ****Ctrl+Shift+C****:

3. Get a shell for the pod by executing the following command:

kubectl exec --stdin --tty mysql-694d95668d-w7lv5 -- /bin/bash

The pod shell replaces the main shell:

4. Type the following command to access the MySQL shell:

mysql -p

5. When prompted, enter the password you defined in the Kubernetes secret.

The MySQL shell appears.

## **Update Your MySQL Deployment**

Edit the relevant YAML file to update any part of the deployment. Apply the changes with:

kubectl apply -f [filename]

However, bear in mind the following two limitations:

* This particular deployment is for ****single-instance**** MySQL deployment. It means that the deployment cannot be scaled - it works on exactly one Pod.
* This deployment does not support [rolling updates](https://phoenixnap.com/kb/kubernetes-rolling-update" \t "https://phoenixnap.com/kb/_blank). Therefore, the **spec.strategy.type** must always be set to ****Recreate****.

## **Delete Your MySQL Instance**

If you wish to remove the entire deployment, use **kubectl** to delete each of the Kubernetes objects related to it:

kubectl delete deployment,svc mysql

kubectl delete pvc mysql-pv-claim

kubectl delete pv mysql-pv-volume

kubectl delete secret mysql-secret

This series of commands delete the deployment, the service, PV, PVC, and the secret you created.