Deploying MySQL on Kubernetes {Guide}

September 9, 2021

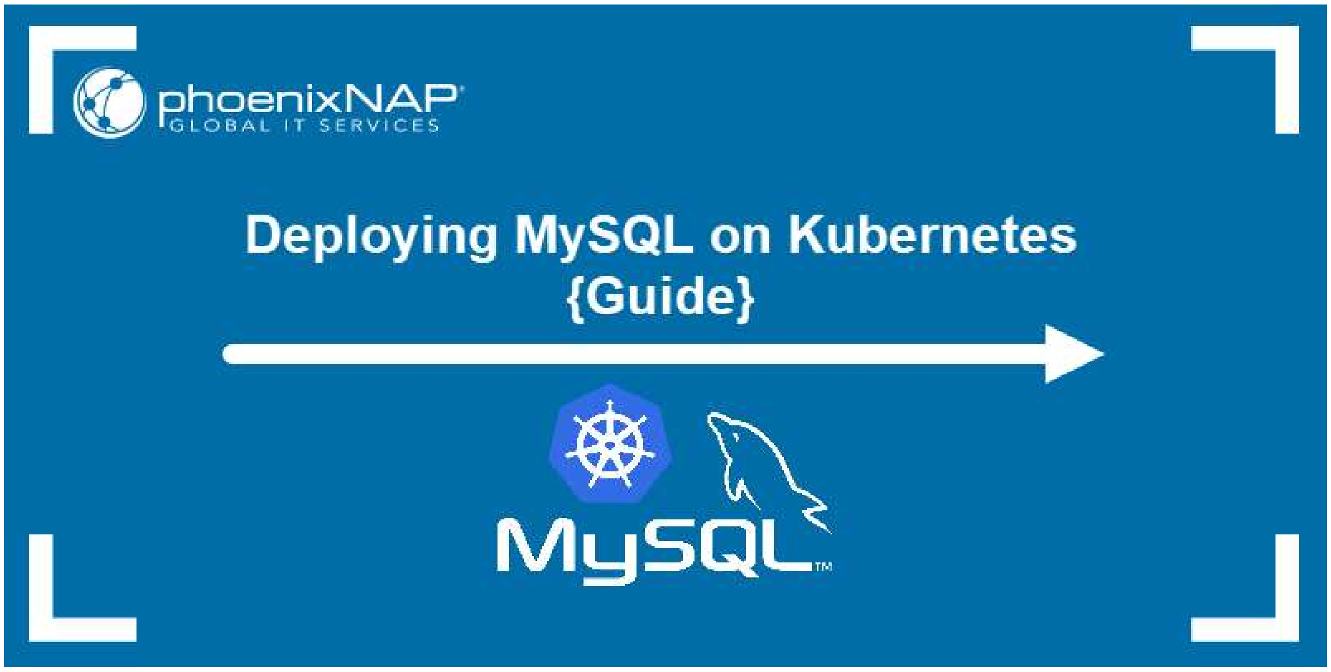
KUBERNETES MYSQL

Home » Databases » Deploying MySQL on Kubernetes {Guide}

Introduction

Deploying a functional database in a containerized environment can be a demanding task. The reason lies in the set of specific challenges databases present in maintaining their availability, state, and redundancy. However, many developers prefer to have their application stacks and data layers organized on the same fast deployment and automation principles offered by platforms such as Kubernetes.

This article will show you how to deploy a MySQL database instance on Kubernetes using persistent volumes. This feature enables stateful apps to overcome the inherent transience of the K8s pods.


phoenixNAP GLOBAL IT SERVICES
Deploying MySQL on Kubernetes {Guide}
MySQL

Prerequisites

- A Kubernetes cluster with kubectl installed
- Administrative access to your system

Note: This tutorial covers the deployment of a single-instance MySQL database. Clustered stateful apps require the creation of StatefulSet objects.

MySQL Deployment on Kubernetes

To successfully deploy a MySQL instance on Kubernetes, create a series of YAML files that you will use to define the following Kubernetes objects:

- A Kubernetes secret for storing the database password.
- A Persistent Volume (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 to create the Secret file.

```
nano mysql-secret.yaml
```

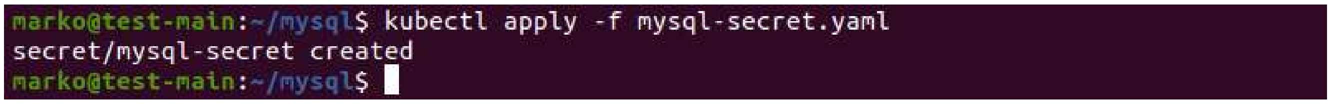
The file defines the secret. Enter the password for the root MySQL account in the **`stringData`** section of the YAML.

```
apiVersion: v1
kind: Secret
metadata:
  name: mysql-secret
type: kubernetes.io/basic-auth
stringData:
  password: test1234
```

Save the file and exit. Use kubectl to apply the changes to the cluster.

```
kubectl apply -f mysql-secret.yaml
```

The system confirms the successful creation of the secret:



```
marko@test-main:~/mysql$ kubectl apply -f mysql-secret.yaml
secret/mysql-secret created
marko@test-main:~/mysql$
```

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.

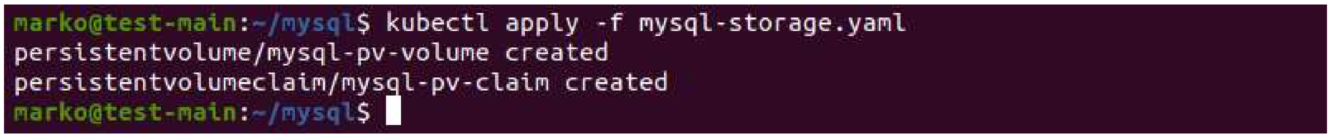
```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mysql-pv-volume
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 20Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/mnt/data"
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-pv-claim
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.



```
marko@test-main:~/mysql$ kubectl apply -f mysql-storage.yaml
persistentvolume/mysql-pv-volume created
persistentvolumeclaim/mysql-pv-claim created
marko@test-main:~/mysql$
```

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:

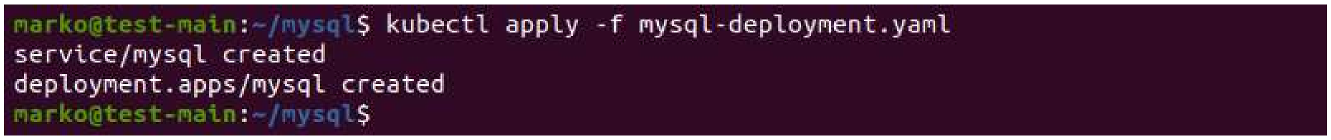
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
spec:
  selector:
    matchLabels:
      app: mysql
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
      - image: mysql:5.6
        name: mysql
        env:
        - 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.



```
marko@test-main:~/mysql$ kubectl apply -f mysql-deployment.yaml
service/mysql created
deployment.apps/mysql created
marko@test-main:~/mysql$
```

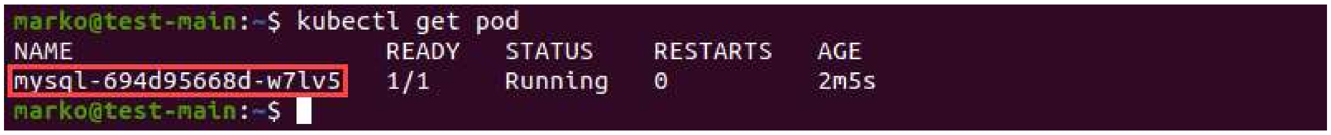
Access Your MySQL Instance

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**:

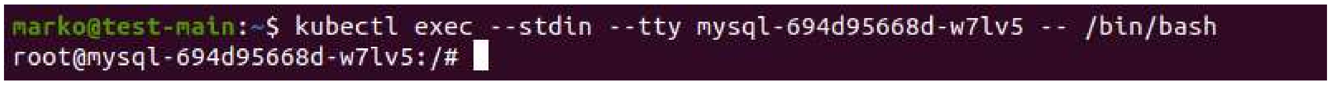


```
marko@test-main:~$ kubectl get pod
NAME                     READY   STATUS    RESTARTS   AGE
mysql-694d95668d-w7lv5   1/1     Running   0          2m5s
marko@test-main:~$
```

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:



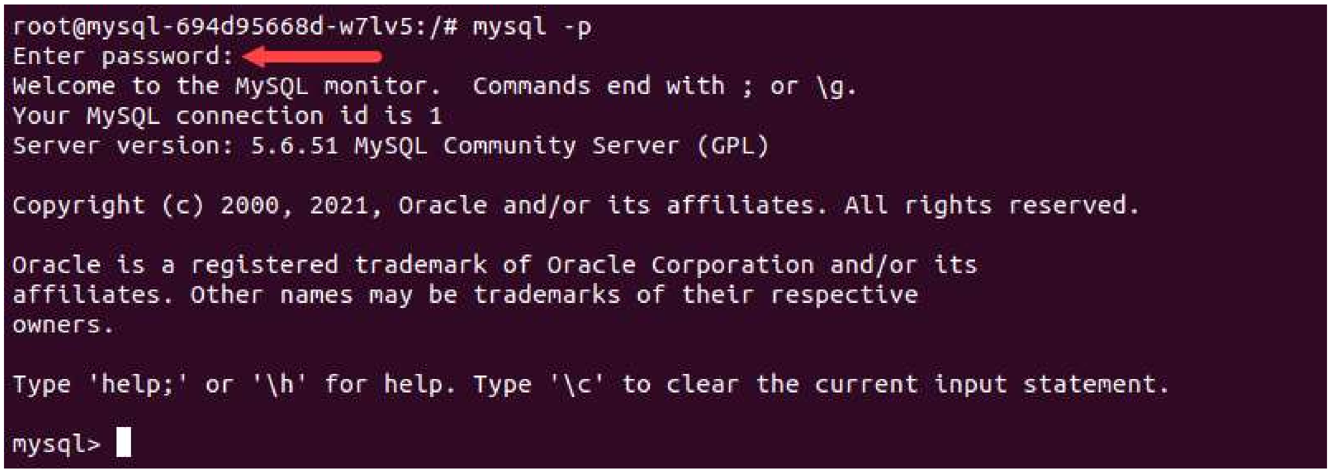
```
marko@test-main:~$ kubectl exec --stdin --tty mysql-694d95668d-w7lv5 -- /bin/bash
root@mysql-694d95668d-w7lv5:/#
```

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.



```
root@mysql-694d95668d-w7lv5:/# mysql -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.6.51 MySQL Community Server (GPL)

Copyright (c) 2000, 2021, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

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. 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. The system confirms the successful deletion:

Conclusion

After completing this tutorial, you should be able to deploy a single MySQL instance on Kubernetes. For an exhaustive list of important MySQL commands, see this MySQL Commands Cheat Sheet.

Was this article helpful? Yes No

Marko Aleksic

Marko Aleksić is a Technical Writer at phoenixNAP. His innate curiosity regarding all things IT, combined with over a decade long background in writing, teaching and working in IT-related fields, led him to technical writing, where he has an opportunity to employ his skills and make technology less daunting to everyone.

Next you should read

MySQL, SysAdmin

MySQL Commands Cheat Sheet

January 20, 2021

To create, modify, and work with relational databases, you need to run the appropriate SQL commands. In this tutorial, you will find the most important MySQL commands as well as a downloadable cheat sheet.

READ MORE

MySQL, SysAdmin

MySQL Performance Tuning and Optimization Tips

January 15, 2020

When using MySQL with large applications, the large amount of data can lead to performance issues. This guide provides tuning tips on how to improve the performance of a MySQL database.

READ MORE

MySQL, SysAdmin

How to Import and Export MySQL Databases in Linux

April 25, 2019

MySQL can be used for something as simple as a product database, or as complex as a WordPress website. This tutorial shows you how to export a MySQL database and import it from a dump file.

READ MORE

Databases, MySQL, SysAdmin

How to Install MySQL on Ubuntu 20.04

June 3, 2021

MySQL is a fast, simple, and scalable SQL-based system, and it represents an integral component of the LAMP stack that runs much of the Internet. In this tutorial, you will learn how to install MySQL on Ubuntu 20.04.

READ MORE