Work With MySQL/MariaDB on Project Astra

Project Astra

Erika Barcott, Ben Cammett
October 02, 2020

NetApp®

Table of Contents

Work With MySQL/MariaDB on Project Astra 1
- Requirements 1
- Install MariaDB/MySQL 2
- Dry Run 3

Work With MySQL/MariaDB on Project Astra

This guide focuses on Helm as the preferred way to deploy Postgres apps. Plain YAML and Operator-based deployments may be covered in future guides.

For express instructions on launching MySQL/MariaDB on Project Astra, see Deploy MySQL/MariaDB from a Helm Chart.

MariaDB and MySQL are validated apps for the Project Astra Beta program. Learn the difference between Validated and Standard apps

MySQL 0.3.22 and MariaDB 14.14 are the only versions supported in the Project Astra Beta program.

Requirements

In order to deploy MySQL/MariaDB from a Helm chart on a cluster registered with the Project Astra Beta program, you will need the following:

GKE Cluster

An up-to-date Kubernetes cluster (version 1.17+) which is connected to Project Astra. For help creating your GKE cluster and connecting it to Project Astra, see the Getting Started Guide.

Kubectl

Kubectl is a standard tool for interacting with Kubernetes. For more information, see the guide Install and Set Up kubectl in the official Kubernetes documentation.

Kubeconfig

The Kubeconfig file contains the credentials which let kubectl communicate with your Kubernetes cluster. to learn how to download your GKE Kubeconfig file, see the Google Cloud guide for configuring cluster access for kubectl.

Cloud Volume Serivice in Google Cloud Platform (CVS-GCP)

CVS is the storage layer and connective elements for Project Astra, respectively. More details on how to configure CVS on GCP may be found in the workflow guide for CVS^.

Helm (v3)

Helm is a popular way to organize and install apps on Kubernetes. To install Helm on your local computer, follow their handy install guide.

MySQL/MariaDB Requirements

For a MySQL/MariaDB application, Project Astra requires:

- `global.storageClass` value to be set to the storageClass representing either CVS or Trident (or, that storageClass is set as your cluster's default provisioner).

Install MariaDB/MySQL

For the Project Astra beta, we recommend the custom Helm chart we have created for this purpose. For instructions on how to deploy from this custom chart, see Deploy MySQL/MariaDB from a Helm Chart.

The values need to be set to consume the volumes provisioned by CVS, be deployed in a namespace other than default, and your stateful app needs to be available to Project Astra.

By default the Bitnami chart uses a cluster's default storage class. Kubernetes clusters registered with project Astra beta use Trident CSI fron NetApp. Trident automatically sets CVS as the default storage class. Use `kubectl get sc` to see what your cluster's storageClasses are. This produces output like the following:

```
NAME                          PROVISIONER            RECLAIMPOLICY   VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION   AGE
netapp-cvs-extreme            csi.trident.netapp.io   Delete          Immediate
true                   26h
netapp-cvs-premium (default)   csi.trident.netapp.io   Delete          Immediate
true                   26h
netapp-cvs-standard           csi.trident.netapp.io   Delete          Immediate
true                   26h
standard                      kubernetes.io/gce-pd    Delete          Immediate
true                   27h
```

You have two options for changing settings in your `values.yaml`. The first option is to open the file and edit it directly. The second option is to add an extra argument to your usual Helm CLI command.

To view and export `values.yaml`, use the `helm show` command:

```
# mariaDB
helm show values bitnami/mariadb
# mySQL
helm show values bitnami/mysql
```

or

```
# mariaDB
helm show values bitnami/mariadb > my-values.yaml
# mySQL
helm show values bitnami/mysql > my-values.yaml
```

This creates a `my-values.yaml` file in your local directory. That file is a copy of the official `values.yaml`.

Dry Run

Before deploying, you can do a dry run to make sure everything is set up correctly.

To do this, edit the values in the `my-values.yaml` file you created in the previous step. Test your deployment using the `-f my-values.yaml` and `--dry-run` flags:

```
# MariaDB
helm install -f my-values.yaml --namespace testdb --generate-name bitnami/mariadb --dry
-run

# MySQL
helm install -f my-values.yaml --namespace testdb --generate-name bitnami/mysql --dry-run
```

If the output from our dry run looks correct, we may deploy to your cluster by removing `--dry-run`.

Before we can run the helm charts for real, you can choose to use an existing namespace or specify to create a new namespace with helm command like below;

```
# MariaDB
helm install -f my-values.yaml --namespace testdb --generate-name bitnami/mariadb
--create-namespace

# MySQL
 helm install -f my-values.yaml --namespace testdb --generate-name bitnami/mysql --create
-namespace
```

After deploying the application using Helm chart Project Astra will be automatically discover the application. After a successful discovery you can manage the app with Project Astra.