MySQL Operator for Kubernetes
Abstract

MySQL Operator for Kubernetes manages MySQL InnoDB Cluster setups inside a Kubernetes Cluster. MySQL Operator for Kubernetes manages the full lifecycle with setup and maintenance including automating upgrades and backups.

For notes detailing the changes in each release, see the MySQL Operator Release Notes.

For legal information, see the Legal Notices.

For help with using MySQL, please visit the MySQL Forums, where you can discuss your issues with other MySQL users.

Document generated on: 2023-08-08 (revision: 76425)
# Table of Contents

Preface and Legal Notices ........................................................................................................... v
1 Introduction ................................................................................................................................. 1
2 Installing MySQL Operator for Kubernetes .................................................................................. 5
   2.1 Install using Helm Charts ....................................................................................................... 5
   2.2 Install using Manifest Files ................................................................................................... 5
3 MySQL InnoDB Cluster ................................................................................................................. 7
   3.1 Deploy using Helm ................................................................................................................. 7
   3.2 Deploy using kubectl ............................................................................................................ 8
   3.3 Manifest Changes for InnoDBCluster .................................................................................... 9
   3.4 MySQL InnoDB Cluster Service Explanation ....................................................................... 11
   3.5 MySQL Accounts Created by InnoDBCluster Deployment .................................................. 13
4 Upgrading MySQL Operator ........................................................................................................ 15
5 Connecting to MySQL InnoDB Cluster ....................................................................................... 17
   5.1 Connect with MySQL Shell .................................................................................................... 17
   5.2 Connect with Port Forwarding ............................................................................................... 18
6 Private Registries .......................................................................................................................... 21
   6.1 Install MySQL Operator for Kubernetes from Private Registry using Helm ......................... 21
   6.2 Install InnoDB Cluster from Private Registry using Helm .................................................... 22
   6.3 Copy Image to Private Registry using Docker ...................................................................... 22
   6.4 Copy Image to Private Registry using Skopeo ....................................................................... 23
7 MySQL Operator Cookbook ........................................................................................................... 25
   7.1 Handling MySQL Backups ..................................................................................................... 25
   7.2 Bootstrap a MySQL InnoDB Cluster from a Dump using Helm ............................................. 28
   7.3 Viewing Logs ......................................................................................................................... 29
8 MySQL Operator Custom Resource Properties ............................................................................. 35
Preface and Legal Notices

MySQL Operator for Kubernetes manages MySQL InnoDB Cluster setups inside a Kubernetes Cluster. MySQL Operator for Kubernetes manages the full lifecycle with setup and maintenance including automating upgrades and backups. This is the MySQL Operator for Kubernetes manual.

Licensing information. This product may include third-party software, used under license. If you are using a Commercial release of MySQL Operator for Kubernetes, see the MySQL Operator for Kubernetes 8.0 Commercial License Information User Manual or MySQL Operator for Kubernetes 8.1 Commercial License Information User Manual for licensing information, including licensing information relating to third-party software that may be included in this Commercial release. If you are using a Community release of MySQL Operator for Kubernetes, see the MySQL Operator for Kubernetes 8.0 Community License Information User Manual or MySQL Operator for Kubernetes 8.1 Community License Information User Manual for licensing information, including licensing information relating to third-party software that may be included in this Community release.

Legal Notices

Copyright © 2009, 2023, Oracle and/or its affiliates.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are “commercial computer software” or “commercial computer software documentation” pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government’s use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle, Java, and MySQL are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.
Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

This documentation is NOT distributed under a GPL license. Use of this documentation is subject to the following terms:

You may create a printed copy of this documentation solely for your own personal use. Conversion to other formats is allowed as long as the actual content is not altered or edited in any way. You shall not publish or distribute this documentation in any form or on any media, except if you distribute the documentation in a manner similar to how Oracle disseminates it (that is, electronically for download on a Web site with the software) or on a CD-ROM or similar medium, provided however that the documentation is disseminated together with the software on the same medium. Any other use, such as any dissemination of printed copies or use of this documentation, in whole or in part, in another publication, requires the prior written consent from an authorized representative of Oracle. Oracle and/or its affiliates reserve any and all rights to this documentation not expressly granted above.

**Access to Oracle Support for Accessibility**

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit

https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.
Chapter 1 Introduction

MySQL and Kubernetes share terminology. For example, a Node might be a Kubernetes Node or a MySQL Node, a Cluster might be a MySQL InnoDB Cluster or Kubernetes Cluster, and a ReplicaSet is a feature in both MySQL and Kubernetes. This documentation prefers the long names but these overloaded terms may still lead to confusion; context is important.

Kubernetes

The Kubernetes system uses Controllers to manage the life-cycle of containerized workloads by running them as Pods in the Kubernetes system. Controllers are general-purpose tools that provide capabilities for a broad range of services, but complex services require additional components and this includes operators. An Operator is software running inside the Kubernetes cluster, and the operator interacts with the Kubernetes API to observe resources and services to assist Kubernetes with the life-cycle management.

MySQL Operator for Kubernetes

The MySQL Operator for Kubernetes is an operator focused on managing one or more MySQL InnoDB Clusters consisting of a group of MySQL Servers and MySQL Routers. The MySQL Operator itself runs in a Kubernetes cluster and is controlled by a Kubernetes Deployment to ensure that the MySQL Operator remains available and running.

The MySQL Operator is deployed in the 'mysql-operator' Kubernetes namespace by default; and watches all InnoDB Clusters and related resources in the Kubernetes cluster. To perform these tasks, the operator subscribes to the Kubernetes API server to update events and connects to the managed MySQL Server instance as needed. On top of the Kubernetes controllers, the operator configures the MySQL servers, replication using MySQL Group Replication, and MySQL Router.

MySQL InnoDB Cluster

Once an InnoDB Cluster (InnoDBCluster) resource is deployed to the Kubernetes API Server, MySQL Operator for Kubernetes creates resources including:

- A Kubernetes StatefulSet for the MySQL Server instances.
  This manages the Pods and assigns the corresponding storage Volume. Each Pod managed by this StatefulSet runs multiple containers. Several provide a sequence of initialisation steps for preparing the MySQL Server configuration and data directory, and then two containers remain active for operational mode. One of those containers (named 'mysql') runs the MySQL Server itself, and the other (named 'sidecar') is a Kubernetes sidecar responsible for local management of the node in coordination with the operator itself.

- A Kubernetes Deployment for the MySQL Routers.
  MySQL Routers are stateless services routing the application to the current Primary or a Replica, depending on the application's choice. The operator can scale the number of routers up or down as required by the Cluster's workload.

A MySQL InnoDB Cluster deployment creates these Kubernetes Services:

- One service is the name of the InnoDB Cluster. It serves as primary entry point for an application and sends incoming connections to the MySQL Router. They provide stable name in the form '{clustername}.svc.cluster.local' and expose specific ports.
See also Section 3.4, “MySQL InnoDB Cluster Service Explanation” and Chapter 5, *Connecting to MySQL InnoDB Cluster*.

- A second service named '{clusternamereg}-instances' provides stable names to the individual servers. Typically these should not be directly used; instead use the main service to reliably reach the current primary or secondary as needed. However, for maintenance or monitoring purposes, direct access to an instance might be needed. Each pod instance has MySQL Shell installed.

MySQL Operator for Kubernetes creates and manages additional resources that should not be manually modified, including:

- A Kubernetes ConfigMap named '{clusternamereg}-initconf' that contains configuration information for the MySQL Servers.

To modify the generated `my.cnf` configuration file, see Section 3.3, “Manifest Changes for InnoDBCluster”.

- A sequence of Kubernetes Secrets with credentials for different parts of the system; names include '{clusternamereg}.backup', '{clusternamereg.privsecrets}', and '{clusternamereg.router}'.

For a list of MySQL accounts (and associated Secrets) created by the operator, see Section 3.5, “MySQL Accounts Created by InnoDBCluster Deployment”.
MySQL Operator for Kubernetes Architecture

Figure 1.1 MySQL Operator for Kubernetes Architecture Diagram
Chapter 2 Installing MySQL Operator for Kubernetes

Table of Contents

2.1 Install using Helm Charts ................................................................................................................................. 5
2.2 Install using Manifest Files ................................................................................................................................. 5

Two different installation methods are documented here; using either helm or manually applying manifests using kubectl. This documentation assumes that kubectl is available on a system configured with the desired Kubernetes context; and all examples use a Unix-like command line.

MySQL Operator for Kubernetes functions with Kubernetes 1.21 and newer.

Note
MySQL Operator for Kubernetes requires these three container images to function:
MySQL Operator for Kubernetes, MySQL Router, and MySQL Server.

2.1 Install using Helm Charts

Helm is an optional package manager for Kubernetes that helps manage Kubernetes applications; Helm uses charts to define, install, and upgrade Kubernetes Operators. For Helm specific usage information, see the Helm Quickstart and Installing Helm guides. Alternatively, see Section 2.2, “Install using Manifest Files”.

Add the Helm repository:

```bash
$> helm repo add mysql-operator https://mysql.github.io/mysql-operator/
$> helm repo update
```

Install MySQL Operator for Kubernetes, this simple example defines the release name as my-mysql-operator using a new namespace named mysql-operator:

```bash
$> helm install my-mysql-operator mysql-operator/mysql-operator \
     --namespace mysql-operator --create-namespace
```

The operator deployment is customizable through other options that override built-in defaults. For example, useful when using a local (air-gapped) private container registry to use with the operator.

To use MySQL Operator for Kubernetes to create MySQL InnoDB Clusters, see Chapter 3, MySQL InnoDB Cluster.

2.2 Install using Manifest Files

This document assumes a familiarity with kubectl, and that you have it installed. Alternatively, see Section 2.1, “Install using Helm Charts”.

MySQL Operator for Kubernetes can be installed using raw manifest files with kubectl.

Note
Using trunk in the URL references the latest MySQL Operator for Kubernetes release because Github is updated at release time. Alternatively, replace trunk in the URL with a specific tagged released version.
First install the Custom Resource Definition (CRD) used by MySQL Operator for Kubernetes:

```bash
$> kubectl apply -f https://raw.githubusercontent.com/mysql/mysql-operator/trunk/deploy/deploy-crds.yaml
// Output is similar to:
customresourcedefinition.apiextensions.k8s.io/innodbclusters.mysql.oracle.com created
customresourcedefinition.apiextensions.k8s.io/mysqlbackups.mysql.oracle.com created
customresourcedefinition.apiextensions.k8s.io/clusterkopfpeerings.zalando.org created
customresourcedefinition.apiextensions.k8s.io/kopfpeerings.zalando.org created
```

Next deploy MySQL Operator for Kubernetes, which also includes RBAC definitions as noted in the output:

```bash
$> kubectl apply -f https://raw.githubusercontent.com/mysql/mysql-operator/trunk/deploy/deploy-operator.yaml
// Output is similar to:
clusterrole.rbac.authorization.k8s.io/mysql-operator created
clusterrole.rbac.authorization.k8s.io/mysql-sidecar created
clusterrolebinding.rbac.authorization.k8s.io/mysql-operator-rolebinding created
clusterkopfpeering.zalando.org/mysql-operator created
namespace/mysql-operator created
serviceaccount/mysql-operator-sa created
deployment.apps/mysql-operator created
```

Verify that the operator is running by checking the deployment that's managing the operator inside the mysql-operator namespace, a configurable namespace defined by `deploy-operator.yaml`:

```bash
$> kubectl get deployment mysql-operator --namespace mysql-operator
```

After MySQL Operator for Kubernetes is ready, the output should look similar to this:

```
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
mysql-operator   1/1     1            1           37s
```

To use MySQL Operator for Kubernetes to create MySQL InnoDB Clusters, see Chapter 3, *MySQL InnoDB Cluster*. 
Chapter 3 MySQL InnoDB Cluster

Table of Contents

3.1 Deploy using Helm ................................................................. 7
3.2 Deploy using kubectl .............................................................. 8
3.3 Manifest Changes for InnoDBCluster ......................................... 9
3.4 MySQL InnoDB Cluster Service Explanation ............................... 11
3.5 MySQL Accounts Created by InnoDBCluster Deployment .................. 13

Examples and documentation assumes the current default namespace is used, which defaults to 'default' although it can be modified, for example:

$> kubectl create namespace newdefaultnamespace
$> kubectl config set-context --current --namespace=newdefaultnamespace

Examples typically use ‘innodbcluster’ as the resource name but may use plural and short names as defined in deploy-crds.yaml:

```yaml
names:
  kind: InnoDBCluster
  listKind: InnoDBClusterList
  singular: innodbcluster
  plural: innodbclusters
  shortNames:
    - ic
    - ics
```

3.1 Deploy using Helm

Potential values for creating a MySQL InnoDB Cluster are visible here:

$> helm show values mysql-operator/mysql-innodbcluster

Public Registry

The most common Helm repository is the public https://artifacthub.io/, which is used by these examples.

This example defines credentials in a file named credentials.yaml, sets tls.useSelfSigned=true to avoid setting up SSL, uses the default namespace, and sets mycluster as the cluster's name:

Example credentials.yaml:

```yaml
credentials:
  root:
    user: root
    password: sakila
    host: ":"%
```

$> helm install mycluster mysql-operator/mysql-innodbcluster \
   --set tls.useSelfSigned=true --values credentials.yaml

The manifest for this simple installation looks similar to this:

$> helm get manifest mycluster

---
# Source: mysql-innodbcluster/templates/service_account_cluster.yaml
Deploy using kubectl

```yaml
apiVersion: v1
class: ServiceAccount
metadata:
  name: mycluster-sa
  namespace: default
---
# Source: mysql-innodbcluster/templates/cluster_secret.yaml
apiVersion: v1
class: Secret
metadata:
  name: mycluster-cluster-secret
  namespace: default
stringData:
  rootUser: "root"
  rootHost: "%"
  rootPassword: "sakila"
---
# Source: mysql-innodbcluster/templates/deployment_cluster.yaml
apiVersion: mysql.oracle.com/v2
class: InnoDBCluster
metadata:
  name: mycluster
  namespace: default
spec:
  instances: 3
tlsUseSelfSigned: true
router:
  instances: 1
secretName: mycluster-cluster-secret
imagePullPolicy : IfNotPresent
baseServerId: 1000
version: 8.0.34
serviceAccountName: mycluster-sa
```

Alternatively set options using command-line parameters:

```bash
$> helm install mycluster mysql-operator/mysql-innodbcluster \
   --set credentials.root.user='root' \n   --set credentials.root.password='sakila' \n   --set credentials.root.host='%' \n   --set serverInstances=3 \n   --set routerInstances=1 \n   --set tls.useSelfSigned=true
```

To view user-supplied values for an existing cluster:

```bash
$> helm get values mycluster
USER-SUPPLIED VALUES:
credentials:
  root:
    host: '%'
    password: sakila
    user: root
routerInstances: 1
serverInstances: 3
tls: useSelfSigned: true
```

See also Chapter 5, Connecting to MySQL InnoDB Cluster.

### 3.2 Deploy using kubectl

To create an InnoDB Cluster with kubectl, first create a secret containing credentials for a new MySQL root user, a secret named 'mypwds' in this example:
Use that newly created user to configure a new MySQL InnoDB Cluster. This example's InnoDBCluster definition creates three MySQL server instances and one MySQL Router instance:

```yaml
apiVersion: mysql.oracle.com/v2
kind: InnoDBCluster
metadata:
  name: mycluster
spec:
  secretName: mypwds
tlsUseSelfSigned: true
  instances: 3
  router:
    instances: 1
```

Assuming a file named `mycluster.yaml` contains this definition, install this simple cluster:

```
$> kubectl apply -f mycluster.yaml
```

Optionally observe the process by watching the `innodbcluster` type for the default namespace:

```
$> kubectl get innodbcluster --watch
```

Output looks similar to this:

<table>
<thead>
<tr>
<th>NAME</th>
<th>STATUS</th>
<th>ONLINE</th>
<th>INSTANCES</th>
<th>ROUTERS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>mycluster</td>
<td>PENDING</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>10s</td>
</tr>
</tbody>
</table>

Until reaching ONLINE status:

<table>
<thead>
<tr>
<th>NAME</th>
<th>STATUS</th>
<th>ONLINE</th>
<th>INSTANCES</th>
<th>ROUTERS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>mycluster</td>
<td>ONLINE</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>2m6s</td>
</tr>
</tbody>
</table>

To demonstrate, this example connects with MySQL Shell to show the host name:

```
$> kubectl run --rm -it myshell --image=container-registry.oracle.com/mysql/community-operator -- mysqlsh root@mycluster --sql

If you don’t see a command prompt, try pressing enter.

******

MySQL mycluster SQL> SELECT @@hostname

+-------------+
| @@hostname  |
+-------------+
| mycluster-0  |
```

This shows a successful connection that was routed to the mycluster-0 pod in the MySQL InnoDB Cluster. For additional information about connecting, see Chapter 5, *Connecting to MySQL InnoDB Cluster*.

### 3.3 Manifest Changes for InnoDBCluster

This section covers common options defined while setting up a MySQL InnoDB Cluster. For a full list of options, see Table 8.1, “Spec table for InnoDBCluster”.
Here’s a simple example that uses most defaults:

```yaml
apiVersion: mysql.oracle.com/v2
kind: InnoDBCluster
metadata:
  name: mycluster
spec:
  secretName: mypwds
tlsUseSelfSigned: true
```

Here’s an expanded version of that with optional changes:

```yaml
apiVersion: mysql.oracle.com/v2
kind: InnoDBCluster
metadata:
  name: mycluster
spec:
  secretName: mypwds
tlsUseSelfSigned: true
  instances: 3
  version: 8.0.34
  router:
    instances: 1
    version: 8.0.34
  datadirVolumeClaimTemplate:
    accessModes:
      - ReadWriteOnce
    resources:
      requests:
        storage: 40Gi
    initDB:
      clone:
        donorUrl: mycluster-0.mycluster-instances.another.svc.cluster.local:3306
        rootUser: root
        secretKeyRef:
          name: mypwds
    mycnf:
      [mysqld]
      max_connections=162
```

Below are explanations of each change made to initial the `InnoDBCluster` configuration.

## Router and Server Versions and Instances

By default, MySQL Operator for Kubernetes installs MySQL Server with the same version as the Operator, and installs Router with the same version as MySQL Server. It also installs 3 MySQL instances and 1 Router instance by default. Optionally configure each:

```yaml
spec:
  instances: 3
  version: 8.0.34
  router:
    instances: 1
    version: 8.0.34
```

## Setting PersistentVolumeClaim Size

Set a MySQL instance’s storage configuration. For storing the MySQL Server’s Data Directory (datadir), a PersistentVolumeClaim (PVC) is used for each MySQL Server pod. Each PVC follows the naming scheme `datadir-<clustername>-[0-9]`. A `datadirVolumeClaimTemplate` template allows setting different options, including size and storage class. For example:

```yaml
datadirVolumeClaimTemplate:
```

---

10
The initDB Object

Optionally initialize an InnoDBCluster with a database using the initDB object; it's only used when the InnoDBCluster is created. It accepts clone or dump definitions.

This simple initDB clone example clones a remote MySQL instance from a cluster. The donor MySQL server's credentials are stored in a Secret on the target server with a 'rootPassword' key for the 'rootUser'.

```yaml
initDB:
  clone:
    donorUrl: mycluster-0.mycluster-instances.another.svc.cluster.local:3306
    rootUser: root
    secretKeyRef:
      name: mypwds
```

MySQL Server restarts after populating with the clone operation, and a "1" is seen in the restart column of the associated pods. Cloning utilizes MySQL Server's The Clone Plugin and behaves accordingly.

For a dump example (instead of clone), see Section 7.2, “Bootstrap a MySQL InnoDB Cluster from a Dump using Helm”.

Modify my.cnf Settings

Use the mycnf option to add custom configuration additions to the my.cnf for each MySQL instance. This example adds a [mysqld] section that sets max_connections to 162:

```yaml
mycnf: |
  [mysqld]
  max_connections=162
```

This is added to the generated my.cnf; the default my.cnf template is visible in the initconf container's ConfigMap. An example to see this template: kubectl get cm ${CLUSTER_NAME}-initconf -o json | jq -r '.data["my.cnf.in"]'.

3.4 MySQL InnoDB Cluster Service Explanation

For connecting to the InnoDB Cluster, a Service is created inside the Kubernetes cluster. The exported ports represent read-write and read-only ports for both the MySQL Protocol and X Protocol.

```bash
$> kubectl describe service mycluster
```
MySQL InnoDB Cluster Service Explanation

Output looks similar to this:

<table>
<thead>
<tr>
<th>Name:</th>
<th>mycluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>Namespace:</td>
<td>default</td>
</tr>
<tr>
<td>Labels:</td>
<td>mysql.oracle.com/cluster=mycluster</td>
</tr>
<tr>
<td></td>
<td>tier=mysql</td>
</tr>
<tr>
<td>Annotations:</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>Selector:</td>
<td>component=mysqlrouter,mysql.oracle.com/cluster=mycluster,tier=mysql</td>
</tr>
<tr>
<td>Type:</td>
<td>ClusterIP</td>
</tr>
<tr>
<td>IP Family Policy:</td>
<td>SingleStack</td>
</tr>
<tr>
<td>IP Families:</td>
<td>IPv4</td>
</tr>
<tr>
<td>IP:</td>
<td>10.106.33.215</td>
</tr>
<tr>
<td>IPs:</td>
<td>10.106.33.215</td>
</tr>
<tr>
<td>Port:</td>
<td>mysql 3306/TCP</td>
</tr>
<tr>
<td>TargetPort:</td>
<td>6446/TCP</td>
</tr>
<tr>
<td>Endpoints:</td>
<td>172.17.0.12:6446</td>
</tr>
<tr>
<td>Port:</td>
<td>mysqlx 33060/TCP</td>
</tr>
<tr>
<td>TargetPort:</td>
<td>6448/TCP</td>
</tr>
<tr>
<td>Endpoints:</td>
<td>172.17.0.12:6448</td>
</tr>
<tr>
<td>Port:</td>
<td>mysql-alternate 6446/TCP</td>
</tr>
<tr>
<td>TargetPort:</td>
<td>6448/TCP</td>
</tr>
<tr>
<td>Endpoints:</td>
<td>172.17.0.12:6448</td>
</tr>
<tr>
<td>Port:</td>
<td>mysql-ro 6447/TCP</td>
</tr>
<tr>
<td>TargetPort:</td>
<td>6447/TCP</td>
</tr>
<tr>
<td>Endpoints:</td>
<td>172.17.0.12:6447</td>
</tr>
<tr>
<td>Port:</td>
<td>mysqlx-ro 6449/TCP</td>
</tr>
<tr>
<td>TargetPort:</td>
<td>6449/TCP</td>
</tr>
<tr>
<td>Endpoints:</td>
<td>172.17.0.12:6449</td>
</tr>
<tr>
<td>Port:</td>
<td>router-rest 8443/TCP</td>
</tr>
<tr>
<td>TargetPort:</td>
<td>8443/TCP</td>
</tr>
<tr>
<td>Endpoints:</td>
<td>172.17.0.12:8443</td>
</tr>
<tr>
<td>Session Affinity:</td>
<td>None</td>
</tr>
<tr>
<td>Events:</td>
<td>&lt;none&gt;</td>
</tr>
</tbody>
</table>

An alternative view showing services named **mycluster** and **mycluster-instances**:

```
$> kubectl get service
```

Output looks similar to this:

```
<table>
<thead>
<tr>
<th>NAMESPACE</th>
<th>NAME</th>
<th>TYPE</th>
<th>CLUSTER-IP</th>
<th>EXTERNAL-IP</th>
<th>PORT(S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>kubernetes</td>
<td>ClusterIP</td>
<td>10.96.0.1</td>
<td>&lt;none&gt;</td>
<td>443/TCP</td>
</tr>
<tr>
<td>default</td>
<td>mycluster</td>
<td>ClusterIP</td>
<td>10.102.198.226</td>
<td>&lt;none&gt;</td>
<td>3306/TCP,33060/TCP,6446/TCP,6448/TCP,6449/TCP,8443/TCP,8444/TCP,8445/TCP</td>
</tr>
<tr>
<td>default</td>
<td>mycluster-instances</td>
<td>ClusterIP</td>
<td>None</td>
<td>&lt;none&gt;</td>
<td>3306/TCP,33060/TCP,33061/TCP</td>
</tr>
</tbody>
</table>
```

The long host name used to connect to an InnoDB Cluster from within a Kubernetes cluster is `{innodbclustername}.{namespace}.svc.cluster.local`, which routes to the current primary/replica using MySQL Router, depending on the port. Acceptable host name forms:

```
{innodbclustername}.{namespace}.svc.cluster.local
{innodbclustername}.{namespace}.svc
{innodbclustername}.{namespace}
{innodbclustername}
```

Using these names goes to the Kubernetes LoadBalancer (part of Kubernetes Service), which redirects to MySQL Router. MySQL Router then talks to the individual server based on the role, such as PRIMARY or SECONDARY.

For example, assuming ‘mycluster’ as the InnoDB Cluster name in the ‘default’ namespace:
MySQL Accounts Created by InnoDBCluster Deployment

mycluster.default.svc.cluster.local

Using only {innodbclustername} as the host name assumes the session’s context is either the default namespace or set accordingly. Alternatively you may use the clusterIP instead of a host name; here’s an example that retrieves it:

$> kubectl get service/mycluster -o jsonpath='{.spec.clusterIP}'

See also Chapter 5, Connecting to MySQL InnoDB Cluster.

3.5 MySQL Accounts Created by InnoDBCluster Deployment

MySQL Operator for Kubernetes creates and/or utilizes several MySQL accounts as when creating an InnoDB Cluster. Internal accounts created and only used by MySQL Operator for Kubernetes may be used by users but they must not be changed (dropped, password changes, grant changes, and so on).

Typically the only account a system administrator uses is the ‘root’ user, whereas other MySQL users are considered internal to the MySQL InnoDB Cluster installation.

Table 3.1 MySQL accounts created and/or used by MySQL Operator.

<table>
<thead>
<tr>
<th>MySQL User</th>
<th>Purpose</th>
<th>Creator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td>General system administration by the user</td>
<td>MySQL Operator for Kubernetes as defined by the user</td>
<td>Defined when InnoDB Cluster is created using a user-supplied Kubernetes secret object as referenced by the secretsName configuration option. It's typically root@'%' but can be overridden using the rootUser and rootHost configuration options. You may want to create less-privileged MySQL accounts with this user.</td>
</tr>
<tr>
<td>localroot</td>
<td>Used by Operator to perform local administration tasks</td>
<td>MySQL Operator for Kubernetes</td>
<td>This local root account specific to MySQL Operator for Kubernetes, and is used by the MySQL sidecar container for local maintenance tasks like creating other accounts, configuring instances, and verifying replication status. It should not be used or edited by users. It's created with auth_socket authentication and PROXY with full</td>
</tr>
<tr>
<td>MySQL User</td>
<td>Purpose</td>
<td>Creator</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------------------------------------------</td>
<td>----------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>mysqladmin</td>
<td>Administration tasks by the Operator</td>
<td>MySQL Operator for Kubernetes</td>
<td>Used to administer the InnoDB Cluster, credentials managed by the &quot;{clustername}-privsecrets&quot; Kubernetes secret</td>
</tr>
<tr>
<td>mysqlbackup</td>
<td>Administration tasks by the Operator</td>
<td>MySQL Operator for Kubernetes</td>
<td>Used to create backups and manage backup jobs, credentials managed by the &quot;{clustername}-backup&quot; Kubernetes secret</td>
</tr>
<tr>
<td>mysqlrouter</td>
<td>Administration tasks by the Operator</td>
<td>MySQL Operator for Kubernetes</td>
<td>Tasks include managing MySQL Router instances to access cluster metadata; credentials managed by the &quot;{clustername}-router&quot; Kubernetes secret</td>
</tr>
<tr>
<td>mysqlhealthchecker</td>
<td>Internal health checks</td>
<td>MySQL Operator for Kubernetes</td>
<td>A local account used for health checks only (liveness and readiness probes); created with auth_socket authentication and no privileges.</td>
</tr>
<tr>
<td>mysql_innodb_cluster_{server_id}</td>
<td>Internal recovery users that enable connections between the servers in the cluster</td>
<td>MySQL InnoDB Cluster</td>
<td>One per MySQL instance, for additional information see Internal User Accounts Created by InnoDB Cluster.</td>
</tr>
<tr>
<td>mysql.infoschema</td>
<td>Reserved</td>
<td>MySQL Server</td>
<td>See Reserved Accounts.</td>
</tr>
<tr>
<td>mysql.session</td>
<td>Reserved</td>
<td>MySQL Server</td>
<td>See Reserved Accounts.</td>
</tr>
<tr>
<td>mysql.sys</td>
<td>Reserved</td>
<td>MySQL Server</td>
<td>See Reserved Accounts.</td>
</tr>
</tbody>
</table>

Related: Deploying MySQL Operator for Kubernetes creates a Kubernetes service account with a name defaulting to mysql-operator-sa in the bundled deploy-operator.yaml and Helm deployment template.

For a list of all ports used by MySQL services, see MySQL Port Reference.
Chapter 4 Upgrading MySQL Operator

Upgrading MySQL Operator

Apply MySQL Operator's latest released CRDs to create a new MySQL Operator deployment that replaces the old. The old operator is terminated after the new operator is started and ready, which should not cause downtime.

```
kubectl apply -f https://raw.githubusercontent.com/mysql/mysql-operator/trunk/deploy/deploy-crds.yaml
kubectl apply -f https://raw.githubusercontent.com/mysql/mysql-operator/trunk/deploy/deploy-operator.yaml
```

**Note**

This only updates MySQL Operator and not the associated MySQL InnoDB Cluster.

Using `trunk` in the URL references the latest MySQL Operator for Kubernetes release because Github is updated at release time. Alternatively, replace `trunk` in the URL with a specific tagged released version.

Upgrading MySQL InnoDB Cluster

This assumes you already upgraded MySQL Operator.

A common upgrade method is to patch the MySQL server version. For example, this updates the MySQL Server version to 8.0.34 for a MySQL InnoDB Cluster named `mycluster`:

```
kubectl patch ic mycluster -p '{"spec": { "version": "8.0.34" } }' --type=merge
```

An update to `spec.version` for a MySQL InnoDB Cluster updates the following:

- Each MySQL server, in a rolling update
  
  Upgrading a MySQL InnoDB Cluster initiates a rolling restart of the MySQL servers; the upgrade replaces MySQL servers in order, by highest value in the name to lowest. This can cause multiple failovers of the primary, depending on the current and assigned primaries.
  
- MySQL Router to the specified `spec.version` unless `spec.router.version` is also explicitly set in the patch
  
- The MySQL sidecar container for each server to the installed MySQL Operator version
  
- MySQL Shell to the specified `spec.version`

The MySQL InnoDB Cluster remains available during the upgrade process but the associated restarts may interrupt existing connections.
Chapter 5 Connecting to MySQL InnoDB Cluster

Table of Contents

5.1 Connect with MySQL Shell ................................................................. 17
5.2 Connect with Port Forwarding .......................................................... 18

This section utilizes the {innodbclustername}.{namespace}.svc.cluster.local form when connecting; and typically refers to the {innodbclustername} shorthand form that assumes the default namespace. See Section 3.4, “MySQL InnoDB Cluster Service Explanation” for additional information.

5.1 Connect with MySQL Shell

Create a new container with MySQL Shell to administer a MySQL InnoDB Cluster. This is the preferred method, although every MySQL Operator for Kubernetes and MySQL InnoDB Cluster container also has MySQL Shell installed if you need to troubleshoot a specific pod.

These examples assume the InnoDB Cluster is named 'mycluster' and using the 'default' namespace.

Create the new container with MySQL Shell; this example uses the MySQL Operator for Kubernetes image but other images work too, such as container-registry.oracle.com/mysql/community-server:8.0.

This example creates a new container named "myshell" using a MySQL Operator image, and immediately executes MySQL Shell:

```
$> kubectl run --rm -it myshell --image=container-registry.oracle.com/mysql/community-operator -- mysqlsh
```

If you don't see a command prompt, try pressing enter.

Now connect to the InnoDB Cluster from within MySQL Shell's interface:

```
MySQL JS> \connect root@mycluster
Creating a session to 'root@mycluster'
Please provide the password for 'root@mycluster': ******
MySQL mycluster JS>
```

The root@mycluster shorthand works as it assumes port 3306 (MySQL Router redirects to 6446) and the default namespace.

Optionally pass in additional arguments to mysqlsh, for example:

```
$> kubectl run --rm -it myshell --image=container-registry.oracle.com/mysql/community-operator -- mysqlsh root@mycluster --sql
```

If you don't see a command prompt, try pressing enter.

```
******
MySQL mycluster SQL>
```

The ****** represents entering the MySQL user's password to MySQL Shell as MySQL Shell prompts for a password by default. The root@mycluster represents user root on host mycluster, and assumes the default namespace. Setting "-sql" initiates MySQL Shell into SQL mode.
Troubleshooting a Specific Container

Every MySQL Operator for Kubernetes and MySQL InnoDB Cluster container has MySQL Shell installed, so for troubleshooting you may need to connect to a specific pod in the cluster. For example, connecting to a pod named mycluster-0:

```
$> kubectl --namespace default exec -it mycluster-0 -- bash
Defaulted container "sidecar" out of: sidecar, mysql, initconf (init), initmysql (init)
bash-4.4#
```

```
bash-4.4# mysqlsh root@localhost
Please provide the password for 'root@localhost': ******
...
```

5.2 Connect with Port Forwarding

Optionally use port forwarding to create a redirection from your local machine to easily use a MySQL client such as MySQL Workbench. We’ll use port 3306 for a read-write connection to the primary on port 6446:

```
$> kubectl port-forward service/mycluster 3306
Forwarding from 127.0.0.1:3306 -> 6446
Forwarding from [::1]:3306 -> 6446
```

To test, open a second terminal using the MySQL command line or MySQL Shell with the InnoDB Cluster user’s credentials:

```
$> mysql -h127.0.0.1 -uroot -p
```

To demonstrate the connection to a local MySQL instance:

```
mysql> select @@hostname;
+-------------+
| @@hostname  |
+-------------+
| mycluster-0  |
+-------------+
```

Not seeing a port-forward to 127.0.0.1:3306 in this example means a local MySQL installation is likely installed and active on the system.

Using port names instead of port numbers also works:

```
$> kubectl port-forward service/mycluster mysql
Forwarding from 127.0.0.1:3306 -> 6446
Forwarding from [::1]:3306 -> 6446
^C
```

```
$> kubectl port-forward service/mycluster mysql-ro
Forwarding from 127.0.0.1:6447 -> 6447
Forwarding from [::1]:6447 -> 6447
```

A list of port names with their associated ports:

```
mysql:            3306
mysqlx:           33060
mysql-alternate:  6446
mysqlx-alternate: 6448
mysql-ro:         6447
mysqlx-ro:        6449
```
For a list of all ports used by MySQL services, see MySQL Port Reference. The ports used here are from MySQL Router.
Chapter 6 Private Registries

Table of Contents

6.1 Install MySQL Operator for Kubernetes from Private Registry using Helm ........................................... 21
6.2 Install InnoDB Cluster from Private Registry using Helm ............................................................................ 22
6.3 Copy Image to Private Registry using Docker ............................................................................................. 22
6.4 Copy Image to Private Registry using Skopeo ............................................................................................... 23

Tasks related to using private registries. This section is a work-in-progress.

6.1 Install MySQL Operator for Kubernetes from Private Registry using Helm

If the private registry is not authenticated, and after pushing the MySQL Operator for Kubernetes image to your private registry, execute the following on the host where helm is installed; and adjust the variable values as needed:

```bash
export REGISTRY="..."   # like 192.168.20.199:5000
export REPOSITORY="..." # like "mysql"
export NAMESPACE="mysql-operator"
helm install mysql-operator helm/mysql-operator \
  --namespace $NAMESPACE \
  --create-namespace \
  --set image.registry=$REGISTRY \
  --set image.repository=$REPOSITORY \
  --set envs.imagesDefaultRegistry=$REGISTRY \
  --set envs.imagesDefaultRepository=$REPOSITORY
```

Authenticated private registries need to create a namespace for MySQL Operator for Kubernetes, and also add a Kubernetes docker-registry secret in the namespace; then execute `helm install` with arguments that look similar to:

```bash
export REGISTRY="..."   # like 192.168.20.199:5000
export REPOSITORY="..." # like "mysql"
export NAMESPACE="mysql-operator"
export DOCKER_SECRET_NAME="priv-reg-secret"
kubectl create namespace $NAMESPACE
kubectl -n $NAMESPACE create secret docker-registry $DOCKER_SECRET_NAME \
  --docker-server="https://$REGISTRY/v2/" \
  --docker-username=user --docker-password=pass \
  --docker-email=user@example.com
helm install mysql-operator helm/mysql-operator \
  --namespace $NAMESPACE \
  --set image.registry=$REGISTRY \
  --set image.repository=$REPOSITORY \
  --set image.pullSecrets.enabled=true \
  --set image.pullSecrets.secretName=$DOCKER_SECRET_NAME \
  --set envs.imagesPullPolicy='IfNotPresent' \
  --set envs.imagesDefaultRegistry=$REGISTRY \
  --set envs.imagesDefaultRepository=$REPOSITORY
```
To confirm the installation, check the status with commands such as `helm list -n $NAMESPACE` and `kubectl -n $NAMESPACE get pods`.

### 6.2 Install InnoDB Cluster from Private Registry using Helm

For example:

```bash
export REGISTRY=...' # like 192.168.20.199:5000
export REPOSITORY=...' # like "mysql"
export NAMESPACE="mynamespace"
export DOCKER_SECRET_NAME="priv-reg-secret"

$> kubectl create namespace $NAMESPACE
$> kubectl -n $NAMESPACE create secret docker-registry $DOCKER_SECRET_NAME \
    --docker-server="https://$REGISTRY/v2/" \
    --docker-username=user --docker-password-pass \
    --docker-email=user@example.com

$> helm install mycluster mysql-operator/mysql-innodbcluster \
    --namespace $NAMESPACE \
    --set credentials.root.user='root' \
    --set credentials.root.password='sakila' \
    --set credentials.root.host='%' \
    --set serverInstances=3 \
    --set routerInstances=1 \
    --set image.registry=$REGISTRY \
    --set image.repository=$REPOSITORY \
    --set image.pullSecrets.enabled=true \
    --set image.pullSecrets.secretName=$DOCKER_SECRET_NAME
```

### 6.3 Copy Image to Private Registry using Docker

Using air-gapped or sanctioned images to avoid pulling images from the internet is another use case and described here.

1. Choose the desired MySQL Operator for Kubernetes version. For example, latest is defined in `helm/mysql-operator/Chart.yaml`. For example, `8.0.34-2.0.11`.
2. Execute `docker pull container-registry.oracle.com/mysql/community-operator:VERSION` where VERSION is the desired MySQL Operator for Kubernetes version.
3. Execute `docker save container-registry.oracle.com/mysql/community-operator:VERSION -o mysql-operator.tar` to export the container image where VERSION is the desired MySQL Operator for Kubernetes version.
4. Copy `mysql-operator.tar` to a host with access to the private registry.
5. Execute `docker load -i mysql-operator.yaml` to load the image into the local Docker cache on that host.
6. Execute `docker tag mysql/mysql-server:VERSION registry:port/repo/mysql-server:VERSION` to retag the image as preparation for pushing to the private registry; adjust VERSION accordingly.
7. Execute `docker push registry:port/repo/mysql-server:VERSION` to push the newly created tag to the private registry; adjust VERSION accordingly.

8. If you won’t need the image from the importing host cache, then you can delete it with `docker rmi mysql/mysql-operator:VERSION registry:port/repo/mysql-server:VERSION`. This removes it from the host but the registry itself won’t be affected. Adjust VERSION accordingly.

Alternatively, you can use the following commands to pull and push in one command. Execute it on a host with Oracle Container Registry (OCR) access. If applicable, this host also needs access to the secure (bastion) host that can access the private registry. Modify the variable values to fit your needs. The command does not consume local space for a tarball but will stream the container image over SSH.

```bash
export BASTION_USER='k8s'
export BASTION_HOST='k8'
export REGISTRY="..." # for example 192.168.20.199:5000
export REPOSITORY="..." # for example mysql
export OPERATOR_VERSION=$(grep appVersion helm/mysql-operator/Chart.yaml | cut -d '"' -f2)
docker pull container-registry.oracle.com/mysql/community-operator:$OPERATOR_VERSION
docker save container-registry.oracle.com/mysql/community-operator:$OPERATOR_VERSION | \
  ssh $BASTION_USER@$BASTION_HOST "docker load && \
  docker tag container-registry.oracle.com/mysql/community-operator:$OPERATOR_VERSION $REGISTRY/$REPOSITORY/mysql-operator:$OPERATOR_VERSION && \
  docker push $REGISTRY/$REPOSITORY/mysql-operator:$OPERATOR_VERSION && \
  docker rmi container-registry.oracle.com/mysql/community-operator:$OPERATOR_VERSION"

docker rmi container-registry.oracle.com/mysql/community-operator:$OPERATOR_VERSION
```

6.4 Copy Image to Private Registry using Skopeo

Similar to Section 6.3, “Copy Image to Private Registry using Docker”, but you might use Skopeo. Skopeo is a container utility that can also run as a container. The following example copies the operator image from the Oracle Container Registry (OCR) to a private registry. It needs to run on a host that has Docker or Podman, and also that has access to both OCR and your private registry. Change the variable names to fit your environment, and change docker to podman if using Podman. The `OPERATOR_VERSION` is the MySQL Operator for Kubernetes version, such as `8.0.34-2.0.11`.

```bash
export REGISTRY="..." # for example 192.168.20.199:5000
export REPOSITORY="..." # for example mysql
export OPERATOR_VERSION=$(grep appVersion helm/mysql-operator/Chart.yaml | cut -d '"' -f2)
```

For authenticated private registries, append `--dest-creds user:pass` to the skopeo command. Also append `--dest-tls-verify=false` if it does not use TLS.
Chapter 7 MySQL Operator Cookbook

Table of Contents

7.1 Handling MySQL Backups ............................................................... 25
7.2 Bootstrap a MySQL InnoDB Cluster from a Dump using Helm ....................... 28
7.3 Viewing Logs ................................................................................. 29

Tasks related to using MySQL Operator for Kubernetes.

7.1 Handling MySQL Backups

There are three main topics related to MySQL backups:

• **Backup profile**: describes the general backup structure that includes storage, schedule, and MySQL Shell dump related options. Defining profiles is optional, and profiles are separated by name.

• **Backup request**: requesting a backup initiates a new object that creates a new pod to perform the backup.

• **Backup schedule**: defined as a cron expression for regular backups, or with no schedule when performing one-off backups.

See also Chapter 8, MySQL Operator Custom Resource Properties for a list of all MySQLBackup resource options.

Backup Profiles with backupProfiles

Backup profiles are defined and reused for regular backups and one-off backups using the backupProfiles specification object. A profile is defined and called from within the InnoDB Cluster specification object, or values can be defined from within the individual backup requests without a profile.

How a Backup is Created

MySQL Operator for Kubernetes supports the dumpInstance() command using MySQL Shell by defining the associated dumpInstance specification object that contains the dumpOptions and storage specification objects:

• The optional dumpOptions value is a dictionary of key-value pairs passed directly to MySQL Shell's DumpInstance() function. See Instance Dump Utility, Schema Dump Utility, and Table Dump Utility for a list of relevant options.

MySQL Operator for Kubernetes adds definitions by default, such as defining threads based on what the system claims as its CPU count; but these values can be overridden.

• The storage configuration specification offers two options as of MySQL Operator for Kubernetes 8.0.29: persistentVolumeClaim or ociObjectStorage (OCI refers to Oracle Cloud Infrastructure).

Note

Limitations: Restore capability is not available for persistentVolumeClaim as of MySQL Operator for Kubernetes 8.0.29, and ociObjectStorage use is specific to the Oracle Cloud Infrastructure (OCI).

• The backupSchedules schedule utilizes the Kubernetes CronJob controller for regular backups.
A PersistentVolumeClaim Scheduled Backup Example

This example uses PersistentVolumeClaim (PVC), sets a daily backup schedule, and defines a backup profile named "myfancyprofile" in the backupProfiles object.

Note

This example defines a single backupProfile and schedule but could define multiple profiles and schedules depending need. For example, a volatile table may have hourly backups in addition to the full nightly backup.

```
apiVersion: mysql.oracle.com/v2
kind: InnoDBCluster
metadata:
  name: mycluster
spec:
  instances: 3
  router:
    instances: 1
  secretName: mypwds
tlsUseSelfSigned: true
backupProfiles:
- name: myfancyprofile  # Embedded backup profile
dumpInstance:         # MySQL Shell Dump
dumpOptions:
  excludeTables: "[world.country]" # Example to exclude one table
storage:
  persistentVolumeClaim:
    claimName: myexample-pvc # store to this pre-existing PVC
backupSchedules:
  name: mygreatschedule
  schedule: "0 0 * * *" # Daily, at midnight
  backupProfileName: myfancyprofile # reference the desired backupProfiles's name
  enabled: true # backup schedules can be temporarily disabled
```

This example requires a PersistentVolumeClaim definition named "myexample-pvc"; see the official Kubernetes Persistent Volumes documentation for PersistentVolumeClaim specifics. A simple example:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: myexample-pv
labels:
  type: local
spec:
  storageClassName: manual
  capacity:
    storage: 2Gi
  accessModes:
  - ReadWriteOnce
hostPath:
  path: /tmp
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: myexample-pvc
spec:
  storageClassName: manual
  accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 2Gi
```
Using OciObjectStorage

The example "mycluster" InnoDB Cluster definition uses a secret named "mypwds" for its root user, for example:

```
$> kubectl create secret generic mypwds \
  --from-literal=rootUser=root \
  --from-literal=rootHost=% \
  --from-literal=rootPassword="sakila"
```

After creating the example InnoDB Cluster, you may want to execute a one-off backup using an existing profile, such as:

```
apiVersion: mysql.oracle.com/v2
kind: MySQLBackup
metadata:
  name: a-cool-one-off-backup
spec:
  clusterName: mycluster
  backupProfileName: myfancyprofile
```

Executing this creates a pod with a name similar to `a-cool-one-off-backup-20220330-215635-t6thv` that executes the backup, and it remains in a Completed state after the backup operation.

Using OciObjectStorage

Using the same example but for Oracle Cloud Infrastructure (OCI) instead of a PVC, modify `dumpInstance.storage` from PrivateVolumeClaim to an ociObjectStorage object similar to:

```
dumpInstance:
  storage:
    ociObjectStorage:
      prefix: someprefix         # a prefix (directory) used for ObjectStorage
      bucketName: bucket         # the ObjectStorage bucket
      credentials: backup-apikey # a secret with credentials ...
```

The backup-apikey secret used in this OCI example looks similar to:

```
apiVersion: v1
kind: Secret
type: Opaque
metadata:
  name: backup-apikey
stringData:
  passphrase: ....
  privatekey: |
    -----BEGIN RSA PRIVATE KEY-----
    MIIEogIBAAKCAQEAwmQ1JGOGUBNwyJuq4msGpBfK24tcKrWaqAkbE1Z/XLOFLvEE
    ...
    region: us-ashburn-1..
    tenancy: ocid1.tenancy....
    user: ocid1.user.....
```

An example method to create the Secret; values are found in the configuration file downloaded from OCI, which is used with the OCI command-line tool.

```
$> kubectl create secret generic <secret_name> \
  --from-literal=user=<userid> \
  --from-literal=fingerprint=<fingerprint> \
  --from-literal=tenancy=<tenancy> \
  --from-literal=region=<region> \
  --from-literal=passphrase=<passphrase> \
  --from-file=privatekey=<path_to_api_key.pem>
```

27
Using profiles (`backupProfileName`) is optional, so instead it may look like the following with the same settings. This example restores to a new InnoDB Cluster from `ociObjectStorage`:

```yaml
apiVersion: mysql.oracle.com/v2
kind: InnoDBCluster
metadata:
  name: newcluster
spec:
  instances: 3
  router:
    instances: 1
  secretName: newpwds
  tlsUseSelfSigned: true
  baseServerId: 2000
  initDB:
    dump:
      name: some-name
  storage:
    ociObjectStorage:
      prefix: someprefix
      bucketName: bucket
      credentials: restore-apikey
```

The secret (`restore-apikey`) could be the same as the backup example (`backup-apikey`) but may be a different user with different permissions, such as no write permissions to the OS.

### Cloning

Data can be initialized using a backup or by cloning an existing and running MySQL instance using `iniDB` and its `donorURL` option:

```yaml
apiVersion: mysql.oracle.com/v2
kind: InnoDBCluster
metadata:
  name: copycluster
spec:
  instances: 1
  secretName: pwds
  tlsUseSelfSigned: true
  initDB:
    clone:
      donorUrl: root@mycluster-0.mycluster-instances.testns.svc.cluster.local:3306
      secretKeyRef:
        name: donorpwds
```

The `donorpwds` secret contains a single field named `rootPassword`, so for example you could reuse the main `secretName` used when creating the original cluster (named `mypwds` in the example). This utilizes MySQL's cloning plugin, so standard limitations apply (such as requiring the same MySQL versions). Cloning can theoretically also be used for creating backups.

### 7.2 Bootstrap a MySQL InnoDB Cluster from a Dump using Helm

A MySQL InnoDB Cluster can be initialized with a database dump created by MySQL Shell or by MySQL Operator for Kubernetes. The backup could reside on a persistent volume accessible from the cluster, but our example uses an OCI Object Storage bucket.

#### Using an OCI Object Storage bucket

If you are bootstrapping from OCI OS, then the following must be known:

- The credentials of the user who has access to OCI OS
7.3 Viewing Logs

Information helpful for debugging and finding relevant log information.

Log locations include each InnoDBCluster Pod, which are divided into a set of containers. There are two operative containers (mysql and sidecar) and three initializer containers (initconf, initmysql, and fixdatadir) as described below here:
### Table 7.1 Containers associated with an InnoDBCluster Pod

<table>
<thead>
<tr>
<th>Container Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sidecar</td>
<td>Initialization, including initial setup of data (initDB) and ongoing maintenance tasks for a specific instance, such as TLS certification updates.</td>
</tr>
<tr>
<td>mysql</td>
<td>The MySQL Server itself.</td>
</tr>
<tr>
<td>initconf</td>
<td>It prepares MySQL configuration files for a specific host. For example, to view its ConfigMap: <code>kubectl get cm {cluster_name}-initconf -o json</code>.</td>
</tr>
<tr>
<td>initmysql</td>
<td>Initializes the MySQL Server, including its data directory.</td>
</tr>
<tr>
<td>fixdatadir</td>
<td>Sets appropriate permissions and ownership of the MySQL data directory, upon initialization.</td>
</tr>
</tbody>
</table>

There's also the dynamic MySQL Operator for Kubernetes and MySQL Router pods.

Examples that assume a basic setup as per `samples/sample-cluster.yaml` which looks like:

```bash
$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
mycluster-0                         2/2     Running   0          99m
mycluster-1                         2/2     Running   0          99m
mycluster-2                         2/2     Running   0          99m
mycluster-router-6d49485474-ftw9r   1/1     Running   0          97m
```

```bash
$ kubectl get pods --namespace mysql-operator
NAME                              READY   STATUS    RESTARTS   AGE
mysql-operator-586f9f5d5b-7wtgl   1/1     Running   0          3h48m
```

Viewing operational Pod logs for debugging active operations:

```bash
$ kubectl logs mycluster-0 -c sidecar
```

```bash
$ kubectl logs mycluster-0 -c mysql
[Entrypoint] MySQL Docker Image 8.0.29-1.2.8-server
2022-04-21T19:15:05.969998Z 0 [Note] [MY-010747] [Server] Plugin 'FEDERATED' is disabled.
2022-04-21T19:15:05.971625Z 0 [Note] [MY-010733] [Server] Shutting down plugin 'MyISAM'
2022-04-21T19:15:07.085923Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.0.29) starting as process
2022-04-21T19:15:07.094129Z 0 [Note] [MY-012366] [InnoDB] Using Linux native AIO
```

```bash
$ kubectl logs mycluster-router-6d49485474-ftw9r
... # Bootstrapping MySQL Router instance at '/tmp/mysqlrouter'
```
Viewing Logs

### MySQL Classic protocol
- Read/Write Connections: localhost:6446
- Read/Only Connections: localhost:6447

```
$> kubectl logs mysql-operator-586f9f5d5b-7wtgl -n mysql-operator
```

2022-04-21 17:06:13: Info: Credential store mechanism is going to be disabled.
2022-04-21 17:06:13: Info: Loading startup files...

```
[2022-04-21 17:06:14,758] kopf.activities.star [INFO    ] OPERATOR_VERSION =2.0.4
[2022-04-21 17:06:14,758] kopf.activities.star [INFO    ] DEFAULT_VERSION_TAG=8.0.29
[2022-04-21 17:06:14,758] kopf.activities.star [INFO    ] SIDECAR_VERSION_TAG=8.0.29-2.0.4
```

Incremental state recovery is now in progress.

```
* Waiting for distributed recovery to finish...
```

```
  online=[<MySQLPod mycluster-0>, <MySQLPod mycluster-1>, <MySQLPod mycluster-2>]
```

Viewing Pods specific to the InnoDBCluster's initialization:

```
$> kubectl logs mycluster-0 -c initmysql
```

[Entrypoint] Initializing database
2022-04-21T14:40.315937Z 0 [Note] [MY-010949] [Server] Basedir set to /usr/.
2022-04-21T14:40.315977Z 0 [System] [MY-013169] [Server] /usr/sbin/mysqld (mysqld 8.0.29) initializing of server in progress as process 19
2022-04-21T14:40.317974Z 0 [Note] [MY-010458] [Server] --initialize specified on an existing data directory.
2022-04-21T14:40.323039Z 0 [Note] [MY-010938] [Server] Generating a new UUID: 4af9d5b7-c1a7-11ec-9663-0242ac110007.
2022-04-21T14:40.329396Z 0 [Note] [MY-012366] [InnoDB] Using Linux native AIO
2022-04-21T14:40.330470Z 0 [Note] [MY-010747] [Server] Plugin 'FEDERATED' is disabled.
2022-04-21T14:40.398051Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
2022-04-21T14:42.103049Z 1 [System] [MY-013172] [Server] Received SHUTDOWN from user root. Shutting down mysqld (Version: 8.0.29).
2022-04-21T14:42.103049Z 0 [System] [MY-010910] [Server] /usr/sbin/mysqld: Shutdown complete (mysqld 8.0.29)
2022-04-21T14:42.103049Z 0 [System] [MY-010909] [Server] MySQL init process done. Ready for start up.

```
[Entrypoint] running /docker-entrypoint-initdb.d/initdb-localroot.sql
```

Viewing Pods specific to the InnoDBCluster's initialization:

```
$> kubectl logs mycluster-0 -c initconf
```

2022-04-21 19:14:35: Info: Credential store mechanism is going to be disabled.
2022-04-21 19:14:35: Info: Loading startup files...

```
  online=[<MySQLPod mycluster-0>, <MySQLPod mycluster-1>, <MySQLPod mycluster-2>]
```

31
Viewing Logs

```
/mnt/initconf:
total 0
lrwxrwxrwx 1 root mysql 19 Apr 21 19:14 00-basic.cnf -> ..data/00-basic.cnf
lrwxrwxrwx 1 root mysql 31 Apr 21 19:14 01-group_replication.cnf -> ..data/01-group_replication.cnf
lrwxrwxrwx 1 root mysql 17 Apr 21 19:14 02-ssl.cnf -> ..data/02-ssl.cnf
lrwxrwxrwx 1 root mysql 19 Apr 21 19:14 99-extra.cnf -> ..data/99-extra.cnf
lrwxrwxrwx 1 root mysql 27 Apr 21 19:14 initdb-localroot.sql -> ..data/initdb-localroot.sql
lrwxrwxrwx 1 root mysql 23 Apr 21 19:14 livenessprobe.sh -> ..data/livenessprobe.sh
lrwxrwxrwx 1 root mysql 16 Apr 21 19:14 my.cnf.in -> ..data/my.cnf.in
lrwxrwxrwx 1 root mysql 24 Apr 21 19:14 readinessprobe.sh -> ..data/readinessprobe.sh

/mnt/mycnfdata:
total 0
2022-04-21T19:14:38 - [INFO] [initmysql] Setting up configurations for mycluster-0 server_id=1000
          report_host=mycluster-0.mycluster-instances.default.svc.cluster.local
2022-04-21T19:14:38 - [INFO] [initmysql] Configuration done

For initconf, you might view their ConfigMap, for example:
```

```
$> kubectl get configmap mycluster-initconf -o yaml

Copied here is the [data] object:
```
```
data:
  00-basic.cnf: |
    # Basic configuration.
    # Do not edit.
    [mysqld]
    plugin_load_add/auth_socket.so
    loose_auth_socket=FORCE_PLUS_PERMANENT
    skip_log_error
    log_error_verbosity=3
  01-group_replication.cnf: |
    # GR and replication related options
    # Do not edit.
    [mysqld]
    log_bin=mycluster
    enforce_gtid_consistency=ON
    gtid_mode=ON
    relay_log_info_repository=TABLE
    skip_slave_start=1
  02-ssl.cnf: |
    # SSL configurations
    # Do not edit.
    [mysqld]
    # ssl-ca=/etc/mysql-ssl/ca.pem
    # ssl-crl=/etc/mysql-ssl/crl.pem
    # ssl-cert=/etc/mysql-ssl/tls.crt
    # ssl-key=/etc/mysql-ssl/tls.key
    loose_group_replication_recovery_use_ssl=1
    loose_group_replication_recovery_ssl_verify_server_cert=1
    loose_group_replication_recovery_ssl_ca=/etc/mysql-ssl/ca.pem
    loose_group_replication_recovery_ssl_crl=/etc/mysql-ssl/crl.pem
    loose_group_replication_recovery_ssl_cert=/etc/mysql-ssl/tls.crt
    loose_group_replication_recovery_ssl_key=/etc/mysql-ssl/tls.key
  99-extra.cnf: |
    # Additional user configurations taken from spec.mycnf in InnoDBCluster.
    # Do not edit directly.
    [mysqld]
    innodb_buffer_pool_size=200M
    innodb_log_file_size=2G
  my.cnf.in: |
# Server identity related options (not shared across instances).
# Do not edit.
[mysqld]
server_id=@@SERVER_ID@@
report_host=@@HOSTNAME@@
datadir=/var/lib/mysql
loose_mysqlx_socket=/var/run/mysqld/mysqlx.sock
socket=/var/run/mysqld/mysql.sock
local-infile=1

[mysql]
socket=/var/run/mysqld/mysql.sock

[mysqladmin]
socket=/var/run/mysqld/mysql.sock

!includedir /etc/my.cnf.d
Chapter 8 MySQL Operator Custom Resource Properties

Resource Types

- InnoDBCluster
- MySQLBackup

InnoDBCluster

Table 8.1 Spec table for InnoDBCluster

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>apiVersion</td>
<td>string</td>
<td>mysql.oracle.com/v2</td>
<td>true</td>
</tr>
<tr>
<td>kind</td>
<td>string</td>
<td>InnoDBCluster</td>
<td>true</td>
</tr>
<tr>
<td>metadata</td>
<td>object</td>
<td>Refer to the Kubernetes API documentation</td>
<td>true</td>
</tr>
<tr>
<td>spec</td>
<td>object</td>
<td></td>
<td>true</td>
</tr>
<tr>
<td>status</td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
</tbody>
</table>

InnoDBCluster.spec

Parent

Table 8.2 Spec table for InnoDBCluster.spec

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>secretName</td>
<td>string</td>
<td>Name of a generic type Secret containing root/default account password</td>
<td>true</td>
</tr>
<tr>
<td>backupProfiles</td>
<td>[]object</td>
<td>Backup profile specifications for the cluster, which can be referenced from backup schedules and one-off backup jobs</td>
<td>false</td>
</tr>
<tr>
<td>backupSchedules</td>
<td>[]object</td>
<td>Schedules for periodically executed backups</td>
<td>false</td>
</tr>
<tr>
<td>baseServerId</td>
<td>integer</td>
<td>Base value for MySQL server_id for instances in the cluster</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Default: 1000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Minimum: 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Maximum: 4294967195</td>
<td></td>
</tr>
<tr>
<td>datadirVolumeClaimTemplate</td>
<td>object</td>
<td>Template for a PersistentVolumeClaim, to be used as datadir</td>
<td>false</td>
</tr>
<tr>
<td>edition</td>
<td>string</td>
<td>MySQL Server Edition (community or enterprise)</td>
<td>false</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
<td>Required</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>imagePullPolicy</td>
<td>string</td>
<td>Defaults to Always, but set to IfNotPresent in deploy-operator.yaml when deploying Operator</td>
<td>false</td>
</tr>
<tr>
<td>imagePullSecrets</td>
<td>[]object</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>imageRepository</td>
<td>string</td>
<td>Repository where images are pulled from; defaults to container-registry.oracle.com/mysql</td>
<td>false</td>
</tr>
<tr>
<td>initDB</td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>instances</td>
<td>integer</td>
<td>Number of MySQL replica instances for the cluster</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Default: 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Minimum: 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Maximum: 9</td>
<td></td>
</tr>
<tr>
<td>keyring</td>
<td>object</td>
<td>Keyring specification</td>
<td>false</td>
</tr>
<tr>
<td>metrics</td>
<td>object</td>
<td>Configuration of a Prometheus-style metrics provider; functionality added in MySQL Operator for Kubernetes 8.1.0-2.1.0.</td>
<td>false</td>
</tr>
<tr>
<td>mycnf</td>
<td>string</td>
<td>Custom configuration additions for my.cnf</td>
<td>false</td>
</tr>
<tr>
<td>podAnnotations</td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>podLabels</td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>podSpec</td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>router</td>
<td>object</td>
<td>MySQL Router specification</td>
<td>false</td>
</tr>
<tr>
<td>serviceAccountName</td>
<td>string</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>tlsCAFileName</td>
<td>string</td>
<td>Name of a generic type Secret containing CA (ca.pem) and optional CRL (crl.pem) for SSL</td>
<td>false</td>
</tr>
<tr>
<td>tlsSecretName</td>
<td>string</td>
<td>Name of a TLS type Secret containing Server certificate and private key for SSL</td>
<td>false</td>
</tr>
<tr>
<td>tlsUseSelfSigned</td>
<td>boolean</td>
<td>Enables use of self-signed TLS certificates, reducing or disabling TLS based security verifications</td>
<td>false</td>
</tr>
</tbody>
</table>
### InnoDBCluster.spec.backupProfiles[index]

**Parent**

Table 8.3 Spec table for InnoDBCluster.spec.backupProfiles[index]

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>name</strong></td>
<td>string</td>
<td>Embedded backup profile, referenced as backupProfileName elsewhere</td>
<td>true</td>
</tr>
<tr>
<td><strong>dumpInstance</strong></td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td><strong>podAnnotations</strong></td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td><strong>podLabels</strong></td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td><strong>snapshot</strong></td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
</tbody>
</table>

### InnoDBCluster.spec.backupProfiles[index].dumpInstance

**Parent**

Table 8.4 Spec table for InnoDBCluster.spec.backupProfiles[index].dumpInstance

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>dumpOptions</strong></td>
<td>object</td>
<td>A dictionary of key-value pairs passed directly to MySQL Shell's DumpInstance()</td>
<td>false</td>
</tr>
<tr>
<td><strong>storage</strong></td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
</tbody>
</table>

### InnoDBCluster.spec.backupProfiles[index].dumpInstance.storage

**Parent**

Table 8.5 Spec table for InnoDBCluster.spec.backupProfiles[index].dumpInstance.storage

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>azure</strong></td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td><strong>ociObjectStorage</strong></td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td><strong>persistentVolumeClaim</strong></td>
<td>object</td>
<td>Specification of the PVC to be used. Used 'as is' in pod executing the backup.</td>
<td>false</td>
</tr>
<tr>
<td><strong>s3</strong></td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
</tbody>
</table>
### Table 8.6 Spec table for `InnoDBCluster.spec.backupProfiles[index].dumpInstance.storage.azure`

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>config</td>
<td>string</td>
<td>Name of a Secret with Azure BLOB Storage configuration and credentials</td>
<td>true</td>
</tr>
<tr>
<td>containerName</td>
<td>string</td>
<td>Name of the Azure BLOB Storage container where the dump is stored</td>
<td>true</td>
</tr>
<tr>
<td>prefix</td>
<td>string</td>
<td>Path in the container where the dump files are stored</td>
<td>false</td>
</tr>
</tbody>
</table>

---

**InnoDBCluster.spec.backupProfiles[index].dumpInstance.storage.ociObjectStorage**

Parent

### Table 8.7 Spec table for `InnoDBCluster.spec.backupProfiles[index].dumpInstance.storage.ociObjectStorage`

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>bucketName</td>
<td>string</td>
<td>Name of the OCI bucket where backup is stored</td>
<td>true</td>
</tr>
<tr>
<td>credentials</td>
<td>string</td>
<td>Name of a Secret with data for accessing the bucket</td>
<td>true</td>
</tr>
<tr>
<td>prefix</td>
<td>string</td>
<td>Path in bucket where backup is stored</td>
<td>false</td>
</tr>
</tbody>
</table>

---

**InnoDBCluster.spec.backupProfiles[index].dumpInstance.storage.s3**

Parent

### Table 8.8 Spec table for `InnoDBCluster.spec.backupProfiles[index].dumpInstance.storage.s3`

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>bucketName</td>
<td>string</td>
<td>Name of the S3 bucket where the dump is stored</td>
<td>true</td>
</tr>
<tr>
<td>config</td>
<td>string</td>
<td>Name of a Secret with S3 configuration and credentials</td>
<td>true</td>
</tr>
<tr>
<td>endpoint</td>
<td>string</td>
<td>Override endpoint URL</td>
<td>false</td>
</tr>
<tr>
<td>prefix</td>
<td>string</td>
<td>Path in the bucket where the dump files are stored</td>
<td>false</td>
</tr>
<tr>
<td>profile</td>
<td>string</td>
<td>Profile being used in configuration files</td>
<td>false</td>
</tr>
<tr>
<td>• Default:</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
InnoDBCluster.spec.backupProfiles[index].snapshot

Table 8.9 Spec table for InnoDBCluster.spec.backupProfiles[index].snapshot

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>storage</td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
</tbody>
</table>

InnoDBCluster.spec.backupProfiles[index].snapshot.storage

Table 8.10 Spec table for InnoDBCluster.spec.backupProfiles[index].snapshot.storage

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>azure</td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>ociObjectStorage</td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>persistentVolumeClaim</td>
<td>object</td>
<td>Specification of the PVC to be used. Used 'as is' in pod executing the backup.</td>
<td>false</td>
</tr>
<tr>
<td>s3</td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
</tbody>
</table>

InnoDBCluster.spec.backupProfiles[index].snapshot.storage.azure

Table 8.11 Spec table for InnoDBCluster.spec.backupProfiles[index].snapshot.storage.azure

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>config</td>
<td>string</td>
<td>Name of a Secret with Azure BLOB Storage configuration and credentials</td>
<td>true</td>
</tr>
<tr>
<td>containerName</td>
<td>string</td>
<td>Name of the Azure BLOB Storage container where the dump is stored</td>
<td>true</td>
</tr>
<tr>
<td>prefix</td>
<td>string</td>
<td>Path in the container where the dump files are stored</td>
<td>false</td>
</tr>
</tbody>
</table>

InnoDBCluster.spec.backupProfiles[index].snapshot.storageOCIObjectStorage

Table 8.12 Spec table for InnoDBCluster.spec.backupProfiles[index].snapshot.storageOCIObjectStorage

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>bucketName</td>
<td>string</td>
<td>Bucket name where backup is stored</td>
<td>true</td>
</tr>
</tbody>
</table>
### InnoDBCluster.spec.backupProfiles[index].snapshot.storage.s3

**Parent**

**Table 8.13 Spec table for InnoDBCluster.spec.backupProfiles[index].snapshot.storage.s3**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>credentials</td>
<td>string</td>
<td>Name of a Secret with data for accessing the bucket</td>
<td>true</td>
</tr>
<tr>
<td>prefix</td>
<td>string</td>
<td>Path in bucket where backup is stored</td>
<td>false</td>
</tr>
</tbody>
</table>

### InnoDBCluster.spec.backupSchedules[index]

**Parent**

**Table 8.14 Spec table for InnoDBCluster.spec.backupSchedules[index]**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the backup schedule</td>
<td>true</td>
</tr>
<tr>
<td>schedule</td>
<td>string</td>
<td>The schedule of the job, syntax as a cron expression</td>
<td>true</td>
</tr>
<tr>
<td>backupProfile</td>
<td>object</td>
<td>backupProfile specification if backupProfileName is not specified</td>
<td>false</td>
</tr>
<tr>
<td>backupProfileName</td>
<td>string</td>
<td>Name of the backupProfile to be used</td>
<td>false</td>
</tr>
<tr>
<td>deleteBackupData</td>
<td>boolean</td>
<td>Whether to delete the backup data in case the MySQLBackup object created by the job is deleted</td>
<td>false</td>
</tr>
</tbody>
</table>
InnoDBCluster.spec.backupSchedules[index].backupProfile

Parent

Description: backupProfile specification if backupProfileName is not specified

Table 8.15 Spec table for InnoDBCluster.spec.backupSchedules[index].backupProfile

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>Whether the schedule is enabled or not</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Default: true</td>
<td></td>
</tr>
</tbody>
</table>

InnoDBCluster.spec.backupSchedules[index].backupProfile.dumpInstance

Parent

Table 8.16 Spec table for InnoDBCluster.spec.backupSchedules[index].backupProfile.dumpInstance

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>dumpOptions</td>
<td>object</td>
<td>A dictionary of key-value pairs passed directly to MySQL Shell's DumpInstance()</td>
<td>false</td>
</tr>
<tr>
<td>storage</td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
</tbody>
</table>

InnoDBCluster.spec.backupSchedules[index].backupProfile.dumpInstance.storage

Parent

Table 8.17 Spec table for InnoDBCluster.spec.backupSchedules[index].backupProfile.dumpInstance.storage

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>azure</td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>ociObjectStorage</td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>persistentVolumeClaim</td>
<td>object</td>
<td>Specification of the PVC to be used. Used 'as is' in pod executing the backup.</td>
<td>false</td>
</tr>
<tr>
<td>s3</td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
</tbody>
</table>
Table 8.18 Spec table for InnoDBCluster.spec.backupSchedules[index].backupProfile.dumpInstance.storage.azure

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>config</td>
<td>string</td>
<td>Name of a Secret with Azure BLOB Storage configuration and credentials</td>
<td>true</td>
</tr>
<tr>
<td>containerName</td>
<td>string</td>
<td>Name of the Azure BLOB Storage container where the dump is stored</td>
<td>true</td>
</tr>
<tr>
<td>prefix</td>
<td>string</td>
<td>Path in the container where the dump files are stored</td>
<td>false</td>
</tr>
</tbody>
</table>

Table 8.19 Spec table for InnoDBCluster.spec.backupSchedules[index].backupProfile.dumpInstance.storage.ociObjectStorage

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>bucketName</td>
<td>string</td>
<td>Name of the OCI Bucket where backup is stored</td>
<td>true</td>
</tr>
<tr>
<td>credentials</td>
<td>string</td>
<td>Name of a Secret with data for accessing the bucket</td>
<td>true</td>
</tr>
<tr>
<td>prefix</td>
<td>string</td>
<td>Path in bucket where backup is stored</td>
<td>false</td>
</tr>
</tbody>
</table>

Table 8.20 Spec table for InnoDBCluster.spec.backupSchedules[index].backupProfile.dumpInstance.storage.s3

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>bucketName</td>
<td>string</td>
<td>Name of the S3 bucket where the dump is stored</td>
<td>true</td>
</tr>
<tr>
<td>config</td>
<td>string</td>
<td>Name of a Secret with S3 configuration and credentials</td>
<td>true</td>
</tr>
<tr>
<td>endpoint</td>
<td>string</td>
<td>Override endpoint URL</td>
<td>false</td>
</tr>
</tbody>
</table>
InnoDBCluster.spec.imagePullSecrets[index]

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefix</td>
<td>string</td>
<td>Path in the bucket where the dump files are stored</td>
<td>false</td>
</tr>
<tr>
<td>profile</td>
<td>string</td>
<td>Profile being used in configuration files</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Default:</td>
<td></td>
</tr>
</tbody>
</table>

InnoDBCluster.spec.imagePullSecrets[index]

Parent

Table 8.21 Spec table for InnoDBCluster.spec.imagePullSecrets[index]

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td></td>
<td>false</td>
</tr>
</tbody>
</table>

InnoDBCluster.spec.initDB

Parent

Table 8.22 Spec table for InnoDBCluster.spec.initDB

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>clone</td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>dump</td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
</tbody>
</table>

InnoDBCluster.spec.initDB.clone

Parent

Table 8.23 Spec table for InnoDBCluster.spec.initDB.clone

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>donorUrl</td>
<td>string</td>
<td>URL of the cluster to clone from</td>
<td>true</td>
</tr>
<tr>
<td>secretKeyRef</td>
<td>object</td>
<td></td>
<td>true</td>
</tr>
<tr>
<td>rootUser</td>
<td>string</td>
<td>User name used for cloning</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Default: root</td>
<td></td>
</tr>
</tbody>
</table>

InnoDBCluster.spec.initDB.clone.secretKeyRef

Parent

Table 8.24 Spec table for InnoDBCluster.spec.initDB.clone.secretKeyRef

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>Secret name with key 'rootPassword' storing</td>
<td>true</td>
</tr>
</tbody>
</table>
InnoDBCluster.spec.initDB.dump

**Parent**

Table 8.25 Spec table for InnoDBCluster.spec.initDB.dump

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>storage</td>
<td>object</td>
<td>Name of the dump. Not used by the operator, but a descriptive hint for the</td>
<td>true</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cluster administrator</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the dump. Not used by the operator, but a descriptive hint for the</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cluster administrator</td>
<td></td>
</tr>
<tr>
<td>options</td>
<td>object</td>
<td>A dictionary of key-value pairs passed directly to MySQL Shell's</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td></td>
<td>loadDump()</td>
<td></td>
</tr>
<tr>
<td>path</td>
<td>string</td>
<td>Path to the dump in the PVC. Use when specifying persistentVolumeClaim.</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Omit for ociObjectStorage, S3, or azure.</td>
<td></td>
</tr>
</tbody>
</table>

InnoDBCluster.spec.initDB.dump.storage

**Parent**

Table 8.26 Spec table for InnoDBCluster.spec.initDB.dump.storage

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>azure</td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>ociObjectStorage</td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>persistentVolumeClaim</td>
<td>object</td>
<td>Specification of the PVC to be used. Used 'as is' in the cloning pod.</td>
<td>false</td>
</tr>
<tr>
<td>s3</td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
</tbody>
</table>

InnoDBCluster.spec.initDB.dump.storage.azure

**Parent**

Table 8.27 Spec table for InnoDBCluster.spec.initDB.dump.storage.azure

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>config</td>
<td>string</td>
<td>Name of a Secret with Azure BLOB Storage</td>
<td>true</td>
</tr>
</tbody>
</table>
InnoDBCluster.spec.initDB.dump.storage.ociObjectStorage

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>containerName</td>
<td>string</td>
<td>Name of the Azure BLOB Storage container where the dump is stored</td>
<td>true</td>
</tr>
<tr>
<td>prefix</td>
<td>string</td>
<td>Path in the container where the dump files are stored</td>
<td>true</td>
</tr>
</tbody>
</table>

InnoDBCluster.spec.initDB.dump.storage.ociObjectStorage

Parent

Table 8.28 Spec table for InnoDBCluster.spec.initDB.dump.storage.ociObjectStorage

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>bucketName</td>
<td>string</td>
<td>Name of the OCI bucket where the dump is stored</td>
<td>true</td>
</tr>
<tr>
<td>credentials</td>
<td>string</td>
<td>Name of a Secret with data for accessing the bucket</td>
<td>true</td>
</tr>
<tr>
<td>prefix</td>
<td>string</td>
<td>Path in the bucket where the dump files are stored</td>
<td>true</td>
</tr>
</tbody>
</table>

InnoDBCluster.spec.initDB.dump.storage.s3

Parent

Table 8.29 Spec table for InnoDBCluster.spec.initDB.dump.storage.s3

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>bucketName</td>
<td>string</td>
<td>Name of the S3 bucket where the dump is stored</td>
<td>true</td>
</tr>
<tr>
<td>config</td>
<td>string</td>
<td>Name of a Secret with S3 configuration and credentials</td>
<td>true</td>
</tr>
<tr>
<td>prefix</td>
<td>string</td>
<td>Path in the bucket where the dump files are stored</td>
<td>true</td>
</tr>
<tr>
<td>endpoint</td>
<td>string</td>
<td>Override endpoint URL</td>
<td>false</td>
</tr>
<tr>
<td>profile</td>
<td>string</td>
<td>Profile being used in configuration files</td>
<td>false</td>
</tr>
</tbody>
</table>

InnoDBCluster.spec.keyring

Parent

Description: Keyring specification
Table 8.30 Spec table for InnoDBCluster.spec.keyring

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>encryptedFile</td>
<td>object</td>
<td>Keyring 'Encrypted File' specification</td>
<td>false</td>
</tr>
<tr>
<td>file</td>
<td>object</td>
<td>Keyring 'File' specification</td>
<td>false</td>
</tr>
<tr>
<td>oci</td>
<td>object</td>
<td>Keyring 'OCI' specification</td>
<td>false</td>
</tr>
</tbody>
</table>

InnoDBCluster.spec.keyring.encryptedFile

**Parent**

Description: Keyring 'Encrypted File' specification

Table 8.31 Spec table for InnoDBCluster.spec.keyring.encryptedFile

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileName</td>
<td>string</td>
<td>Full path to the keyring file name inside the storage volume</td>
<td>true</td>
</tr>
<tr>
<td>password</td>
<td>string</td>
<td>Name of a secret that contains password for the keyring in the key 'keyring_password'</td>
<td>true</td>
</tr>
<tr>
<td>storage</td>
<td>object</td>
<td>Specification of the volume to be mounted where the keyring file resides</td>
<td>true</td>
</tr>
<tr>
<td>readOnly</td>
<td>boolean</td>
<td>Whether to open the keyring file in read-only mode</td>
<td>false</td>
</tr>
</tbody>
</table>

- **Default**: false

InnoDBCluster.spec.keyring.file

**Parent**

Description: Keyring 'File' specification

Table 8.32 Spec table for InnoDBCluster.spec.keyring.file

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileName</td>
<td>string</td>
<td>Full path to the keyring file name inside the storage volume</td>
<td>true</td>
</tr>
<tr>
<td>storage</td>
<td>object</td>
<td>Specification of the volume to be mounted where the keyring file resides</td>
<td>true</td>
</tr>
</tbody>
</table>
InnoDBCluster.spec.keyring.oci

Parent

Description: Keyring 'OCI' specification

Table 8.33 Spec table for InnoDBCluster.spec.keyring.oci

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>readOnly</td>
<td>boolean</td>
<td>Whether to open the keyring file in read-only mode</td>
<td>false</td>
</tr>
<tr>
<td>keyFingerprint</td>
<td>string</td>
<td>Private key fingerprint</td>
<td>true</td>
</tr>
<tr>
<td>keySecret</td>
<td>string</td>
<td>A secret that contains the private key under the field 'privatekey'</td>
<td>true</td>
</tr>
<tr>
<td>tenancy</td>
<td>string</td>
<td>Tenancy identifier in the form ocid1.tenancy.oc1...</td>
<td>true</td>
</tr>
<tr>
<td>user</td>
<td>string</td>
<td>User identifier in the form of ocid1.user.oc1...</td>
<td>true</td>
</tr>
<tr>
<td>caCertificate</td>
<td>string</td>
<td>Secret that contains ca.crt field with CA certificate bundle file that the keyring_oci plugin uses for Oracle Cloud Infrastructure certificate verification</td>
<td>false</td>
</tr>
<tr>
<td>compartment</td>
<td>string</td>
<td>Compartment identifier in the form ocid1.compartment.oc1...</td>
<td>false</td>
</tr>
<tr>
<td>endpoints</td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>masterKey</td>
<td>string</td>
<td>Master key identified in the form ocid1.key.oc1...</td>
<td>false</td>
</tr>
<tr>
<td>virtualVault</td>
<td>string</td>
<td>Vault identifier in the form ocid1.vault.oc1...</td>
<td>false</td>
</tr>
</tbody>
</table>

InnoDBCluster.spec.keyring.oci.endpoints

Parent

Table 8.34 Spec table for InnoDBCluster.spec.keyring.oci.endpoints

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>encryption</td>
<td>string</td>
<td>Encryption endpoint URI like {identifier}-crypto.kms. {region}.oraclecloud.com</td>
<td>false</td>
</tr>
</tbody>
</table>
### InnoDBCluster.spec.metrics

#### Description
Configuration of a Prometheus-style metrics provider; functionality added in MySQL Operator for Kubernetes 8.1.0-2.1.0.

#### Table 8.35 Spec table for InnoDBCluster.spec.metrics

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>management</td>
<td>string</td>
<td>Management endpoint URI like (identifier)-management.kms. (region).oraclecloud.com</td>
<td>false</td>
</tr>
<tr>
<td>secrets</td>
<td>string</td>
<td>Secrets endpoint URI like secrets.vaults. (region).oci.oraclecloud.com</td>
<td>false</td>
</tr>
<tr>
<td>vaults</td>
<td>string</td>
<td>Vaults endpoint URI like vaults. (region).oci.oraclecloud.com</td>
<td>false</td>
</tr>
<tr>
<td>enable</td>
<td>boolean</td>
<td>Toggle to enable or disable the metrics sidecar</td>
<td>true</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Default: false</td>
<td></td>
</tr>
<tr>
<td>image</td>
<td>string</td>
<td>Name of an image to be used for the metrics sidecar, if provided metrics will be enabled</td>
<td>true</td>
</tr>
<tr>
<td>monitor</td>
<td>boolean</td>
<td>Create a ServiceMonitor for Prometheus Operator</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Default: false</td>
<td></td>
</tr>
<tr>
<td>monitorSpec</td>
<td>object</td>
<td>Custom configuration for the ServiceMonitor object</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Default: map[]</td>
<td></td>
</tr>
<tr>
<td>options</td>
<td>[]string</td>
<td>Options passed to the metrics provider as command line arguments</td>
<td>false</td>
</tr>
<tr>
<td>tlsSecret</td>
<td>string</td>
<td>Name of a Secret with TLS certificate, key and CA, which will be mounted at /tls into the container an can be used from webConfig</td>
<td>false</td>
</tr>
<tr>
<td>webConfig</td>
<td>string</td>
<td>Name of a ConfigMap with a web.config file, if this option is provided a</td>
<td>false</td>
</tr>
</tbody>
</table>
### InnoDBCluster.spec.router

**Parent**

Description: MySQL Router specification

#### Table 8.36 Spec table for InnoDBCluster.spec.router

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>instances</td>
<td>integer</td>
<td>Number of MySQL Router instances to deploy</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Default: 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Minimum: 0</td>
<td></td>
</tr>
<tr>
<td>podAnnotations</td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>podLabels</td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>podSpec</td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>tlsSecretName</td>
<td>string</td>
<td>Name of a TLS type Secret containing MySQL Router certificate and private key used for SSL</td>
<td>false</td>
</tr>
<tr>
<td>version</td>
<td>string</td>
<td>Override MySQL Router version</td>
<td>false</td>
</tr>
</tbody>
</table>

### MySQLBackup

#### Table 8.37 Spec table for MySQLBackup

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>apiVersion</td>
<td>string</td>
<td>mysql.oracle.com/v2</td>
<td>true</td>
</tr>
<tr>
<td>kind</td>
<td>string</td>
<td>MySQLBackup</td>
<td>true</td>
</tr>
<tr>
<td>metadata</td>
<td>object</td>
<td>Refer to the Kubernetes API documentation</td>
<td>true</td>
</tr>
<tr>
<td>spec</td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>status</td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
</tbody>
</table>

### MySQLBackup.spec

**Parent**

#### Table 8.38 Spec table for MySQLBackup.spec

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>clusterName</td>
<td>string</td>
<td></td>
<td>true</td>
</tr>
</tbody>
</table>
### MySQLBackup.spec.backupProfile

**Parent**

Description: backupProfile specification if backupProfileName is not specified

**Table 8.39 Spec table for MySQLBackup.spec.backupProfile**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>addTimestampToBackupDirectory</td>
<td>boolean</td>
<td>• Default: true</td>
<td>false</td>
</tr>
<tr>
<td>backupProfile</td>
<td>object</td>
<td>backupProfile specification if backupProfileName is not specified</td>
<td>false</td>
</tr>
<tr>
<td>backupProfileName</td>
<td>string</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>deleteBackupData</td>
<td>boolean</td>
<td>• Default: false</td>
<td>false</td>
</tr>
</tbody>
</table>

### MySQLBackup.spec.backupProfile.dumpInstance

**Parent**

**Table 8.40 Spec table for MySQLBackup.spec.backupProfile.dumpInstance**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>dumpInstance</td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>podAnnotations</td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>podLabels</td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
</tbody>
</table>

### MySQLBackup.spec.backupProfile.dumpInstance.storage

**Parent**

**Table 8.41 Spec table for MySQLBackup.spec.backupProfile.dumpInstance.storage**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>azure</td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>ociObjectStorage</td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>persistentVolumeClaim</td>
<td>object</td>
<td>Specification of the PVC to be used. Used 'as is' in pod executing the backup.</td>
<td>false</td>
</tr>
<tr>
<td>s3</td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
</tbody>
</table>
### MySQLBackup.spec.backupProfile.dumpInstance.storage.azure

Parent

Table 8.42 Spec table for MySQLBackup.spec.backupProfile.dumpInstance.storage.azure

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>config</td>
<td>string</td>
<td>Name of a Secret with Azure BLOB Storage configuration and credentials</td>
<td>true</td>
</tr>
<tr>
<td>containerName</td>
<td>string</td>
<td>Name of the Azure BLOB Storage container where the dump is stored</td>
<td>true</td>
</tr>
<tr>
<td>prefix</td>
<td>string</td>
<td>Path in the container where the dump files are stored</td>
<td>false</td>
</tr>
</tbody>
</table>

### MySQLBackup.spec.backupProfile.dumpInstance.storage.ociObjectStorage

Parent

Table 8.43 Spec table for MySQLBackup.spec.backupProfile.dumpInstance.storage.ociObjectStorage

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>bucketName</td>
<td>string</td>
<td>Name of the OCI bucket where backup is stored</td>
<td>true</td>
</tr>
<tr>
<td>credentials</td>
<td>string</td>
<td>Name of a Secret with data for accessing the bucket</td>
<td>true</td>
</tr>
<tr>
<td>prefix</td>
<td>string</td>
<td>Path in bucket where backup is stored</td>
<td>false</td>
</tr>
</tbody>
</table>

### MySQLBackup.spec.backupProfile.dumpInstance.storage.s3

Parent

Table 8.44 Spec table for MySQLBackup.spec.backupProfile.dumpInstance.storage.s3

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>bucketName</td>
<td>string</td>
<td>Name of the S3 bucket where the dump is stored</td>
<td>true</td>
</tr>
<tr>
<td>config</td>
<td>string</td>
<td>Name of a Secret with S3 configuration and credentials</td>
<td>true</td>
</tr>
<tr>
<td>endpoint</td>
<td>string</td>
<td>Override endpoint URL</td>
<td>false</td>
</tr>
<tr>
<td>prefix</td>
<td>string</td>
<td>Path in the bucket where the dump files are stored</td>
<td>false</td>
</tr>
<tr>
<td>profile</td>
<td>string</td>
<td>Profile being used in configuration files</td>
<td>false</td>
</tr>
</tbody>
</table>
MySQLBackup.status

Parent

Table 8.45 Spec table for MySQLBackup.status

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>bucket</td>
<td>string</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>completionTime</td>
<td>string</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>container</td>
<td>string</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>elapsedTime</td>
<td>string</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>message</td>
<td>string</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>method</td>
<td>string</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>ociTenancy</td>
<td>string</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>output</td>
<td>string</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>size</td>
<td>string</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>source</td>
<td>string</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>spaceAvailable</td>
<td>string</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>startTime</td>
<td>string</td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>status</td>
<td>string</td>
<td></td>
<td>false</td>
</tr>
</tbody>
</table>

Resource Types

- ClusterKopfPeering
- KopfPeering

ClusterKopfPeering

Table 8.46 Spec table for ClusterKopfPeering

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>apiVersion</td>
<td>string</td>
<td>zalando.org/v1</td>
<td>true</td>
</tr>
<tr>
<td>kind</td>
<td>string</td>
<td>ClusterKopfPeering</td>
<td>true</td>
</tr>
<tr>
<td>metadata</td>
<td>object</td>
<td>Refer to the Kubernetes API documentation</td>
<td>true</td>
</tr>
<tr>
<td>status</td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
</tbody>
</table>

KopfPeering

Table 8.47 Spec table for KopfPeering

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>apiVersion</td>
<td>string</td>
<td>zalando.org/v1</td>
<td>true</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
<td>Required</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td>--------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>kind</td>
<td>string</td>
<td>KopfPeering</td>
<td>true</td>
</tr>
<tr>
<td>metadata</td>
<td>object</td>
<td>Refer to the Kubernetes API documentation</td>
<td>true</td>
</tr>
<tr>
<td>status</td>
<td>object</td>
<td></td>
<td>false</td>
</tr>
</tbody>
</table>