Percona Distribution for MySQL Operator

Release 1.9.0

Aug 09, 2021

CONTENTS

Kubernetes and the OpenShift platform, based on Kubernetes, have added a way to manage containerized systems, including database clusters. This management is achieved by controllers, declared in configuration files. These controllers provide automation with the ability to create objects, such as a container or a group of containers called pods, to listen for an specific event and then perform a task.

This automation adds a level of complexity to the container-based architecture and stateful applications, such as a database. A Kubernetes Operator is a special type of controller introduced to simplify complex deployments. The Operator extends the Kubernetes API with custom resources.

Percona XtraDB Cluster is an open-source enterprise MySQL solution that helps you to ensure data availability for your applications while improving security and simplifying the development of new applications in the most demanding public, private, and hybrid cloud environments.

Based on our best practices for deployment and configuration, Percona Distribution for MySQL Operator contains everything you need to quickly and consistently deploy and scale Percona XtraDB Cluster instances in a Kubernetes-based environment on-premises or in the cloud.

Part I

Requirements

CHAPTER

ONE

SYSTEM REQUIREMENTS

The Operator supports Percona XtraDB Cluster (PXC) 5.7 and 8.0.

The new `caching_sha2_password` authentication plugin which is default in 8.0 is not supported for the ProxySQL compatibility reasons. Therefore both Percona XtraDB Cluster 5.7 and 8.0 are configured with `default_authentication_plugin = mysql_native_password`.

1.1 Officially supported platforms

The following platforms were tested and are officially supported by the Operator 1.9.0:

- OpenShift 3.11
- OpenShift 4.7
- Google Kubernetes Engine (GKE) 1.16 - 1.20
- Amazon Elastic Kubernetes Service (EKS) 1.19
- Minikube 1.19

Other Kubernetes platforms may also work but have not been tested.

1.2 Resource Limits

A cluster running an officially supported platform contains at least three Nodes, with the following resources:

- 2GB of RAM,
- 2 CPU threads per Node for Pods provisioning,
- at least 60GB of available storage for Persistent Volumes provisioning.

1.3 Platform-specific limitations

The Operator is subsequent to specific platform limitations.

- Minikube doesn't support multi-node cluster configurations because of its local nature, which is in collision with the default affinity requirements of the Operator. To arrange this, the *Install Percona XtraDB Cluster on Minikube* instruction includes an additional step which turns off the requirement of having not less than three Nodes.

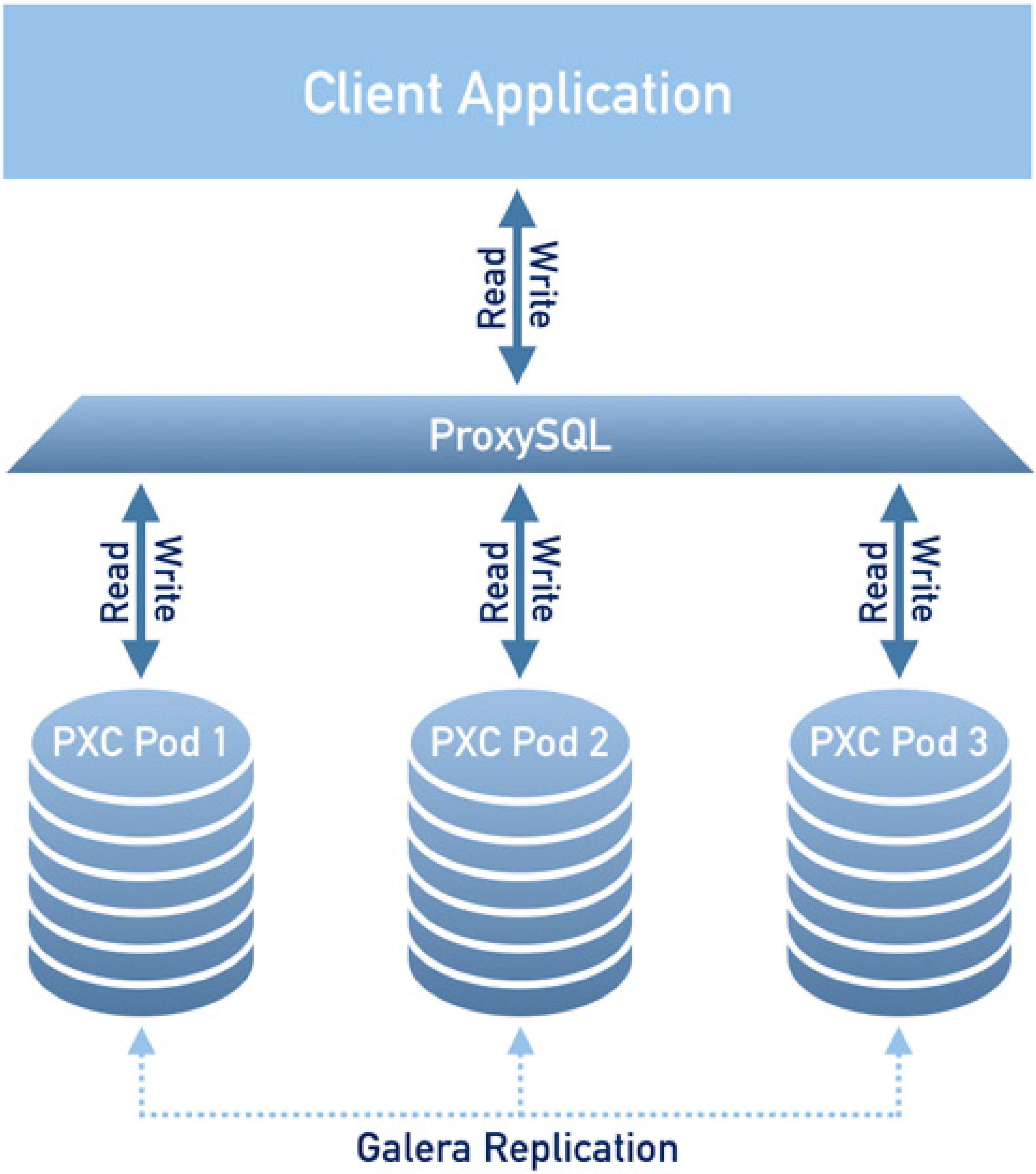
CHAPTER

TWO

DESIGN OVERVIEW

Percona XtraDB Cluster integrates *Percona Server for MySQL* running with the XtraDB storage engine, and *Percona XtraBackup* with the *Galera library* to enable synchronous multi-primary replication.

The design of the Percona Distribution for MySQL Operator is highly bound to the Percona XtraDB Cluster high availability implementation, which in its turn can be briefly described with the following diagram.

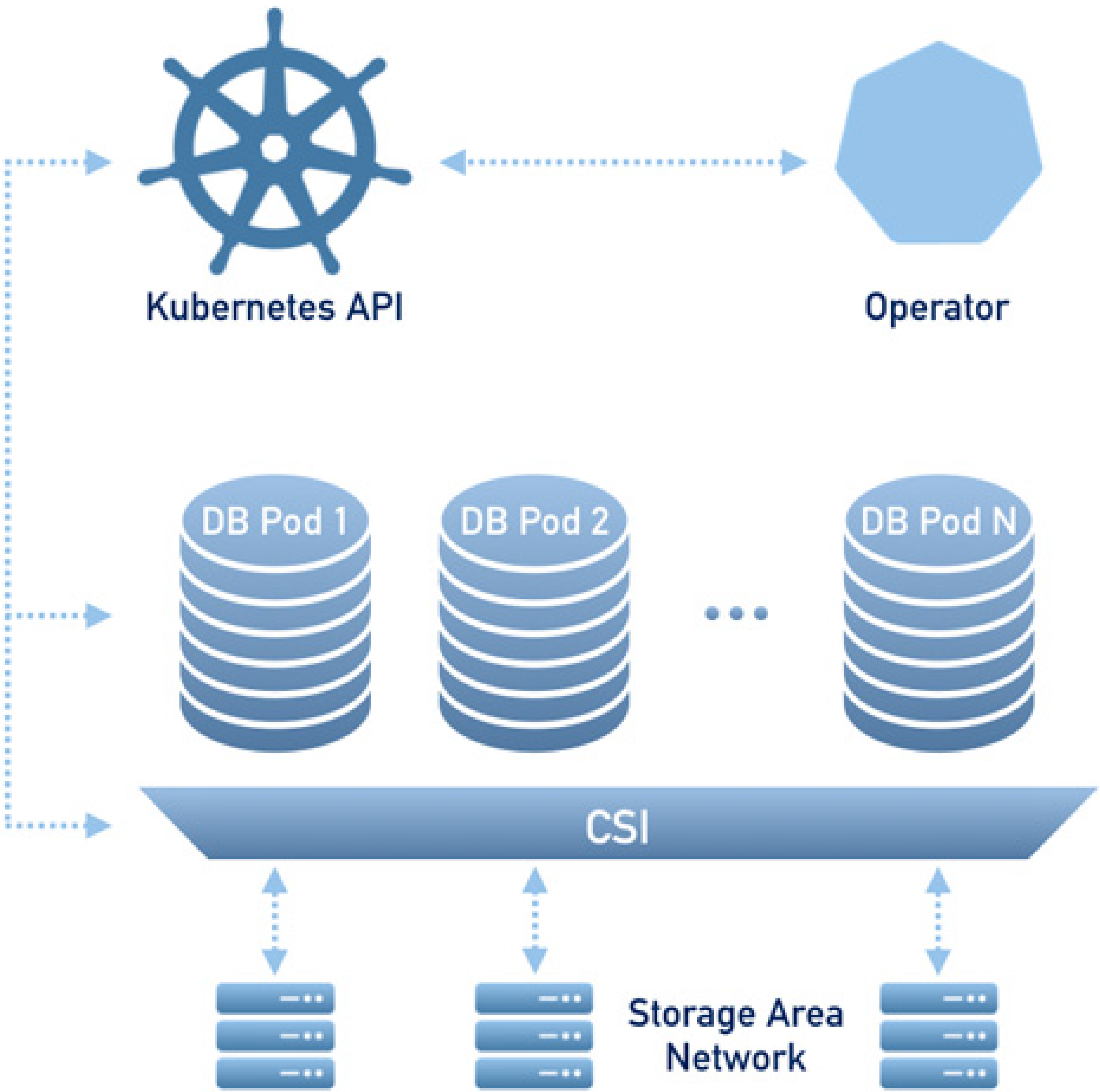


Being a regular MySQL Server instance, each node contains the same set of data synchronized accross nodes. The recommended configuration is to have at least 3 nodes. In a basic setup with this amount of nodes, Percona XtraDB Cluster provides high availability, continuing to function if you take any of the nodes down. Additionally load balancing can be achieved with the ProxySQL daemon, which accepts incoming traffic from MySQL clients and forwards it to backend MySQL servers.

Note: Using ProxySQL results in more efficient database workload management in comparison with other load balancers which are not SQL-aware, including built-in ones of the cloud providers, or the Kubernetes NGINX Ingress

Controller.

To provide high availability operator uses node affinity to run Percona XtraDB Cluster instances on separate worker nodes if possible. If some node fails, the pod with it is automatically re-created on another node.



To provide data storage for stateful applications, Kubernetes uses Persistent Volumes. A *PersistentVolumeClaim* (PVC) is used to implement the automatic storage provisioning to pods. If a failure occurs, the Container Storage Interface (CSI) should be able to re-mount storage on a different node. The PVC StorageClass must support this feature (Kubernetes and OpenShift support this in versions 1.9 and 3.9 respectively).

The Operator functionality extends the Kubernetes API with *PerconaXtraDBCluster* object, and it is implemented as a golang application. Each *PerconaXtraDBCluster* object maps to one separate Percona XtraDB Cluster setup. The Operator listens to all events on the created objects. When a new PerconaXtraDBCluster object is created, or an existing one undergoes some changes or deletion, the operator automatically creates/changes/deletes all needed Kubernetes objects with the appropriate settings to provide a proper Percona XtraDB Cluster operation.

Part II

Quickstart guides

CHAPTER

THREE

INSTALL PERCONA XTRADB CLUSTER ON MINIKUBE

Installing the Percona Distribution for MySQL Operator on minikube is the easiest way to try it locally without a cloud provider. Minikube runs Kubernetes on GNU/Linux, Windows, or macOS system using a system-wide hypervisor, such as VirtualBox, KVM/QEMU, VMware Fusion or Hyper-V. Using it is a popular way to test the Kubernetes application locally prior to deploying it on a cloud.

The following steps are needed to run Percona Distribution for MySQL Operator on Minikube:

1. Install Minikube, using a way recommended for your system. This includes the installation of the following three components:

   1. kubectl tool,
   2. a hypervisor, if it is not already installed,
   3. actual Minikube package

   After the installation, run `minikube start --memory=4096 --cpus=3` (parameters increase the virtual machine limits for the CPU cores and memory, to ensure stable work of the Operator). Being executed, this command will download needed virtualized images, then initialize and run the cluster. After Minikube is successfully started, you can optionally run the Kubernetes dashboard, which visually represents the state of your cluster. Executing `minikube dashboard` will start the dashboard and open it in your default web browser.

2. Deploy the operator with the following command:

   ```
   kubectl apply -f https://raw.githubusercontent.com/percona/percona-xtradb-cluster-
   ↪operator/v1.9.0/deploy/bundle.yaml
   ```

3. Deploy Percona XtraDB Cluster:

   ```
   kubectl apply -f https://raw.githubusercontent.com/percona/percona-xtradb-cluster-
   ↪operator/v1.9.0/deploy/cr-minimal.yaml
   ```

   This deploys one Percona XtraDB Cluster node and one HAProxy node. `deploy/cr-minimal.yaml` is for minimal non-production deployment. For more configuration options please see `deploy/cr.yaml` and *Custom Resource Options*. Creation process will take some time. The process is over when both operator and replica set pod have reached their Running status. `kubectl get pods` output should look like this:

   ```
   NAME                                               READY   STATUS    RESTARTS   AGE
   percona-xtradb-cluster-operator-d99c748-sqddq      1/1     Running   0          49m
   minimal-cluster-pxc-0                              3/3     Running   0          47m
   minimal-cluster-haproxy-0                          2/2     Running   0          47m
   ```

   You can clone the repository with all manifests and source code by executing the following command:

```
git clone -b v1.9.0 https://github.com/percona/percona-xtradb-cluster-operator
```

4. During previous steps, the Operator has generated several secrets, including the password for the `root` user, which you will definitely need to access the cluster. Use `kubectl get secrets` to see the list of Secrets objects (by default Secrets object you are interested in has `minimal-cluster-secrets` name). Then `kubectl get secret minimal-cluster-secrets -o yaml` will return the YAML file with generated secrets, including the root password which should look as follows:

```
...
data:
  ...
  root: cm9vdF9wYXNzd29yZA==
```

   Here the actual password is base64-encoded, and `echo 'cm9vdF9wYXNzd29yZA==' | base64 --decode` will bring it back to a human-readable form.

5. Check connectivity to a newly created cluster.

   First of all, run percona-client and connect its console output to your terminal (running it may require some time to deploy the correspondent Pod):

```
kubectl run -i --rm --tty percona-client --image=percona:8.0 --restart=Never --␣
↪bash -il
```

   Now run `mysql` tool in the percona-client command shell using the password obtained from the secret:

```
mysql -h minimal-cluster-haproxy -uroot -proot_password
```

   This command will connect you to the MySQL monitor.

```
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1872
Server version: 8.0.22-13.1 Percona XtraDB Cluster (GPL), Release rel13, Revision␣
↪a48e6d5, WSREP version 26.4.3

Copyright (c) 2009-2021 Percona LLC and/or its affiliates
Copyright (c) 2000, 2021, Oracle and/or its affiliates.

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

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

CHAPTER

FOUR

INSTALL PERCONA XTRADB CLUSTER ON GOOGLE KUBERNETES ENGINE (GKE)

This quickstart shows you how to configure the Percona Distribution for MySQL Operator with the Google Kubernetes Engine. The document assumes some experience with Google Kubernetes Engine (GKE). For more information on the GKE, see the Kubernetes Engine Quickstart.

4.1 Prerequisites

All commands from this quickstart can be run either in the **Google Cloud shell** or in **your local shell**.

To use *Google Cloud shell*, you need nothing but a modern web browser.

If you would like to use *your local shell*, install the following:

1. gcloud. This tool is part of the Google Cloud SDK. To install it, select your operating system on the official Google Cloud SDK documentation page and then follow the instructions.
2. kubectl. It is the Kubernetes command-line tool you will use to manage and deploy applications. To install the tool, run the following command:

```
$ gcloud auth login
$ gcloud components install kubectl
```

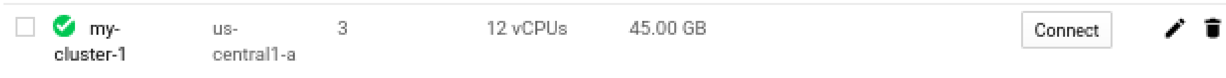
4.2 Configuring default settings for the cluster

You can configure the settings using the `gcloud` tool. You can run it either in the Cloud Shell or in your local shell (if you have installed Google Cloud SDK locally on the previous step). The following command will create a cluster named `my-cluster-1`:

```
$ gcloud container clusters create my-cluster-1 --project <project name> --zone us-
↪central1-a --cluster-version 1.20 --machine-type n1-standard-4 --num-nodes=3
```

Note: You must edit the following command and other command-line statements to replace the `<project name>` placeholder with your project name. You may also be required to edit the *zone location*, which is set to `us-central1` in the above example. Other parameters specify that we are creating a cluster with 3 nodes and with machine type of 4 vCPUs and 45 GB memory.

You may wait a few minutes for the cluster to be generated, and then you will see it listed in the Google Cloud console (select *Kubernetes Engine* → *Clusters* in the left menu panel):



Now you should configure the command-line access to your newly created cluster to make `kubectl` be able to use it.

In the Google Cloud Console, select your cluster and then click the *Connect* shown on the above image. You will see the connect statement configures command-line access. After you have edited the statement, you may run the command in your local shell:

```
$ gcloud container clusters get-credentials my-cluster-1 --zone us-central1-a --project <project name>
```

4.3 Installing the Operator

1. First of all, use your Cloud Identity and Access Management (Cloud IAM) to control access to the cluster. The following command will give you the ability to create Roles and RoleBindings:

   ```
   $ kubectl create clusterrolebinding cluster-admin-binding --clusterrole cluster-admin --user $(gcloud config get-value core/account)
   ```

   The return statement confirms the creation:

   ```
   clusterrolebinding.rbac.authorization.k8s.io/cluster-admin-binding created
   ```

2. Create a namespace and set the context for the namespace. The resource names must be unique within the namespace and provide a way to divide cluster resources between users spread across multiple projects.

   So, create the namespace and save it in the namespace context for subsequent commands as follows (replace the `<namespace name>` placeholder with some descriptive name):

   ```
   $ kubectl create namespace <namespace name>
   $ kubectl config set-context $(kubectl config current-context) --namespace=<namespace name>
   ```

   At success, you will see the message that namespace/<namespace name> was created, and the context (gke_<project name>_<zone location>_<cluster name>) was modified.

3. Use the following `git clone` command to download the correct branch of the percona-xtradb-cluster-operator repository:

   ```
   git clone -b v1.9.0 https://github.com/percona/percona-xtradb-cluster-operator
   ```

   After the repository is downloaded, change the directory to run the rest of the commands in this document:

   ```
   cd percona-xtradb-cluster-operator
   ```

4. Deploy the Operator with the following command:

   ```
   kubectl apply -f deploy/bundle.yaml
   ```

   The following confirmation is returned:

```
customresourcedefinition.apiextensions.k8s.io/perconaxtradbclusters.pxc.percona.com␣
↪created
customresourcedefinition.apiextensions.k8s.io/perconaxtradbclusterbackups.pxc.
↪percona.com created
customresourcedefinition.apiextensions.k8s.io/perconaxtradbclusterrestores.pxc.
↪percona.com created
customresourcedefinition.apiextensions.k8s.io/perconaxtradbbackups.pxc.percona.com␣
↪created
role.rbac.authorization.k8s.io/percona-xtradb-cluster-operator created
serviceaccount/percona-xtradb-cluster-operator created
rolebinding.rbac.authorization.k8s.io/service-account-percona-xtradb-cluster-
↪operator created
deployment.apps/percona-xtradb-cluster-operator created
```

5. The operator has been started, and you can create the Percona XtraDB cluster:

```
$ kubectl apply -f deploy/cr.yaml
```

The process could take some time. The return statement confirms the creation:

```
perconaxtradbcluster.pxc.percona.com/cluster1 created
```

6. During previous steps, the Operator has generated several secrets, including the password for the `root` user, which you will need to access the cluster.

Use `kubectl get secrets` command to see the list of Secrets objects (by default Secrets object you are interested in has `my-cluster-secrets` name). Then `kubectl get secret my-cluster-secrets -o yaml` will return the YAML file with generated secrets, including the root password which should look as follows:

```
...
data:
  ...
  root: cm9vdF9wYXNzd29yZA==
```

Here the actual password is base64-encoded, and `echo 'cm9vdF9wYXNzd29yZA==' | base64 --decode` will bring it back to a human-readable form.

4.4 Verifying the cluster operator

It may take ten minutes to get the cluster started. You can verify its creation with the `kubectl get pods` command:

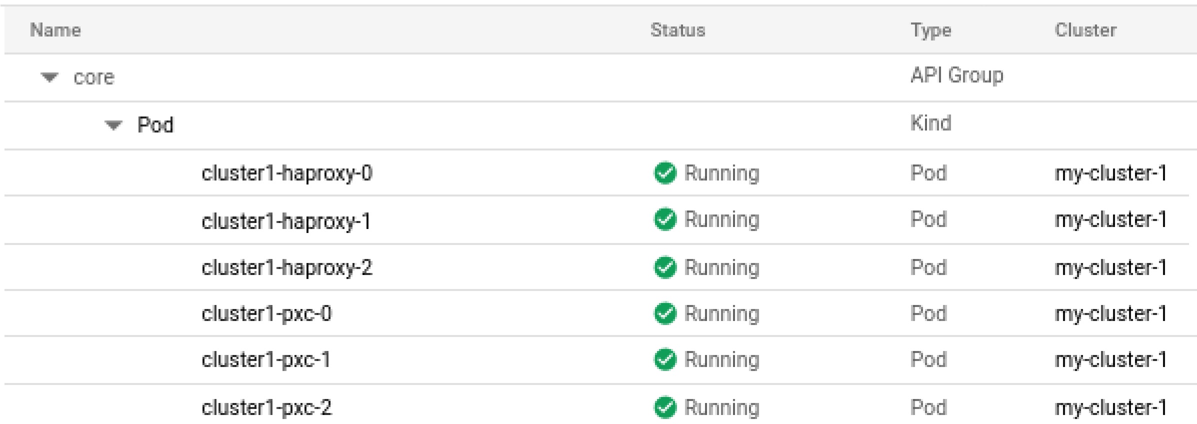
```
NAME                                               READY   STATUS    RESTARTS  ␣
↪ AGE
cluster1-haproxy-0                                 2/2     Running   0         ␣
↪ 6m17s
cluster1-haproxy-1                                 2/2     Running   0         ␣
↪ 4m59s
cluster1-haproxy-2                                 2/2     Running   0         ␣
↪ 4m36s
cluster1-pxc-0                                     3/3     Running   0         ␣
↪ 6m17s
cluster1-pxc-1                                     3/3     Running   0         ␣
↪ 5m3s
```

(continues on next page)

(continued from previous page)

```
cluster1-pxc-2                                     3/3     Running   0          3m56s
percona-xtradb-cluster-operator-79966668bd-rswbk   1/1     Running   0          9m54s
```

Also, you can see the same information when browsing Pods of your cluster in Google Cloud console via the *Object Browser*:



Name	Status	Type	Cluster
▼ core		API Group	
▼ Pod		Kind	
cluster1-haproxy-0	Running	Pod	my-cluster-1
cluster1-haproxy-1	Running	Pod	my-cluster-1
cluster1-haproxy-2	Running	Pod	my-cluster-1
cluster1-pxc-0	Running	Pod	my-cluster-1
cluster1-pxc-1	Running	Pod	my-cluster-1
cluster1-pxc-2	Running	Pod	my-cluster-1

If all nodes are up and running, you can try to connect to the cluster with the following command:

```
$ kubectl run -i --rm --tty percona-client --image=percona:8.0 --restart=Never -- bash -il
```

Executing this command will open a bash command prompt:

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

Now run mysql tool in the percona-client command shell using the password obtained from the secret:

```
mysql -h cluster1-haproxy -uroot -proot_password
```

This command will connect you to the MySQL monitor.

```
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1976
Server version: 8.0.19-10 Percona XtraDB Cluster (GPL), Release rel10, Revision 727f180, WSREP version 26.4.3

Copyright (c) 2009-2020 Percona LLC and/or its affiliates
Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

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

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

The following example will use the MySQL prompt to check the `max_connections` variable:

```
mysql> SHOW VARIABLES LIKE "max_connections";
```

The return statement displays the current max_connections.

```
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| max_connections | 79    |
+-----------------+-------+
1 row in set (0.02 sec)
```

4.5 Troubleshooting

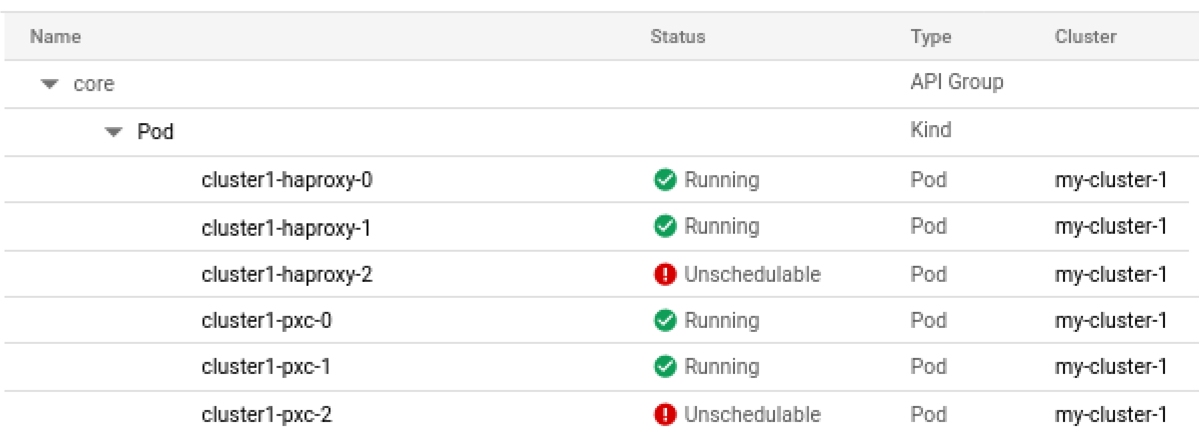
If `kubectl get pods` command had shown some errors, you can examine the problematic Pod with the `kubectl describe <pod name>` command. For example, this command returns information for the selected Pod:

```
kubectl describe pod cluster1-haproxy-2
```

Review the detailed information for `Warning` statements and then correct the configuration. An example of a warning is as follows:

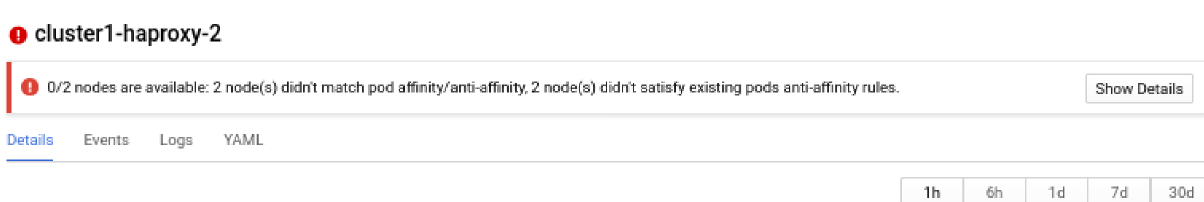
> *Warning FailedScheduling 68s (x4 over 2m22s) default-scheduler 0/1 nodes are available: 1 node(s) didn't match pod affinity/anti-affinity, 1 node(s) didn't satisfy existing pods anti-affinity rules.*

Alternatively, you can examine your Pods via the *object browser*. Errors will look as follows:



Name	Status	Type	Cluster
core		API Group	
Pod		Kind	
cluster1-haproxy-0	Running	Pod	my-cluster-1
cluster1-haproxy-1	Running	Pod	my-cluster-1
cluster1-haproxy-2	Unschedulable	Pod	my-cluster-1
cluster1-pxc-0	Running	Pod	my-cluster-1
cluster1-pxc-1	Running	Pod	my-cluster-1
cluster1-pxc-2	Unschedulable	Pod	my-cluster-1

Clicking the problematic Pod will bring you to the details page with the same warning:



4.6 Removing the GKE cluster

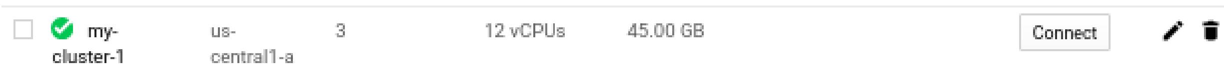
There are several ways that you can delete the cluster.

You can clean up the cluster with the `gcloud` command as follows:

```
gcloud container clusters delete <cluster name>
```

The return statement requests your confirmation of the deletion. Type `y` to confirm.

Also, you can delete your cluster via the GKE console. Just click the appropriate trashcan icon in the clusters list:



The cluster deletion may take time.

CHAPTER FIVE

INSTALL PERCONA XTRADB CLUSTER ON AMAZON ELASTIC KUBERNETES SERVICE (EKS)

This quickstart shows you how to deploy Percona Distribution for MySQL Operator on Amazon Elastic Kubernetes Service (EKS). The document assumes some experience with Amazon EKS. For more information on the EKS, see the Amazon EKS official documentation.

5.1 Prerequisites

The following tools are used in this guide and therefore should be preinstalled:

1. **AWS Command Line Interface (AWS CLI)** for interacting with the different parts of AWS. You can install it following the official installation instructions for your system.
2. **eksctl** to simplify cluster creation on EKS. It can be installed along its installation notes on GitHub.
3. **kubectl** to manage and deploy applications on Kubernetes. Install it following the official installation instructions.

Also, you need to configure AWS CLI with your credentials according to the official guide.

5.2 Create the EKS cluster

To create your cluster, you will need the following data:

- name of your EKS cluster,
- AWS region in which you wish to deploy your cluster,
- the amount of nodes you would like tho have,
- the amount of on-demand and spot instances to use.

Note: spot instances are not recommended for production environment, but may be useful e.g. for testing purposes.

The most easy and visually clear way is to describe the desired cluster in YAML and to pass this configuration to the `eksctl` command.

The following example configures a EKS cluster with one managed node group:

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
```

(continues on next page)

(continued from previous page)

```yaml
metadata:
    name: test-cluster
    region: eu-west-2

nodeGroups:
    - name: ng-1
      minSize: 3
      maxSize: 5
      instancesDistribution:
        maxPrice: 0.15
        instanceTypes: ["m5.xlarge", "m5.2xlarge"] # At least two instance types should␣
↪be specified
        onDemandBaseCapacity: 0
        onDemandPercentageAboveBaseCapacity: 50
        spotInstancePools: 2
      tags:
        'iit-billing-tag': 'cloud'
      preBootstrapCommands:
          - "echo 'OPTIONS=\"--default-ulimit nofile=1048576:1048576\"' >> /etc/
↪sysconfig/docker"
          - "systemctl restart docker"
```

Note: `preBootstrapCommands` section is used in the above example to increase the limits for the amount of opened files: this is important and shouldn't be omitted, taking into account the default EKS soft limit of 65536 files.

When the cluster configuration file is ready, you can actually create your cluster by the following command:

```
$ eksctl create cluster -f ~/cluster.yaml
```

5.3 Install the Operator

1. Create a namespace and set the context for the namespace. The resource names must be unique within the namespace and provide a way to divide cluster resources between users spread across multiple projects.

   So, create the namespace and save it in the namespace context for subsequent commands as follows (replace the `<namespace name>` placeholder with some descriptive name):

   ```
   $ kubectl create namespace <namespace name>
   $ kubectl config set-context $(kubectl config current-context) --namespace=
   ↪<namespace name>
   ```

   At success, you will see the message that namespace/<namespace name> was created, and the context was modified.

2. Use the following `git clone` command to download the correct branch of the percona-xtradb-cluster-operator repository:

   ```
   git clone -b v1.9.0 https://github.com/percona/percona-xtradb-cluster-operator
   ```

   After the repository is downloaded, change the directory to run the rest of the commands in this document:

```
cd percona-xtradb-cluster-operator
```

3. Deploy the Operator with the following command:

```
kubectl apply -f deploy/bundle.yaml
```

The following confirmation is returned:

```
customresourcedefinition.apiextensions.k8s.io/perconaxtradbclusters.pxc.percona.com␣
↪created
customresourcedefinition.apiextensions.k8s.io/perconaxtradbclusterbackups.pxc.
↪percona.com created
customresourcedefinition.apiextensions.k8s.io/perconaxtradbclusterrestores.pxc.
↪percona.com created
customresourcedefinition.apiextensions.k8s.io/perconaxtradbbackups.pxc.percona.com␣
↪created
role.rbac.authorization.k8s.io/percona-xtradb-cluster-operator created
serviceaccount/percona-xtradb-cluster-operator created
rolebinding.rbac.authorization.k8s.io/service-account-percona-xtradb-cluster-
↪operator created
deployment.apps/percona-xtradb-cluster-operator created
```

4. The operator has been started, and you can create the Percona XtraDB cluster:

```
$ kubectl apply -f deploy/cr.yaml
```

The process could take some time. The return statement confirms the creation:

```
perconaxtradbcluster.pxc.percona.com/cluster1 created
```

5. During previous steps, the Operator has generated several secrets, including the password for the `root` user, which you will need to access the cluster.

Use `kubectl get secrets` command to see the list of Secrets objects (by default Secrets object you are interested in has `my-cluster-secrets` name). Then `kubectl get secret my-cluster-secrets -o yaml` will return the YAML file with generated secrets, including the root password which should look as follows:

```
...
data:
  ...
  root: cm9vdF9wYXNzd29yZA==
```

Here the actual password is base64-encoded, and `echo 'cm9vdF9wYXNzd29yZA==' | base64 --decode` will bring it back to a human-readable form (in this example it will be a `root_password` string).

6. Now you can check wether you are able to connect to MySQL from the outside with the help of the `kubectl port-forward` command as follows:

```
kubectl port-forward svc/example-proxysql 3306:3306 &
mysql -h 127.0.0.1 -P 3306 -uroot -proot_password
```

Part III

Advanced Installation Guides

CHAPTER

SIX

INSTALL PERCONA XTRADB CLUSTER ON KUBERNETES

1. First of all, clone the percona-xtradb-cluster-operator repository:

```
git clone -b v1.9.0 https://github.com/percona/percona-xtradb-cluster-operator
cd percona-xtradb-cluster-operator
```

Note: It is crucial to specify the right branch with `-b` option while cloning the code on this step. Please be careful.

2. Now Custom Resource Definition for Percona XtraDB Cluster should be created from the `deploy/crd.yaml` file. Custom Resource Definition extends the standard set of resources which Kubernetes "knows" about with the new items (in our case ones which are the core of the operator).

   This step should be done only once; it does not need to be repeated with the next Operator deployments, etc.

```
$ kubectl apply -f deploy/crd.yaml
```

3. The next thing to do is to add the `pxc` namespace to Kubernetes, not forgetting to set the correspondent context for further steps:

```
$ kubectl create namespace pxc
$ kubectl config set-context $(kubectl config current-context) --namespace=pxc
```

4. Now RBAC (role-based access control) for Percona XtraDB Cluster should be set up from the `deploy/rbac.yaml` file. Briefly speaking, role-based access is based on specifically defined roles and actions corresponding to them, allowed to be done on specific Kubernetes resources (details about users and roles can be found in Kubernetes documentation).

```
$ kubectl apply -f deploy/rbac.yaml
```

Note: Setting RBAC requires your user to have cluster-admin role privileges. For example, those using Google Kubernetes Engine can grant user needed privileges with the following command: `$ kubectl create clusterrolebinding cluster-admin-binding --clusterrole=cluster-admin --user=$(gcloud config get-value core/account)`

   Finally it's time to start the operator within Kubernetes:

```
$ kubectl apply -f deploy/operator.yaml
```

5. Now that's time to add the Percona XtraDB Cluster Users secrets to Kubernetes. They should be placed in the data section of the `deploy/secrets.yaml` file as logins and plaintext passwords for the user accounts (see Kubernetes documentation for details).

   After editing is finished, users secrets should be created using the following command:

   ```
   $ kubectl create -f deploy/secrets.yaml
   ```

   More details about secrets can be found in *Users*.

6. Now certificates should be generated. By default, the Operator generates certificates automatically, and no actions are required at this step. Still, you can generate and apply your own certificates as secrets according to the *TLS instructions*.

7. After the operator is started and user secrets are added, Percona XtraDB Cluster can be created at any time with the following command:

   ```
   $ kubectl apply -f deploy/cr.yaml
   ```

   Creation process will take some time. The process is over when both operator and replica set pod have reached their Running status:

   ```
   NAME                                               READY   STATUS    RESTARTS   AGE
   cluster1-haproxy-0                                 2/2     Running   0          ␣
   ↪6m17s
   cluster1-haproxy-1                                 2/2     Running   0          ␣
   ↪4m59s
   cluster1-haproxy-2                                 2/2     Running   0          ␣
   ↪4m36s
   cluster1-pxc-0                                     3/3     Running   0          ␣
   ↪6m17s
   cluster1-pxc-1                                     3/3     Running   0           5m3s
   cluster1-pxc-2                                     3/3     Running   0          ␣
   ↪3m56s
   percona-xtradb-cluster-operator-79966668bd-rswbk   1/1     Running   0          ␣
   ↪9m54s
   ```

8. Check connectivity to newly created cluster

   ```
   $ kubectl run -i --rm --tty percona-client --image=percona:8.0 --restart=Never --␣
   ↪bash -il
   percona-client:/$ mysql -h cluster1-haproxy -uroot -proot_password
   ```

   This command will connect you to the MySQL monitor.

   ```
   mysql: [Warning] Using a password on the command line interface can be insecure.
   Welcome to the MySQL monitor.  Commands end with ; or \g.
   Your MySQL connection id is 1976
   Server version: 8.0.19-10 Percona XtraDB Cluster (GPL), Release rel10, Revision␣
   ↪727f180, WSREP version 26.4.3

   Copyright (c) 2009-2020 Percona LLC and/or its affiliates
   Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

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

(continues on next page)

(continued from previous page)

```
owners.

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

CHAPTER SEVEN

INSTALL PERCONA XTRADB CLUSTER ON OPENSHIFT

Percona Operator for Percona XtrabDB Cluster is a Red Hat Certified Operator. This means that Percona Operator is portable across hybrid clouds and fully supports the Red Hat OpenShift lifecycle.

Installing Percona XtraDB Cluster on OpenShift includes two steps:

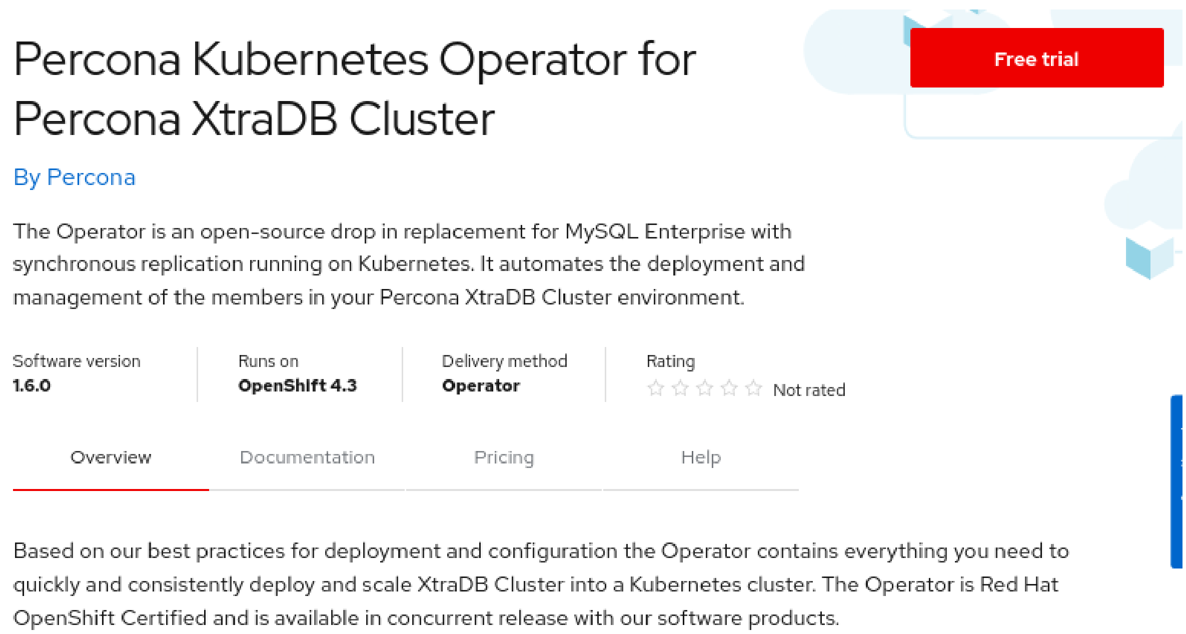
- Installing the Percona Distribution for MySQL Operator,
- Install Percona XtraDB Cluster using the Operator.

7.1 Install the Operator

You can install Percona Distribution for MySQL Operator on OpenShift using the Red Hat Marketplace web interface or using the command line interface.

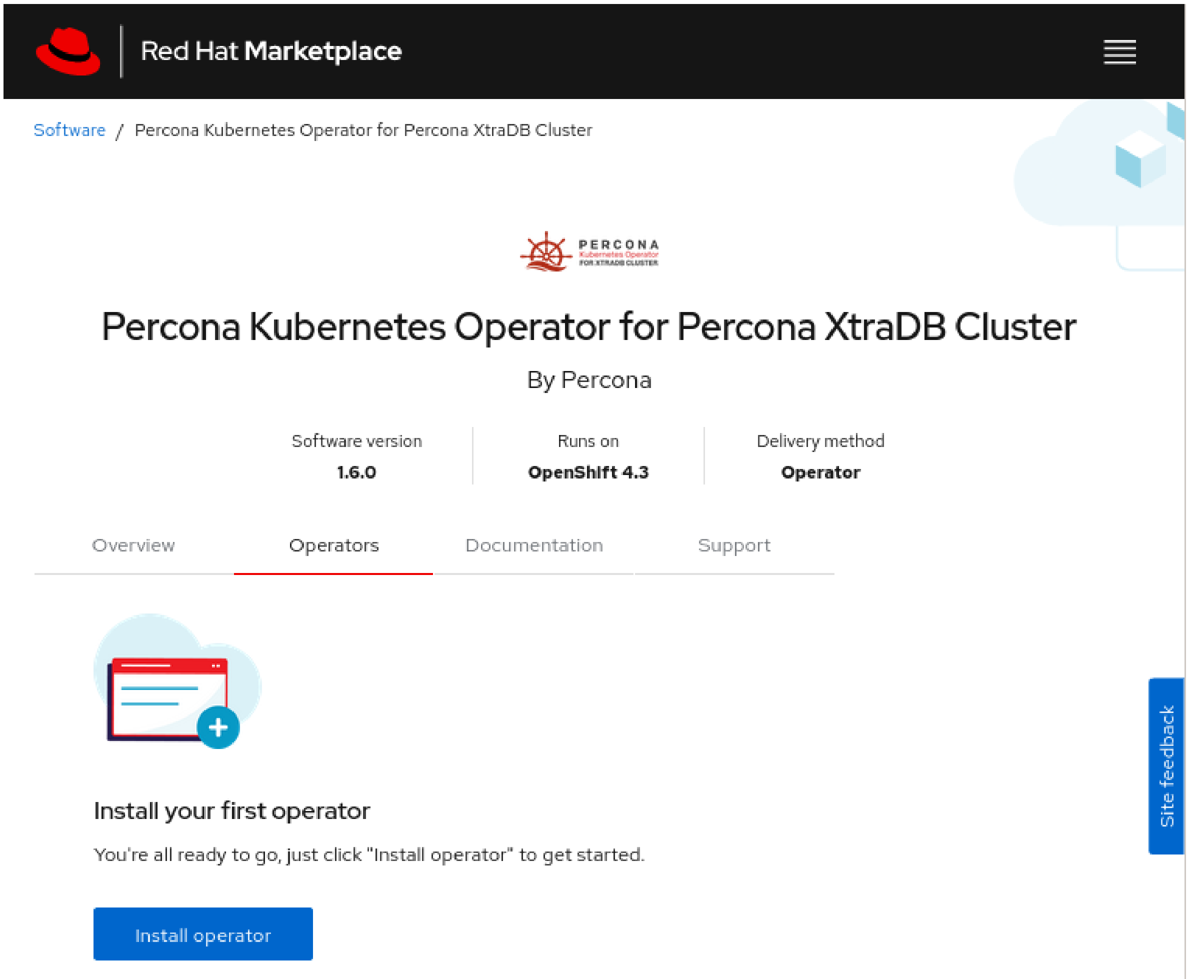
7.1.1 Install the Operator via the Red Hat Marketplace

1. login to the Red Hat Marketplace and register your cluster following the official instructions.
2. Go to the Percona Distribution for MySQL Operator page and click the *Free trial* button:



Here you can "start trial" of the Operator for 0.0 USD.

3. When finished, chose `Workspace->Software` in the system menu on the top and choose the Operator:



Click the `Install Operator` button.

7.1.2 Install the Operator via the command-line interface

1. Clone the percona-xtradb-cluster-operator repository:

```
git clone -b v1.9.0 https://github.com/percona/percona-xtradb-cluster-operator
cd percona-xtradb-cluster-operator
```

Note: It is crucial to specify the right branch with the-*b* option while cloning the code on this step. Please be careful.

2. Now Custom Resource Definition for Percona XtraDB Cluster should be created from the `deploy/crd.yaml` file. Custom Resource Definition extends the standard set of resources which Kubernetes "knows" about with the new items (in our case ones which are the core of the operator).

   This step should be done only once; it does not need to be repeated with the next Operator deployments, etc.

```
$ oc apply -f deploy/crd.yaml
```

Note: Setting Custom Resource Definition requires your user to have cluster-admin role privileges.

If you want to manage your Percona XtraDB Cluster with a non-privileged user, necessary permissions can be granted by applying the next clusterrole:

```
$ oc create clusterrole pxc-admin --verb="*" --resource=perconaxtradbclusters.pxc.
↪percona.com,perconaxtradbclusters.pxc.percona.com/status,
↪perconaxtradbclusterbackups.pxc.percona.com,perconaxtradbclusterbackups.pxc.
↪percona.com/status,perconaxtradbclusterrestores.pxc.percona.com,
↪perconaxtradbclusterrestores.pxc.percona.com/status
$ oc adm policy add-cluster-role-to-user pxc-admin <some-user>
```

If you have a cert-manager installed, then you have to execute two more commands to be able to manage certificates with a non-privileged user:

```
$ oc create clusterrole cert-admin --verb="*" --resource=issuers.certmanager.k8s.io,
↪certificates.certmanager.k8s.io
$ oc adm policy add-cluster-role-to-user cert-admin <some-user>
```

3. The next thing to do is to create a new `pxc` project:

```
$ oc new-project pxc
```

4. Now RBAC (role-based access control) for Percona XtraDB Cluster should be set up from the `deploy/rbac.yaml` file. Briefly speaking, role-based access is based on specifically defined roles and actions corresponding to them, allowed to be done on specific Kubernetes resources (details about users and roles can be found in OpenShift documentation).

```
$ oc apply -f deploy/rbac.yaml
```

Finally, it's time to start the operator within OpenShift:

```
$ oc apply -f deploy/operator.yaml
```

7.2 Install Percona XtraDB Cluster

1. Now that's time to add the Percona XtraDB Cluster Users secrets to OpenShift. They should be placed in the data section of the `deploy/secrets.yaml` file as logins and plaintext passwords for the user accounts (see Kubernetes documentation for details).

After editing is finished, users secrets should be created using the following command:

```
$ oc create -f deploy/secrets.yaml
```

More details about secrets can be found in *Users*.

2. Now certificates should be generated. By default, the Operator generates certificates automatically, and no actions are required at this step. Still, you can generate and apply your own certificates as secrets according to the *TLS instructions*.

3. After the operator is started and user secrets are added, Percona XtraDB Cluster can be created at any time with the following command:

```
$ oc apply -f deploy/cr.yaml
```

Creation process will take some time. The process is over when both operator and replica set pod have reached their Running status:

```
NAME                                               READY   STATUS    RESTARTS   AGE
cluster1-haproxy-0                                 2/2     Running   0          ␣
↪6m17s
cluster1-haproxy-1                                 2/2     Running   0          ␣
↪4m59s
cluster1-haproxy-2                                 2/2     Running   0          ␣
↪4m36s
cluster1-pxc-0                                     3/3     Running   0          ␣
↪6m17s
cluster1-pxc-1                                     3/3     Running   0           5m3s
cluster1-pxc-2                                     3/3     Running   0          ␣
↪3m56s
percona-xtradb-cluster-operator-79966668bd-rswbk   1/1     Running   0          ␣
↪9m54s
```

4. Check connectivity to newly created cluster

```
$ oc run -i --rm --tty percona-client --image=percona:8.0 --restart=Never -- bash -
↪il
percona-client:/$ mysql -h cluster1-haproxy -uroot -proot_password
```

This command will connect you to the MySQL monitor.

```
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1976
Server version: 8.0.19-10 Percona XtraDB Cluster (GPL), Release rel10, Revision␣
↪727f180, WSREP version 26.4.3

Copyright (c) 2009-2020 Percona LLC and/or its affiliates
Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

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

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

CHAPTER

EIGHT

USE DOCKER IMAGES FROM A CUSTOM REGISTRY

Using images from a private Docker registry may be useful in different situations: it may be related to storing images inside of a company, for privacy and security reasons, etc. In such cases, Percona Distribution for MySQL Operator allows to use a custom registry, and the following instruction illustrates how this can be done by the example of the Operator deployed in the OpenShift environment.

1. First of all login to the OpenShift and create project.

```
$ oc login
Authentication required for https://192.168.1.100:8443 (openshift)
Username: admin
Password:
Login successful.
$ oc new-project pxc
Now using project "pxc" on server "https://192.168.1.100:8443".
```

2. There are two things you will need to configure your custom registry access:

   - the token for your user
   - your registry IP address.

   The token can be find out with the following command:

```
$ oc whoami -t
ADO8CqCDappWR4hxjfDqwijEHei31yXAvWg61Jg210s
```

   And the following one tells you the registry IP address:

```
$ kubectl get services/docker-registry -n default
NAME              TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
docker-registry   ClusterIP   172.30.162.173   <none>        5000/TCP   1d
```

3. Now you can use the obtained token and address to login to the registry:

```
$ docker login -u admin -p ADO8CqCDappWR4hxjfDqwijEHei31yXAvWg61Jg210s 172.30.162.
↪173:5000
Login Succeeded
```

4. Pull the needed image by its SHA digest:

```
$ docker pull docker.io/perconalab/percona-xtradb-cluster-
↪operator@sha256:841c07eef30605080bfe80e549f9332ab6b9755fcbc42aacbf86e4ac9ef0e444
Trying to pull repository docker.io/perconalab/percona-xtradb-cluster-operator ...
```

(continues on next page)

(continued from previous page)

```
sha256:841c07eef30605080bfe80e549f9332ab6b9755fcbc42aacbf86e4ac9ef0e444: Pulling␣
↪from docker.io/perconalab/percona-xtradb-cluster-operator
Digest: sha256:841c07eef30605080bfe80e549f9332ab6b9755fcbc42aacbf86e4ac9ef0e444
Status: Image is up to date for docker.io/perconalab/percona-xtradb-cluster-
↪operator@sha256:841c07eef30605080bfe80e549f9332ab6b9755fcbc42aacbf86e4ac9ef0e444
```

You can find correct names and SHA digests in the *current list of the Operator-related images officially certified by Percona*.

5. The following way is used to push an image to the custom registry (into the OpenShift pxc project):

```
$ docker tag \
    docker.io/perconalab/percona-xtradb-cluster-
↪operator@sha256:841c07eef30605080bfe80e549f9332ab6b9755fcbc42aacbf86e4ac9ef0e444 \
    172.30.162.173:5000/pxc/percona-xtradb-cluster-operator:1.2.0
$ docker push 172.30.162.173:5000/pxc/percona-xtradb-cluster-operator:1.2.0
```

6. Check the image in the OpenShift registry with the following command:

```
$ oc get is
NAME                              DOCKER REPO                                      ␣
↪                       TAGS      UPDATED
percona-xtradb-cluster-operator   docker-registry.default.svc:5000/pxc/percona-
↪xtradb-cluster-operator   1.9.0     2 hours ago
```

7. When the custom registry image is Ok, put a Docker Repo + Tag string (it should look like `docker-registry.default.svc:5000/pxc/percona-xtradb-cluster-operator:1.9.0`) into the `image:` option in `deploy/operator.yaml` configuration file.

   Please note it is possible to specify `imagePullSecrets` option for all images, if the registry requires authentication.

8. Repeat steps 3-5 for other images, and update corresponding options in the `deploy/cr.yaml` file.

9. Now follow the standard Percona Distribution for MySQL Operator installation instruction.

CHAPTER

NINE

DEPLOY PERCONA XTRADB CLUSTER WITH SERVICE BROKER

Percona Service Broker provides the Open Service Broker object to facilitate the operator deployment within high-level visual tools. Following steps are needed to use it while installing the Percona XtraDB Cluster on the OpenShift platform:

1. The Percona Service Broker is to be deployed based on the `percona-broker.yaml` file. To use it you should first enable the Service Catalog, which can be done with the following command:

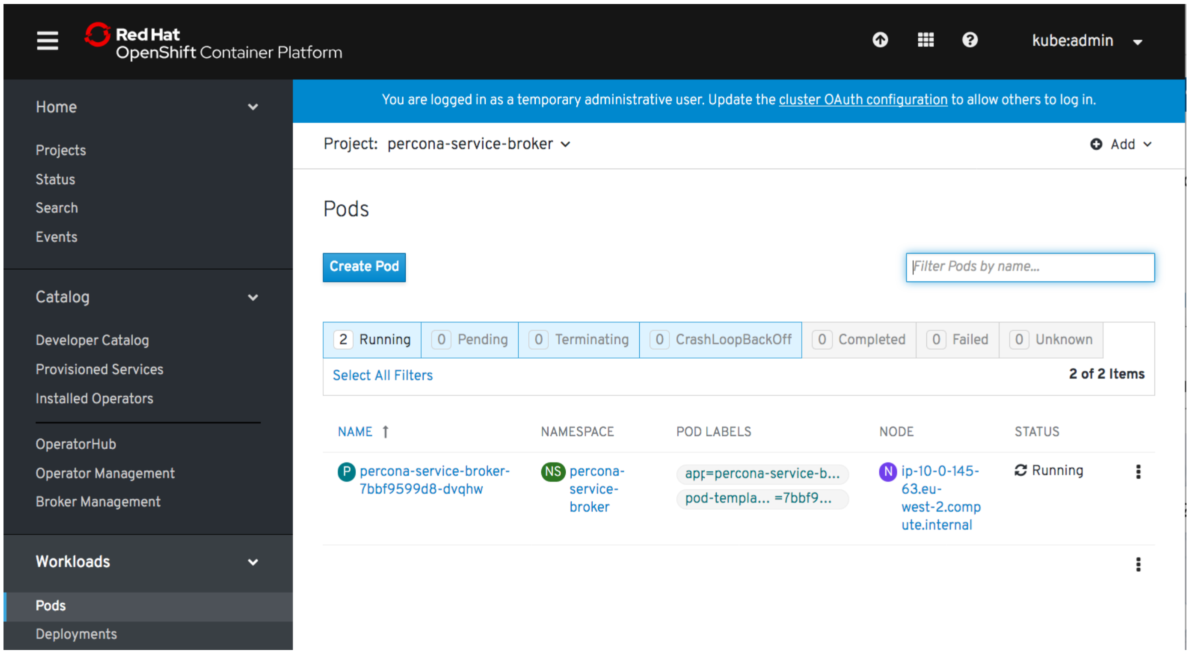
```
$ oc patch servicecatalogapiservers cluster --patch '{"spec":{"managementState":
↪"Managed"}}' --type=merge
$ oc patch servicecatalogcontrollermanagers cluster --patch '{"spec":{
↪"managementState":"Managed"}}' --type=merge
```

   When Service Catalog is enabled, download and install the Percona Service Broker in a typical OpenShift way:

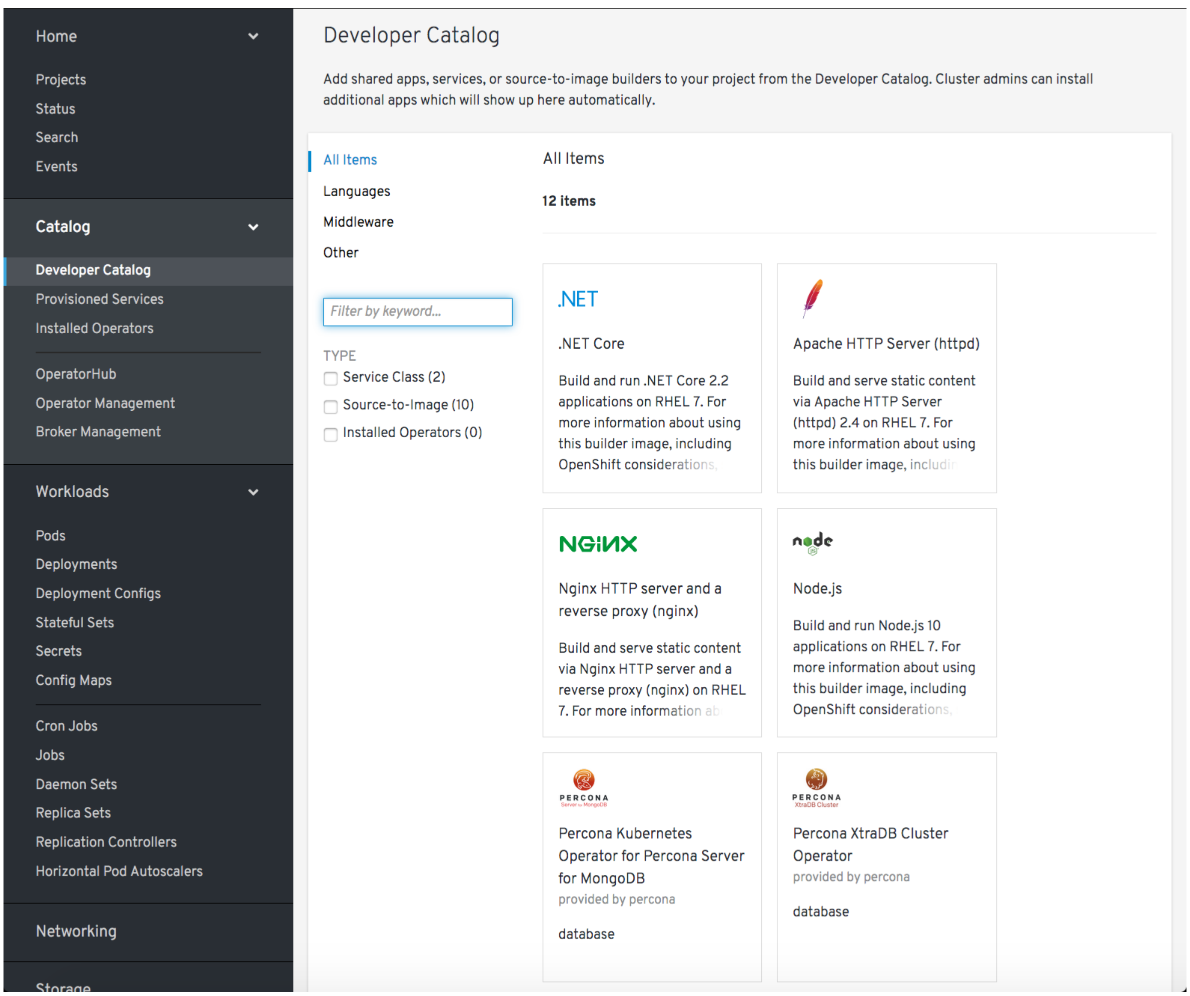
```
$ oc apply -f https://raw.githubusercontent.com/Percona-Lab/percona-dbaas-cli/
↪broker/deploy/percona-broker.yaml
```

   Note: This step should be done only once; the step does not need to be repeated with any other Operator deployments. It will automatically create and setup the needed service and projects catalog with all necessary objects.

2. Now login to your OpenShift Console Web UI and switch to the percona-service-broker project. You can check its Pod running on a correspondent page:

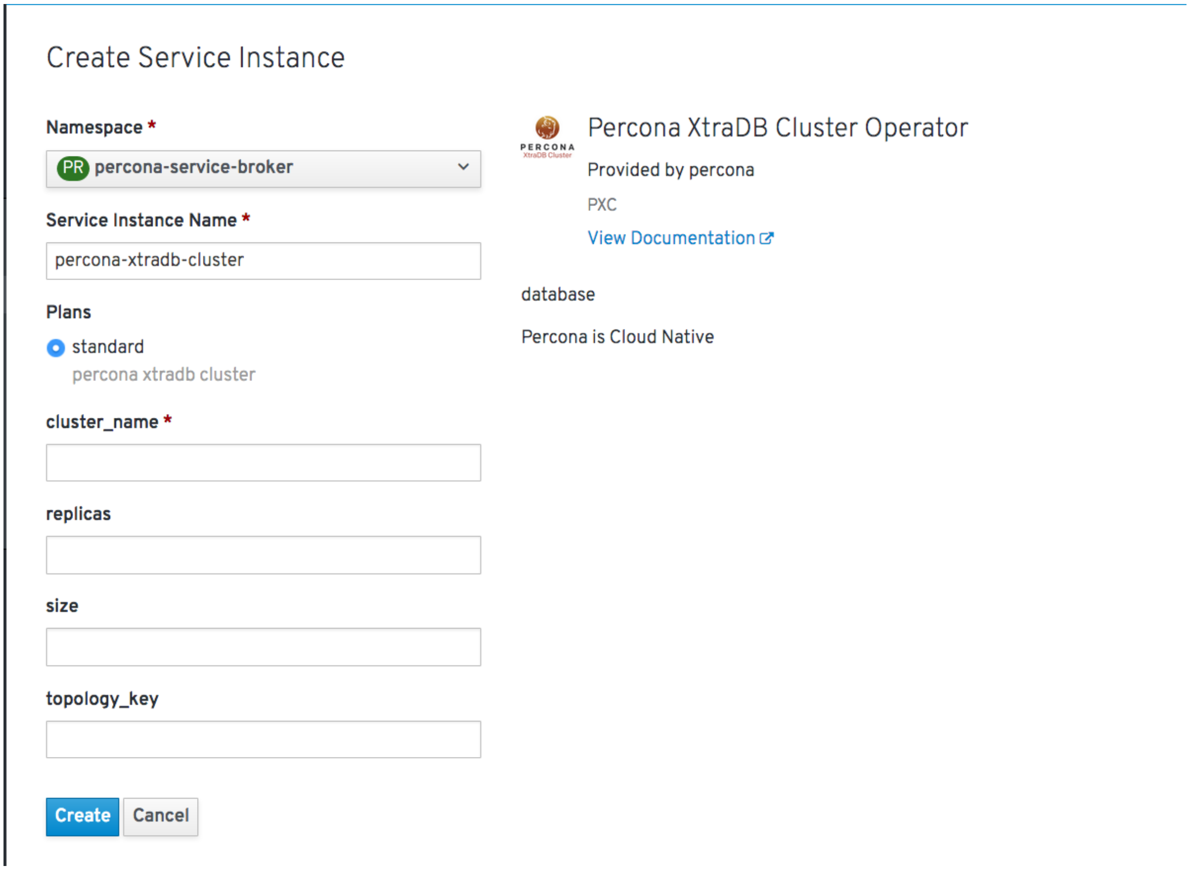


Now switch to the Developer Catalog and select Percona Distribution for MySQL Operator:



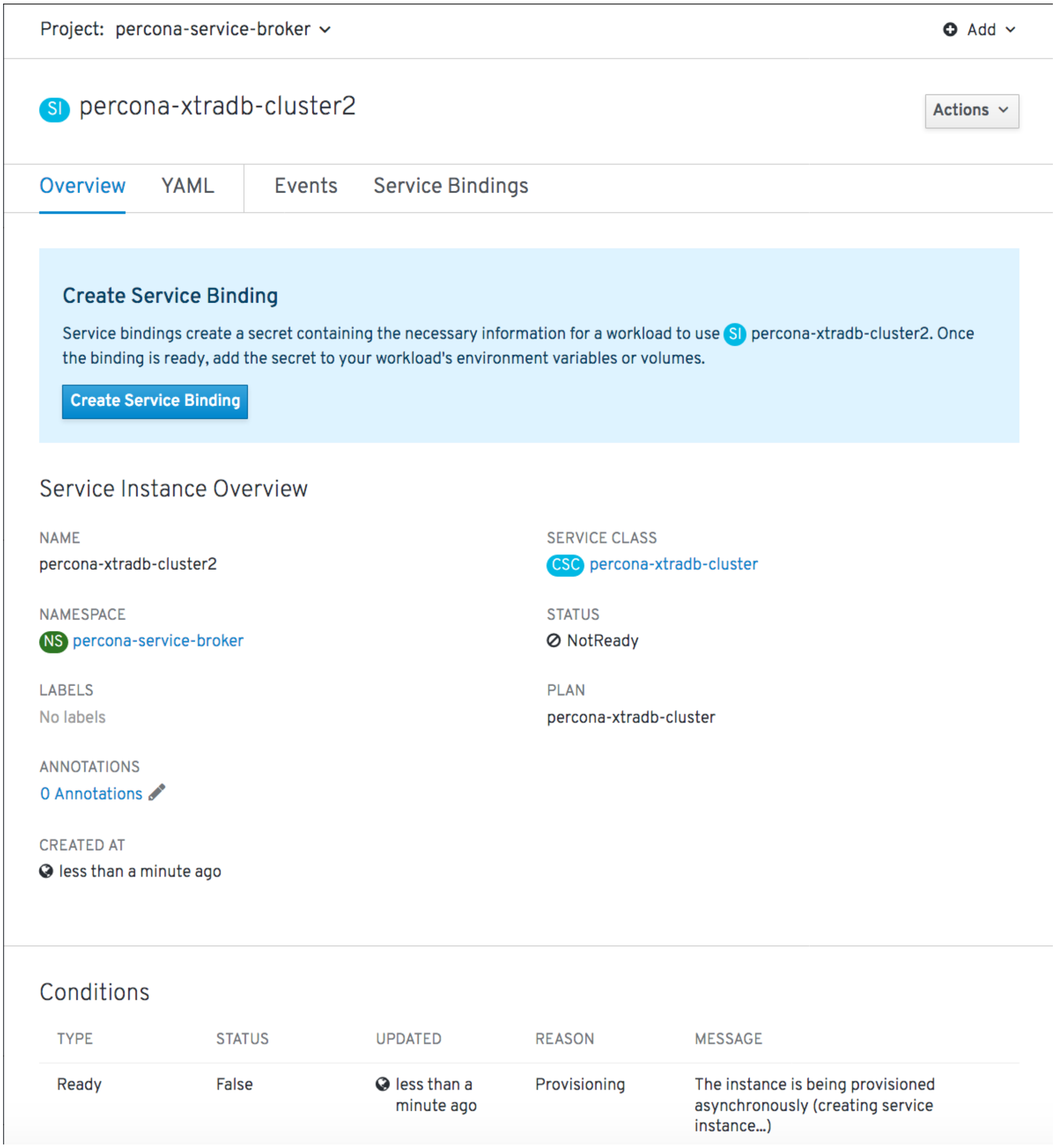
Choose `Percona XtraDB Cluster Operator` item. This will lead you to the Operator page with the *Create Service Instance* button.

3. Clicking the *Create Service Instance* button guides you to the next page:



The two necessary fields are *Service Instance Name* and *Cluster Name*, which should be unique for your project.

4. Clicking the *Create* button gets you to the *Overview* page, which reflects the process of the cluster creation process:



You can also track Pods to see when they are deployed and track any errors.

CHAPTER

TEN

INSTALL PERCONA XTRADB CLUSTER USING HELM

Helm is the package manager for Kubernetes.

10.1 Pre-requisites

Install Helm following its official installation instructions.

Note: Helm v3 is needed to run the following steps.

10.2 Installation

1. Add the Percona's Helm charts repository and make your Helm client up to date with it:

```
helm repo add percona https://percona.github.io/percona-helm-charts/
helm repo update
```

2. Install the Percona Distribution for MySQL Operator:

```
helm install my-op percona/pxc-operator
```

The `my-op` parameter in the above example is the name of a new release object which is created for the Operator when you install its Helm chart (use any name you like).

Note: If nothing explicitly specified, `helm install` command will work with `default` namespace. To use different namespace, provide it with the following additional parameter: `--namespace my-namespace`.

3. Install Percona XtraDB Cluster:

```
helm install my-db percona/pxc-db
```

The `my-db` parameter in the above example is the name of a new release object which is created for the Percona XtraDB Cluster when you install its Helm chart (use any name you like).

10.3 Installing Percona XtraDB Cluster with customized parameters

The command above installs Percona XtraDB Cluster with *default parameters*. Custom options can be passed to a `helm install` command as a `--set key=value[,key=value]` argument. The options passed with a chart can be any of the Operator's *Custom Resource options*.

The following example will deploy a Percona XtraDB Cluster Cluster in the `pxc` namespace, with disabled backups and 20 Gi storage:

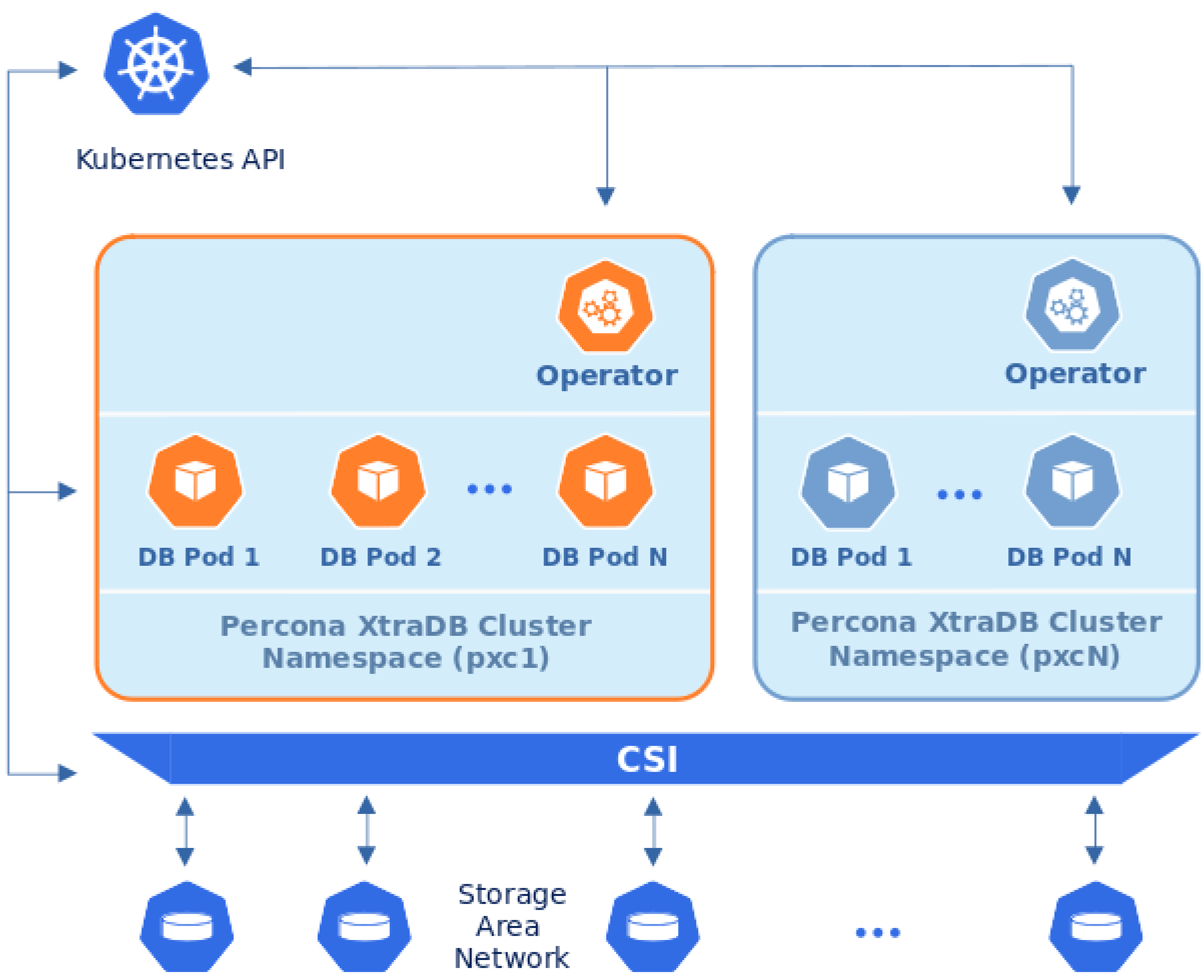
```
helm install my-db percona/pxc-db --namespace pxc \
  --set pxc.volumeSpec.resources.requests.storage=20Gi \
  --set backup.enabled=false
```

CHAPTER

ELEVEN

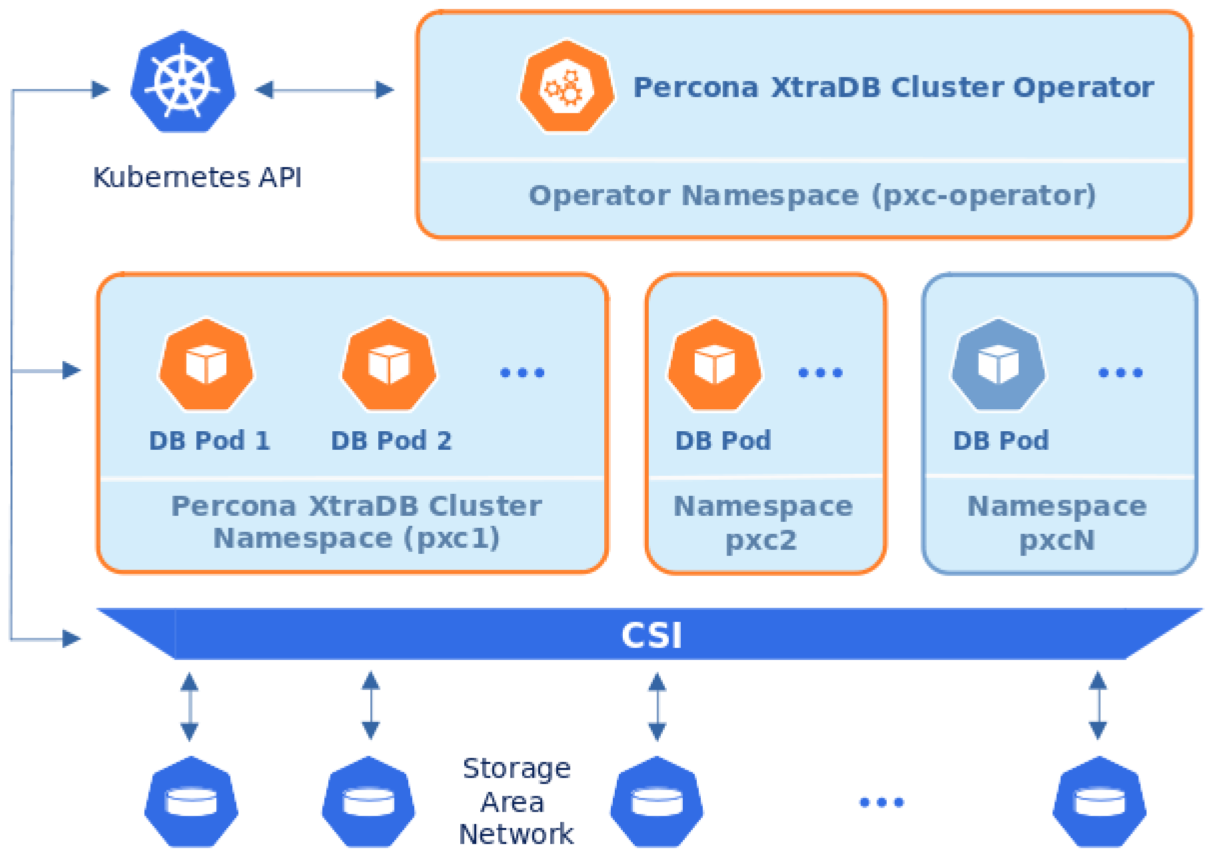
INSTALL PERCONA XTRADB CLUSTER CLUSTER-WIDE

By default, Percona XtraDB Cluster Operator functions in a specific Kubernetes namespace. You can create one during installation (like it is shown in the *installation instructions*) or just use the `default` namespace. This approach allows several Operators to co-exist in one Kubernetes-based environment, being separated in different namespaces:



Still, sometimes it is more convenient to have one Operator watching for Percona XtraDB Cluster custom resources in several namespaces.

We recommend running Percona XtraDB Cluster Operator in a traditional way, limited to a specific namespace. But it is possible to run it in so-called *cluster-wide* mode, one Operator watching several namespaces, if needed:



Note: Please take into account that if several Operators are configured to watch the same namespace, it is entirely unpredictable which one will get ownership of the Custom Resource in it, so this situation should be avoided.

To use the Operator in such *cluster-wide* mode, you should install it with a different set of configuration YAML files, which are available in the `deploy` folder and have filenames with a special `cw-` prefix: e.g. `deploy/cw-bundle.yaml`.

While using this cluster-wide versions of configuration files, you should set the following information there:

- `subjects.namespace` option should contain the namespace which will host the Operator,
- `WATCH_NAMESPACE` key-value pair in the `env` section should have `value` equal to a comma-separated list of the namespaces to be watched by the Operator, *and* the namespace in which the Operator resides (or just a blank string to make the Operator deal with *all namespaces* in a Kubernetes cluster).

  Note: The list of namespaces to watch is fully supported by Percona Distribution for MySQL Operator starting from the version 1.7. In version 1.6 you can only use cluster-wide mode with empty `WATCH_NAMESPACE` key to watch all namespaces.

The following simple example shows how to install Operator cluster-wide on Kubernetes.

1. First of all, clone the percona-xtradb-cluster-operator repository:

```
git clone -b v1.9.0 https://github.com/percona/percona-xtradb-cluster-operator
cd percona-xtradb-cluster-operator
```

2. Let's suppose that Operator's namespace should be the `pxc-operator` one. Create it as follows:

```
$ kubectl create namespace pxc-operator
```

   Namespaces to be watched by the Operator should be created in the same way if not exist. Let's say the Operator should watch the `pxc` namespace:

```
$ kubectl create namespace pxc
```

3. Apply the `deploy/cw-bundle.yaml` file with the following command:

```
$ kubectl apply -f deploy/cw-bundle.yaml -n pxc-operator
```

4. After the Operator is started, Percona XtraDB Cluster can be created at any time by applying the `deploy/cr.yaml` configuration file, like in the case of normal installation:

```
$ kubectl apply -f deploy/cr.yaml -n pxc
```

   The creation process will take some time. The process is over when both operator and replica set Pods have reached their Running status:

```
NAME                                               READY   STATUS    RESTARTS   AGE
cluster1-haproxy-0                                 2/2     Running   0          ␣
↪6m17s
cluster1-haproxy-1                                 2/2     Running   0          ␣
↪4m59s
cluster1-haproxy-2                                 2/2     Running   0          ␣
↪4m36s
cluster1-pxc-0                                     3/3     Running   0          ␣
↪6m17s
cluster1-pxc-1                                     3/3     Running   0          5m3s
cluster1-pxc-2                                     3/3     Running   0          ␣
↪3m56s
percona-xtradb-cluster-operator-79966668bd-rswbk   1/1     Running   0          ␣
↪9m54s
```

5. Check connectivity to newly created cluster

```
$ kubectl run -i --rm --tty percona-client --image=percona:5.7 --restart=Never --
↪env="POD_NAMESPACE=pxc" -- bash -il
percona-client:/$ mysql -h cluster1-proxysql -uroot -proot_password
```

Part IV

Configuration

CHAPTER

TWELVE

USERS

MySQL user accounts within the Cluster can be divided into two different groups:

- *application-level users*: the unprivileged user accounts,
- *system-level users*: the accounts needed to automate the cluster deployment and management tasks, such as Percona XtraDB Cluster Health checks or ProxySQL integration.

As these two groups of user accounts serve different purposes, they are considered separately in the following sections.

12.1 Unprivileged users

There are no unprivileged (general purpose) user accounts created by default. If you need general purpose users, please run commands below:

```
$ kubectl run -it --rm percona-client --image=percona:8.0 --restart=Never -- mysql -
↪hcluster1-pxc -uroot -proot_password
mysql> GRANT ALL PRIVILEGES ON database1.* TO 'user1'@'%' IDENTIFIED BY 'password1';
```

Note: MySQL password here should not exceed 32 characters due to the replication-specific limit introduced in MySQL 5.7.5.

Verify that the user was created successfully. If successful, the following command will let you successfully login to MySQL shell via ProxySQL:

```
$ kubectl run -it --rm percona-client --image=percona:8.0 --restart=Never -- bash -il
percona-client:/$ mysql -h cluster1-proxysql -uuser1 -ppassword1
mysql> SELECT * FROM database1.table1 LIMIT 1;
```

You may also try executing any simple SQL statement to ensure the permissions have been successfully granted.

12.2 System Users

To automate the deployment and management of the cluster components, the Operator requires system-level Percona XtraDB Cluster users.

Credentials for these users are stored as a Kubernetes Secrets object. The Operator requires to be deployed before the Percona XtraDB Cluster is started. The name of the required secrets (`my-cluster-secrets` by default) should be set in the `spec.secretsName` option of the `deploy/cr.yaml` configuration file.

The following table shows system users' names and purposes.

> **Warning:** These users should not be used to run an application.

User Purpose	Username	Password Secret Key	Description
Admin	root	root	Database administrative user, can be used by the application if needed
ProxySQLAdmin	proxyadmin	proxyadmin	ProxySQL administrative user, can be used to add general-purpose ProxySQL users
Backup	xtrabackup	xtrabackup	User to run backups
Cluster Check	clustercheck	clustercheck	User for liveness checks and readiness checks
Monitoring	monitor	monitor	User for internal monitoring purposes and PMM agent
PMM Server Password	should be set through the operator options	pmmserver	Password used to access PMM Server
Operator Admin	operator	operator	Database administrative user, should be used only by the Operator
Replication	replication	replication	Administrative user needed for cross-site Percona XtraDB Cluster

12.2.1 YAML Object Format

The default name of the Secrets object for these users is `my-cluster-secrets` and can be set in the CR for your cluster in `spec.secretName` to something different. When you create the object yourself, it should match the following simple format:

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: my-cluster-secrets
type: Opaque
stringData:
  root: root_password
  xtrabackup: backup_password
  monitor: monitory
  clustercheck: clustercheckpassword
  proxyadmin: admin_password
  pmmserver: supa|^|pazz
  operator: operatoradmin
  replication: repl_password
```

The example above matches *what is shipped in deploy/secrets.yaml* which contains default passwords. You should NOT use these in production, but they are present to assist in automated testing or simple use in a development environment.

As you can see, because we use the `stringData` type when creating the Secrets object, all values for each key/value pair are stated in plain text format convenient from the user's point of view. But the resulting Secrets object contains passwords stored as `data` - i.e., base64-encoded strings. If you want to update any field, you'll need to encode the value into base64 format. To do this, you can run `echo -n "password" | base64` in your local shell to get valid values. For example, setting the PMM Server user's password to `new_password` in the `my-cluster-name-secrets` object can be done with the following command:

```
kubectl patch secret/my-cluster-name-secrets -p '{"data":{"pmmserver": '$(echo -n new_
↪password | base64)'}}'
```

12.2.2 Password Rotation Policies and Timing

When there is a change in user secrets, the Operator creates the necessary transaction to change passwords. This rotation happens almost instantly (the delay can be up to a few seconds), and it's not needed to take any action beyond changing the password.

Note: Please don't change `secretName` option in CR, make changes inside the secrets object itself.

12.2.3 Marking System Users In MySQL

Starting with MySQL 8.0.16, a new feature called Account Categories has been implemented, which allows us to mark our system users as such. See the official documentation on this feature for more details.

12.3 Development Mode

To make development and testing easier, `deploy/secrets.yaml` secrets file contains default passwords for Percona XtraDB Cluster system users.

These development mode credentials from `deploy/secrets.yaml` are:

Secret Key	Secret Value
root	`root_password`
xtrabackup	`backup_password`
monitor	`monitor`
clustercheck	`clustercheckpassword`
proxyuser	`s3cret`
proxyadmin	`admin_password`
pmmserver	`supa|^|pazz`
operator	`operatoradmin`
replication	`repl_password`

Warning: Do not use the default Percona XtraDB Cluster user passwords in production!

CHAPTER

THIRTEEN

LOCAL STORAGE SUPPORT FOR THE PERCONA DISTRIBUTION FOR MYSQL OPERATOR

Among the wide rage of volume types, supported by Kubernetes, there are two which allow Pod containers to access part of the local filesystem on the node. Two such options are *emptyDir* and *hostPath* volumes.

13.1 emptyDir

The name of this option is self-explanatory. When Pod having an emptyDir volume is assigned to a Node, a directory with the specified name is created on this node and exists until this Pod is removed from the node. When the Pod have been deleted, the directory is deleted too with all its content. All containers in the Pod which have mounted this volume will gain read and write access to the correspondent directory.

The `emptyDir` options in the deploy/cr.yaml file can be used to turn the emptyDir volume on by setting the directory name.

13.2 hostPath

A hostPath volume mounts some existing file or directory from the node's filesystem into the Pod.

The `volumeSpec.hostPath` subsection in the deploy/cr.yaml file may include `path` and `type` keys to set the node's filesystem object path and to specify whether it is a file, a directory, or something else (e.g. a socket):

```
volumeSpec:
  hostPath:
    path: /data
    type: Directory
```

Please note, that hostPath directory is not created automatically! Is should be created manually and should have following correct attributives: 1. access permissions 2. ownership 3. SELinux security context

`hostPath` is useful when you are able to perform manual actions during the first run and have strong need in improved disk performance. Also, please consider using tolerations to avoid cluster migration to different hardware in case of a reboot or a hardware failure.

More details can be found in the official hostPath Kubernetes documentation.

CHAPTER

FOURTEEN

BINDING PERCONA XTRADB CLUSTER COMPONENTS TO SPECIFIC KUBERNETES/OPENSHIFT NODES

The operator does good job automatically assigning new Pods to nodes with sufficient to achieve balanced distribution across the cluster. Still there are situations when it worth to ensure that pods will land on specific nodes: for example, to get speed advantages of the SSD equipped machine, or to reduce costs choosing nodes in a same availability zone.

Both `pxc` and `proxysql` sections of the deploy/cr.yaml file contain keys which can be used to do this, depending on what is the best for a particular situation.

14.1 Node selector

`nodeSelector` contains one or more key-value pairs. If the node is not labeled with each key-value pair from the Pod's `nodeSelector`, the Pod will not be able to land on it.

The following example binds the Pod to any node having a self-explanatory `disktype: ssd` label:

```
nodeSelector:
  disktype: ssd
```

14.2 Affinity and anti-affinity

Affinity makes Pod eligible (or not eligible - so called "anti-affinity") to be scheduled on the node which already has Pods with specific labels. Particularly this approach is good to to reduce costs making sure several Pods with intensive data exchange will occupy the same availability zone or even the same node - or, on the contrary, to make them land on different nodes or even different availability zones for the high availability and balancing purposes.

Percona Distribution for MySQL Operator provides two approaches for doing this:

- simple way to set anti-affinity for Pods, built-in into the Operator,
- more advanced approach based on using standard Kubernetes constraints.

14.2.1 Simple approach - use topologyKey of the Percona Distribution for MySQL Operator

Percona Distribution for MySQL Operator provides a `topologyKey` option, which may have one of the following values:

- `kubernetes.io/hostname` - Pods will avoid residing within the same host,
- `failure-domain.beta.kubernetes.io/zone` - Pods will avoid residing within the same zone,
- `failure-domain.beta.kubernetes.io/region` - Pods will avoid residing within the same region,
- `none` - no constraints are applied.

The following example forces Percona XtraDB Cluster Pods to avoid occupying the same node:

```
affinity:
  topologyKey: "kubernetes.io/hostname"
```

14.2.2 Advanced approach - use standard Kubernetes constraints

Previous way can be used with no special knowledge of the Kubernetes way of assigning Pods to specific nodes. Still in some cases more complex tuning may be needed. In this case `advanced` option placed in the deploy/cr.yaml file turns off the effect of the `topologyKey` and allows to use standard Kubernetes affinity constraints of any complexity:

```
affinity:
   advanced:
     podAffinity:
       requiredDuringSchedulingIgnoredDuringExecution:
       - labelSelector:
           matchExpressions:
           - key: security
             operator: In
             values:
             - S1
         topologyKey: failure-domain.beta.kubernetes.io/zone
     podAntiAffinity:
       preferredDuringSchedulingIgnoredDuringExecution:
       - weight: 100
         podAffinityTerm:
           labelSelector:
             matchExpressions:
             - key: security
               operator: In
               values:
               - S2
           topologyKey: kubernetes.io/hostname
     nodeAffinity:
       requiredDuringSchedulingIgnoredDuringExecution:
         nodeSelectorTerms:
         - matchExpressions:
           - key: kubernetes.io/e2e-az-name
             operator: In
             values:
```

(continues on next page)

(continued from previous page)

```
            - e2e-az1
            - e2e-az2
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
          - key: another-node-label-key
            operator: In
            values:
            - another-node-label-value
```

See explanation of the advanced affinity options in Kubernetes documentation.

14.3 Tolerations

Tolerations allow Pods having them to be able to land onto nodes with matching *taints*. Toleration is expressed as a `key` with and `operator`, which is either `exists` or `equal` (the latter variant also requires a `value` the key is equal to). Moreover, toleration should have a specified `effect`, which may be a self-explanatory `NoSchedule`, less strict `PreferNoSchedule`, or `NoExecute`. The last variant means that if a *taint* with `NoExecute` is assigned to node, then any Pod not tolerating this *taint* will be removed from the node, immediately or after the `tolerationSeconds` interval, like in the following example:

```
tolerations:
- key: "node.alpha.kubernetes.io/unreachable"
  operator: "Exists"
  effect: "NoExecute"
  tolerationSeconds: 6000
```

The Kubernetes Taints and Toleratins contains more examples on this topic.

14.4 Priority Classes

Pods may belong to some *priority classes*. This allows scheduler to distinguish more and less important Pods to resolve the situation when some higher priority Pod cannot be scheduled without evicting a lower priority one. This can be done adding one or more PriorityClasses in your Kubernetes cluster, and specifying the `PriorityClassName` in the deploy/cr.yaml file:

```
priorityClassName: high-priority
```

See the Kubernetes Pods Priority and Preemption documentation to find out how to define and use priority classes in your cluster.

14.5 Pod Disruption Budgets

Creating the *Pod Disruption Budget* is the Kubernetes style to limits the number of Pods of an application that can go down simultaneously due to such *voluntary disruptions* as cluster administrator's actions during the update of deployments or nodes, etc. By such a way Distribution Budgets allow large applications to retain their high availability while maintenance and other administrative activities.

We recommend to apply Pod Disruption Budgets manually to avoid situation when Kubernetes stopped all your database Pods. See the official Kubernetes documentation for details.

CHAPTER

FIFTEEN

CHANGING MYSQL OPTIONS

You may require a configuration change for your application. MySQL allows the option to configure the database with a configuration file. You can pass options from the my.cnf configuration file to be included in the MySQL configuration in one of the following ways:

- edit the `deploy/cr.yaml` file,
- use a ConfigMap,
- use a Secret object.

15.1 Edit the `deploy/cr.yaml` file

You can add options from the my.cnf configuration file by editing the configuration section of the `deploy/cr.yaml`. Here is an example:

```
spec:
  secretsName: my-cluster-secrets
  pxc:
    ...
      configuration: |
        [mysqld]
        wsrep_debug=CLIENT
        [sst]
        wsrep_debug=CLIENT
```

See the Custom Resource options, PXC section for more details

15.2 Use a ConfigMap

You can use a configmap and the cluster restart to reset configuration options. A configmap allows Kubernetes to pass or update configuration data inside a containerized application.

Use the `kubectl` command to create the configmap from external resources, for more information see Configure a Pod to use a ConfigMap.

For example, let's suppose that your application requires more connections. To increase your `max_connections` setting in MySQL, you define a `my.cnf` configuration file with the following setting:

```
[mysqld]
...
max_connections=250
```

You can create a configmap from the my.cnf file with the kubectl create configmap command.

You should use the combination of the cluster name with the -pxc suffix as the naming convention for the configmap. To find the cluster name, you can use the following command:

```
kubectl get pxc
```

The syntax for kubectl create configmap command is:

```
kubectl create configmap <cluster-name>-pxc <resource-type=resource-name>
```

The following example defines cluster1-pxc as the configmap name and the my.cnf file as the data source:

```
kubectl create configmap cluster1-pxc --from-file=my.cnf
```

To view the created configmap, use the following command:

```
kubectl describe configmaps cluster1-pxc
```

15.3 Use a Secret Object

The Operator can also store configuration options in Kubernetes Secrets. This can be useful if you need additional protection for some sensitive data.

You should create a Secret object with a specific name, composed of your cluster name and the pxc suffix.

Note: To find the cluster name, you can use the following command:

```
$ kubectl get pxc
```

Configuration options should be put inside a specific key inside of the data section. The name of this key is my.cnf for Percona XtraDB Cluster Pods.

Actual options should be encoded with Base64.

For example, let's define a my.cnf configuration file and put there a pair of MySQL options we used in the previous example:

```
[mysqld]
wsrep_debug=CLIENT
[sst]
wsrep_debug=CLIENT
```

You can get a Base64 encoded string from your options via the command line as follows:

```
$ cat my.cnf | base64
```

Note: Similarly, you can read the list of options from a Base64 encoded string:

```
$ echo "W215c3FsZF0Kd3NyZXBfZGVidWc9T04KW3NzdF0Kd3NyZXBfZGVidWc9T04K" | base64 --decode
```

Finally, use a yaml file to create the Secret object. For example, you can create a `deploy/my-pxc-secret.yaml` file with the following contents:

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: cluster1-pxc
data:
  my.cnf: "W215c3FsZF0Kd3NyZXBfZGVidWc9T04KW3NzdF0Kd3NyZXBfZGVidWc9T04K"
```

When ready, apply it with the following command:

```
$ kubectl create -f deploy/my-pxc-secret.yaml
```

Note: Do not forget to restart Percona XtraDB Cluster to ensure the cluster has updated the configuration.

15.4 Make changed options visible to the Percona XtraDB Cluster

Do not forget to restart Percona XtraDB Cluster to ensure the cluster has updated the configuration (see details on how to connect in the Install Percona XtraDB Cluster on Kubernetes page).

15.5 Auto-tuning MySQL options

Few configuration options for MySQL can be calculated and set by the Operator automatically based on the available Pod resources (memory and CPU) **if these options are not specified by user** (either in CR.yaml or in ConfigMap).

Options which can be set automatically are the following ones:

- `innodb_buffer_pool_size`
- `max_connections`

If Percona XtraDB Cluster Pod limits are defined, then limits values are used to calculate these options. If Percona XtraDB Cluster Pod limits are not defined, Operator looks for Percona XtraDB Cluster Pod requests as the basis for calculations. if neither Percona XtraDB Cluster Pod limits nor Percona XtraDB Cluster Pod requests are defined, auto-tuning is not done.

CHAPTER

SIXTEEN

DEFINE ENVIRONMENT VARIABLES

Sometimes you need to define new environment variables to provide additional configuration for the components of your cluster. For example, you can use it to customize the configuration of HAProxy, or to add additional options for PMM Client.

The Operator can store environment variables in Kubernetes Secrets. Here is an example with several HAProxy options:

```
apiVersion: v1
kind: Secret
metadata:
  name: my-env-var-secrets
type: Opaque
data:
  HA_CONNECTION_TIMEOUT: MTAwMA==
  OK_IF_DONOR: MQ==
  HA_SERVER_OPTIONS: Y2hlY2sgaW50ZXIgMzAwMDAgcmlzZSAxIGZhbGwgNSB3ZWlnaHQgMQ==
```

As you can see, environment variables are stored as `data` - i.e., base64-encoded strings, so you'll need to encode the value of each variable. For example, To have `HA_CONNECTION_TIMEOUT` variable equal to `100`, you can run `echo -n "100" | base64` in your local shell and get `MTAwMA==`.

Note: Similarly, you can read the list of options from a Base64-encoded string:

```
$ echo "MTAwMAo=" | base64 --decode
```

When ready, apply the YAML file with the following command:

```
$ kubectl create -f deploy/my-env-secret.yaml
```

Put the name of this Secret to the `envVarsSecret` key either in `pxc`, `haproxy` or `proxysql` section of the *deploy/cr.yaml*` configuration file:

```
haproxy:
  ....
  envVarsSecret: my-env-var-secrets
  ....
```

Now apply the `deploy/cr.yaml` file with the following command:

```
kubectl apply -f deploy/cr.yaml
```

CHAPTER

SEVENTEEN

CONFIGURING LOAD BALANCING WITH HAPROXY

Percona Distribution for MySQL Operator provides a choice of two cluster components to provide load balancing and proxy service: you can use either HAProxy or ProxySQL. You can control which one to use, if any, by enabling or disabling via the `haproxy.enabled` and `proxysql.enabled` options in the `deploy/cr.yaml` configuration file.

Use the following command to enable HAProxy:

```
kubectl patch pxc cluster1 --type=merge --patch '{
  "spec": {
     "haproxy": {
        "enabled": true,
        "size": 3,
        "image": "percona/percona-xtradb-cluster-operator:1.9.0-haproxy" },
     "proxysql": { "enabled": false }
  }}'
```

Note: For obvious reasons the Operator will not allow the simultaneous enabling of both HAProxy and ProxySQL.

The resulting HAPproxy setup will contain two services:

- `cluster1-haproxy` service listening on ports 3306 (MySQL) and 3309 (the proxy protocol). This service is pointing to the number zero Percona XtraDB Cluster member (`cluster1-pxc-0`) by default when this member is available. If a zero member is not available, members are selected in descending order of their numbers (e.g. `cluster1-pxc-2`, then `cluster1-pxc-1`, etc.). This service can be used for both read and write load, or it can also be used just for write load (single writer mode) in setups with split write and read loads.
- `cluster1-haproxy-replicas` listening on port 3306 (MySQL). This service selects Percona XtraDB Cluster members to serve queries following the Round Robin load balancing algorithm.

When the cluster with HAProxy is upgraded, the following steps take place. First, reader members are upgraded one by one: the Operator waits until the upgraded Percona XtraDB Cluster member becomes synced, and then proceeds to upgrade the next member. When the upgrade is finished for all the readers, then the writer Percona XtraDB Cluster member is finally upgraded.

17.1 Passing custom configuration options to HAProxy

You can pass custom configuration to HAProxy in one of the following ways: * edit the `deploy/cr.yaml` file, * use a ConfigMap, * use a Secret object.

Note: If you specify a custom HAProxy configuration in this way, the Operator doesn't provide its own HAProxy configuration file. That's why you should specify either a full set of configuration options or nothing.

17.1.1 Edit the `deploy/cr.yaml` file

You can add options from the haproxy.cfg configuration file by editing `haproxy.configuration` key in the `deploy/cr.yaml` file. Here is an example:

```
...
haproxy:
    enabled: true
    size: 3
    image: percona/percona-xtradb-cluster-operator:1.5.0-haproxy
    configuration: |
      global
        maxconn 2048
        external-check
        stats socket /var/run/haproxy.sock mode 600 expose-fd listeners level user
      defaults
        log global
        mode tcp
        retries 10
        timeout client 10000
        timeout connect 100500
        timeout server 10000
      frontend galera-in
        bind *:3309 accept-proxy
        bind *:3306
        mode tcp
        option clitcpka
        default_backend galera-nodes
      frontend galera-replica-in
        bind *:3309 accept-proxy
        bind *:3307
        mode tcp
        option clitcpka
        default_backend galera-replica-nodes
```

17.1.2 Use a ConfigMap

You can use a configmap and the cluster restart to reset configuration options. A configmap allows Kubernetes to pass or update configuration data inside a containerized application.

Use the `kubectl` command to create the configmap from external resources, for more information see Configure a Pod to use a ConfigMap.

For example, you define a `haproxy.cfg` configuration file with the following setting:

```
global
  maxconn 2048
  external-check
  stats socket /var/run/haproxy.sock mode 600 expose-fd listeners level user
defaults
  log global
  mode tcp
  retries 10
  timeout client 10000
  timeout connect 100500
  timeout server 10000
frontend galera-in
  bind *:3309 accept-proxy
  bind *:3306
  mode tcp
  option clitcpka
  default_backend galera-nodes
frontend galera-replica-in
  bind *:3309 accept-proxy
  bind *:3307
  mode tcp
  option clitcpka
  default_backend galera-replica-nodes
```

You can create a configmap from the `haproxy.cfg` file with the `kubectl create configmap` command.

You should use the combination of the cluster name with the `-haproxy` suffix as the naming convention for the configmap. To find the cluster name, you can use the following command:

```
kubectl get pxc
```

The syntax for `kubectl create configmap` command is:

```
kubectl create configmap <cluster-name>-haproxy <resource-type=resource-name>
```

The following example defines `cluster1-haproxy` as the configmap name and the `haproxy.cfg` file as the data source:

```
kubectl create configmap cluster1-haproxy --from-file=haproxy.cfg
```

To view the created configmap, use the following command:

```
kubectl describe configmaps cluster1-haproxy
```

17.1.3 Use a Secret Object

The Operator can also store configuration options in Kubernetes Secrets. This can be useful if you need additional protection for some sensitive data.

You should create a Secret object with a specific name, composed of your cluster name and the `haproxy` suffix.

Note: To find the cluster name, you can use the following command:

```
$ kubectl get pxc
```

Configuration options should be put inside a specific key inside of the `data` section. The name of this key is `haproxy.cfg` for ProxySQL Pods.

Actual options should be encoded with Base64.

For example, let's define a `haproxy.cfg` configuration file and put there options we used in the previous example:

```
global
  maxconn 2048
  external-check
  stats socket /var/run/haproxy.sock mode 600 expose-fd listeners level user
defaults
  log global
  mode tcp
  retries 10
  timeout client 10000
  timeout connect 100500
  timeout server 10000
frontend galera-in
  bind *:3309 accept-proxy
  bind *:3306
  mode tcp
  option clitcpka
  default_backend galera-nodes
frontend galera-replica-in
  bind *:3309 accept-proxy
  bind *:3307
  mode tcp
  option clitcpka
  default_backend galera-replica-nodes
```

You can get a Base64 encoded string from your options via the command line as follows:

```
$ cat haproxy.cfg | base64
```

Note: Similarly, you can read the list of options from a Base64 encoded string:


```
$ echo "IGdsb2JhbAogICBtYXhjb25uIDIwNDgKICAgZXh0ZXJuYWwtY2hlY2sKICAgc3RhdHMgc29ja2V0\
  IC92YXIvcnVuL2hhcHJveHkuc29jayBtb2RlIDYwMCBleHBvc2UtZmQgbGlzdGVuZXJzIGxldmVs\
  IHVzZXIKIGRlZmF1bHRzCiAgIGxvZyBnbG9iYWwKICAgbW9kZSB0Y3AKICAgcmV0cmllcyAxMAog\
  ICB0aW1lb3V0IGNsaWVudCAxMDAwMAogICB0aW1lb3V0IGNvbm5lY3QgMTAwNTAwCiAgIHRpbWVv\
  dXQgc2VydmVyIDEwMDAwCiBmcm9udGVuZCBnYWxlcmEtaW4KICAgYmluZCAqOjMzMDkgYWNjZXB0\
```


(continues on next page)

(continued from previous page)

```
  LXByb3h5CiAgIGJpbmQgKjozMzA2CiAgIG1vZGUgdGNwCiAgIG9wdGlvbiBjbGl0Y3BrYQogICBk\
  ZWZhdWx0X2JhY2tlbmQgZ2FsZXJhLW5vZGVzCiBmcm9udGVuZCBnYWxlcmEtcmVwbGljYS1pbgog\
  ICBiaW5kICo6MzMwOSBhY2NlcHQtcHJveHkKICAgYmluZCAqOjMzMDcKICAgbW9kZSB0Y3AKICAg\
  b3B0aW9uIGNsaXRjcGthCiAgIGRlZmF1bHRfYmFja2VuZCBnYWxlcmEtcmVwbGljYS1ub2Rlcwo=" | base64␣
↪--decode
```

Finally, use a yaml file to create the Secret object. For example, you can create a `deploy/my-haproxy-secret.yaml` file with the following contents:

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: cluster1-haproxy
data:
  my.cnf: "IGdsb2JhbAogICBtYXhjb25uIDIwNDgKICAgZXh0ZXJuYWwtY2hlY2sKICAgc3RhdHMgc29ja2V0\
     IC92YXIvcnVuL2hhcHJveHkuc29jayBtb2RlIDYwMCBleHBvc2UtZmQgbGlzdGVuZXJzIGxldmVs\
     IHVzZXIKIGRlZmF1bHRzCiAgIGxvZyBnbG9iYWwKICAgbW9kZSB0Y3AKICAgcmV0cmllcyAxMAog\
     ICB0aW1lb3V0IGNsaWVudCAxMDAwMAogICB0aW1lb3V0IGNvbm5lY3QgMTAwNTAwCiAgIHRpbWVv\
     dXQgc2VydmVyIDEwMDAwCiBmcm9udGVuZCBnYWxlcmEtaW4KICAgYmluZCAqOjMzMDkgYWNjZXB0\
     LXByb3h5CiAgIGJpbmQgKjozMzA2CiAgIG1vZGUgdGNwCiAgIG9wdGlvbiBjbGl0Y3BrYQogICBk\
     ZWZhdWx0X2JhY2tlbmQgZ2FsZXJhLW5vZGVzCiBmcm9udGVuZCBnYWxlcmEtcmVwbGljYS1pbgog\
     ICBiaW5kICo6MzMwOSBhY2NlcHQtcHJveHkKICAgYmluZCAqOjMzMDcKICAgbW9kZSB0Y3AKICAg\
     b3B0aW9uIGNsaXRjcGthCiAgIGRlZmF1bHRfYmFja2VuZCBnYWxlcmEtcmVwbGljYS1ub2Rlcwo="
```

When ready, apply it with the following command:

```
$ kubectl create -f deploy/my-haproxy-secret.yaml
```

Note: Do not forget to restart Percona XtraDB Cluster to ensure the cluster has updated the configuration.

17.2 Enabling the Proxy protocol

The Proxy protocol allows HAProxy to provide a real client address to Percona XtraDB Cluster.

Note: To use this feature, you should have a Percona XtraDB Cluster image version `8.0.21` or newer.

Normally Proxy protocol is disabled, and Percona XtraDB Cluster sees the IP address of the proxying server (HAProxy) instead of the real client address. But there are scenarios when making real client IP-address visible for Percona XtraDB Cluster is important: e.g. it allows to have privilege grants based on client/application address, and significantly enhance auditing.

You can enable Proxy protocol on Percona XtraDB Cluster by adding proxy_protocol_networks option to *pxc.configuration* key in the `deploy/cr.yaml` configuration file.

Note: Depending on the load balancer of your cloud provider, you may also need setting *haproxy.externaltrafficpolicy* option in `deploy/cr.yaml`.

More information about Proxy protocol can be found in the official HAProxy documentation.

CHAPTER

EIGHTEEN

CONFIGURING LOAD BALANCING WITH PROXYSQL

Percona Distribution for MySQL Operator provides a choice of two cluster components to provide load balancing and proxy service: you can use either HAProxy or ProxySQL. You can control which one to use, if any, by enabling or disabling via the `haproxy.enabled` and `proxysql.enabled` options in the `deploy/cr.yaml` configuration file.

Use the following command to enable ProxySQL:

```
kubectl patch pxc cluster1 --type=merge --patch '{
  "spec": {
     "proxysql": {
        "enabled": true,
        "size": 3,
        "image": "percona/percona-xtradb-cluster-operator:1.9.0-proxysql" },
     "haproxy": { "enabled": false }
  }}'
```

Note: For obvious reasons the Operator will not allow the simultaneous enabling of both HAProxy and ProxySQL.

The resulting setup will use the number zero Percona XtraDB Cluster member (`cluster1-pxc-0` by default) as writer.

When a cluster with ProxySQL is upgraded, the following steps take place. First, reader members are upgraded one by one: the Operator waits until the upgraded member shows up in ProxySQL with online status, and then proceeds to upgrade the next member. When the upgrade is finished for all the readers, then the writer Percona XtraDB Cluster member is finally upgraded.

Note: when both ProxySQL and Percona XtraDB Cluster are upgraded, they are upgraded in parallel.

18.1 Passing custom configuration options to ProxySQL

You can pass custom configuration to ProxySQL

- edit the `deploy/cr.yaml` file,
- use a ConfigMap,
- use a Secret object.

Note: If you specify a custom ProxySQL configuration in this way, ProxySQL will try to merge the passed parameters with the previously set configuration parameters, if any. If ProxySQL fails to merge some option, you will see a warning in its log.

18.1.1 Edit the deploy/cr.yaml file

You can add options from the proxysql.cnf configuration file by editing the proxysql.configuration key in the deploy/cr.yaml file. Here is an example:

```
...
proxysql:
  enabled: false
  size: 3
  image: percona/percona-xtradb-cluster-operator:1.9.0-proxysql
  configuration: |
    datadir="/var/lib/proxysql"

    admin_variables =
    {
      admin_credentials="proxyadmin:admin_password"
      mysql_ifaces="0.0.0.0:6032"
      refresh_interval=2000

      cluster_username="proxyadmin"
      cluster_password="admin_password"
      cluster_check_interval_ms=200
      cluster_check_status_frequency=100
      cluster_mysql_query_rules_save_to_disk=true
      cluster_mysql_servers_save_to_disk=true
      cluster_mysql_users_save_to_disk=true
      cluster_proxysql_servers_save_to_disk=true
      cluster_mysql_query_rules_diffs_before_sync=1
      cluster_mysql_servers_diffs_before_sync=1
      cluster_mysql_users_diffs_before_sync=1
      cluster_proxysql_servers_diffs_before_sync=1
    }

    mysql_variables=
    {
      monitor_password="monitor"
      monitor_galera_healthcheck_interval=1000
      threads=2
      max_connections=2048
      default_query_delay=0
      default_query_timeout=10000
      poll_timeout=2000
      interfaces="0.0.0.0:3306"
      default_schema="information_schema"
      stacksize=1048576
      connect_timeout_server=10000
      monitor_history=60000
      monitor_connect_interval=20000
      monitor_ping_interval=10000
      ping_timeout_server=200
```

(continues on next page)

(continued from previous page)

```
      commands_stats=true
      sessions_sort=true
      have_ssl=true
      ssl_p2s_ca="/etc/proxysql/ssl-internal/ca.crt"
      ssl_p2s_cert="/etc/proxysql/ssl-internal/tls.crt"
      ssl_p2s_key="/etc/proxysql/ssl-internal/tls.key"
      ssl_p2s_cipher="ECDHE-RSA-AES128-GCM-SHA256"
    }
```

18.1.2 Use a ConfigMap

You can use a configmap and the cluster restart to reset configuration options. A configmap allows Kubernetes to pass or update configuration data inside a containerized application.

Use the `kubectl` command to create the configmap from external resources, for more information see Configure a Pod to use a ConfigMap.

For example, you define a `proxysql.cnf` configuration file with the following setting:

```
datadir="/var/lib/proxysql"

admin_variables =
{
  admin_credentials="proxyadmin:admin_password"
  mysql_ifaces="0.0.0.0:6032"
  refresh_interval=2000

  cluster_username="proxyadmin"
  cluster_password="admin_password"
  cluster_check_interval_ms=200
  cluster_check_status_frequency=100
  cluster_mysql_query_rules_save_to_disk=true
  cluster_mysql_servers_save_to_disk=true
  cluster_mysql_users_save_to_disk=true
  cluster_proxysql_servers_save_to_disk=true
  cluster_mysql_query_rules_diffs_before_sync=1
  cluster_mysql_servers_diffs_before_sync=1
  cluster_mysql_users_diffs_before_sync=1
  cluster_proxysql_servers_diffs_before_sync=1
}

mysql_variables=
{
  monitor_password="monitor"
  monitor_galera_healthcheck_interval=1000
  threads=2
  max_connections=2048
  default_query_delay=0
  default_query_timeout=10000
  poll_timeout=2000
  interfaces="0.0.0.0:3306"
  default_schema="information_schema"
```

(continues on next page)

(continued from previous page)

```
  stacksize=1048576
  connect_timeout_server=10000
  monitor_history=60000
  monitor_connect_interval=20000
  monitor_ping_interval=10000
  ping_timeout_server=200
  commands_stats=true
  sessions_sort=true
  have_ssl=true
  ssl_p2s_ca="/etc/proxysql/ssl-internal/ca.crt"
  ssl_p2s_cert="/etc/proxysql/ssl-internal/tls.crt"
  ssl_p2s_key="/etc/proxysql/ssl-internal/tls.key"
  ssl_p2s_cipher="ECDHE-RSA-AES128-GCM-SHA256"
}
```

You can create a configmap from the `proxysql.cnf` file with the `kubectl create configmap` command.

You should use the combination of the cluster name with the `-proxysql` suffix as the naming convention for the configmap. To find the cluster name, you can use the following command:

```
kubectl get pxc
```

The syntax for `kubectl create configmap` command is:

```
kubectl create configmap <cluster-name>-proxysql <resource-type=resource-name>
```

The following example defines `cluster1-proxysql` as the configmap name and the `proxysql.cnf` file as the data source:

```
kubectl create configmap cluster1-proxysql --from-file=proxysql.cnf
```

To view the created configmap, use the following command:

```
kubectl describe configmaps cluster1-proxysql
```

18.1.3 Use a Secret Object

The Operator can also store configuration options in Kubernetes Secrets. This can be useful if you need additional protection for some sensitive data.

You should create a Secret object with a specific name, composed of your cluster name and the `proxysql` suffix.

Note: To find the cluster name, you can use the following command:

```
$ kubectl get pxc
```

Configuration options should be put inside a specific key inside of the `data` section. The name of this key is `proxysql.cnf` for ProxySQL Pods.

Actual options should be encoded with Base64.

For example, let's define a `proxysql.cnf` configuration file and put there options we used in the previous example:

```
datadir="/var/lib/proxysql"

admin_variables =
{
  admin_credentials="proxyadmin:admin_password"
  mysql_ifaces="0.0.0.0:6032"
  refresh_interval=2000

  cluster_username="proxyadmin"
  cluster_password="admin_password"
  cluster_check_interval_ms=200
  cluster_check_status_frequency=100
  cluster_mysql_query_rules_save_to_disk=true
  cluster_mysql_servers_save_to_disk=true
  cluster_mysql_users_save_to_disk=true
  cluster_proxysql_servers_save_to_disk=true
  cluster_mysql_query_rules_diffs_before_sync=1
  cluster_mysql_servers_diffs_before_sync=1
  cluster_mysql_users_diffs_before_sync=1
  cluster_proxysql_servers_diffs_before_sync=1
}

mysql_variables=
{
  monitor_password="monitor"
  monitor_galera_healthcheck_interval=1000
  threads=2
  max_connections=2048
  default_query_delay=0
  default_query_timeout=10000
  poll_timeout=2000
  interfaces="0.0.0.0:3306"
  default_schema="information_schema"
  stacksize=1048576
  connect_timeout_server=10000
  monitor_history=60000
  monitor_connect_interval=20000
  monitor_ping_interval=10000
  ping_timeout_server=200
  commands_stats=true
  sessions_sort=true
  have_ssl=true
  ssl_p2s_ca="/etc/proxysql/ssl-internal/ca.crt"
  ssl_p2s_cert="/etc/proxysql/ssl-internal/tls.crt"
  ssl_p2s_key="/etc/proxysql/ssl-internal/tls.key"
  ssl_p2s_cipher="ECDHE-RSA-AES128-GCM-SHA256"
}
```

You can get a Base64 encoded string from your options via the command line as follows:

```
$ cat proxysql.cnf | base64
```

Note: Similarly, you can read the list of options from a Base64 encoded string:

```
$ echo "ZGF0YWRpcj0iL3Zhci9saWIvcHJveHlzcWwiCgphZG1pbl92YXJpYWJsZXMgPQp7CiBhZG1pbl9j\
  cmVkZW50aWFscz0icHJveHlhZG1pbjphZG1pbl9wYXNzd29yZCIKIG15c3FsX2lmYWNlcz0iMC4w\
  LjAuMDo2MDMyIgogcmVmcmVzaF9pbnRlcnZhbD0yMDAwCgogY2x1c3Rlcl91c2VybmFtZT0icHJv\
  eHlhZG1pbiIKIGNsdXN0ZXJfcGFzc3dvcmQ9ImFkbWluX3Bhc3N3b3JkIgogY2x1c3Rlcl9jaGVj\
  a19pbnRlcnZhbF9tcz0yMDAKIGNsdXN0ZXJfY2hlY2tfc3RhdHVzX2ZyZXF1ZW5jeT0xMDAKIGNs\
  dXN0ZXJfbXlzcWxfcXVlcnlfcnVsZXNfc2F2ZV90b19kaXNrPXRydWUKIGNsdXN0ZXJfbXlzcWxf\
  c2VydmVyc19zYXZlX3RvX2Rpc2s9dHJ1ZQogY2x1c3Rlcl9teXNxbF91c2Vyc19zYXZlX3RvX2Rp\
  c2s9dHJ1ZQogY2x1c3Rlcl9wcm94eXNxbF9zZXJ2ZXJzX3NhdmVfdG9fZGlzaz10cnVlCiBjbHVz\
  dGVyX215c3FsX3F1ZXJ5X3J1bGVzX2RpZmZzX2JlZm9yZV9zeW5jPTEKIGNsdXN0ZXJfbXlzcWxf\
  c2VydmVyc19kaWZmc19iZWZvcmVfc3luYz0xCiBjbHVzdGVyX215c3FsX3VzZXJzX2RpZmZzX2Jl\
  Zm9yZV9zeW5jPTEKIGNsdXN0ZXJfcHJveHlzcWxfc2VydmVyc19kaWZmc19iZWZvcmVfc3luYz0x\
  Cn0KCm15c3FsX3ZhcmlhYmxlcz0KewogbW9uaXRvcl9wYXNzd29yZD0ibW9uaXRvciIKIG1vbml0\
  b3JfZ2FsZXJhX2hlYWx0aGNoZWNrX2ludGVydmFsPTEwMDAKIHRocmVhZHM9MgogbWF4X2Nvbm5l\
  Y3Rpb25zPTIwNDgKIGRlZmF1bHRfcXVlcnlfZGVsYXk9MAogZGVmYXVsdF9xdWVyeV90aW1lb3V0\
  PTEwMDAwCiBwb2xsX3RpbWVvdXQ9MjAwMAogaW50ZXJmYWNlcz0iMC4wLjAuMDozMzA2IgogZGVm\
  YXVsdF9zY2hlbWE9ImluZm9ybWF0aW9uX3NjaGVtYSIKIHN0YWNrc2l6ZT0xMDQ4NTc2CiBjb25u\
  ZWN0X3RpbWVvdXRfc2VydmVyPTEwMDAwCiBtb25pdG9yX2hpc3Rvcnk9NjAwMDAKIG1vbml0b3Jf\
  Y29ubmVjdF9pbnRlcnZhbD0yMDAwMAogbW9uaXRvcl9waW5nX2ludGVydmFsPTEwMDAwCiBwaW5n\
  X3RpbWVvdXRfc2VydmVyPTIwMAogY29tbWFuZHNfc3RhdHM9dHJ1ZQogc2Vzc2lvbnNfc29ydD10\
  cnVlCiBoYXZlX3NzbD10cnVlCiBzc2xfcDJzX2NhPSIvZXRjL3Byb3h5c3FsL3NzbC1pbnRlcm5h\
  bC9jYS5jcnQiCiBzc2xfcDJzX2NlcnQ9Ii9ldGMvcHJveHlzcWwvc3NsLWludGVybmFsL3Rscy5j\
  cnQiCiBzc2xfcDJzX2tleT0iL2V0Yy9wcm94eXNxbC9zc2wtaW50ZXJuYWwvdGxzLmtleSIKIHNz\
  bF9wMnNfY2lwaGVyPSJFQ0RIRS1SU0EtQUVTMTI4LUdDTS1TSEEyNTYiCn0K" | base64 --decode
```

Finally, use a yaml file to create the Secret object. For example, you can create a `deploy/my-proxysql-secret.yaml` file with the following contents:

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: cluster1-proxysql
data:
  my.cnf: "ZGF0YWRpcj0iL3Zhci9saWIvcHJveHlzcWwiCgphZG1pbl92YXJpYWJsZXMgPQp7CiBhZG1pbl9j\
     cmVkZW50aWFscz0icHJveHlhZG1pbjphZG1pbl9wYXNzd29yZCIKIG15c3FsX2lmYWNlcz0iMC4w\
     LjAuMDo2MDMyIgogcmVmcmVzaF9pbnRlcnZhbD0yMDAwCgogY2x1c3Rlcl91c2VybmFtZT0icHJv\
     eHlhZG1pbiIKIGNsdXN0ZXJfcGFzc3dvcmQ9ImFkbWluX3Bhc3N3b3JkIgogY2x1c3Rlcl9jaGVj\
     a19pbnRlcnZhbF9tcz0yMDAKIGNsdXN0ZXJfY2hlY2tfc3RhdHVzX2ZyZXF1ZW5jeT0xMDAKIGNs\
     dXN0ZXJfbXlzcWxfcXVlcnlfcnVsZXNfc2F2ZV90b19kaXNrPXRydWUKIGNsdXN0ZXJfbXlzcWxf\
     c2VydmVyc19zYXZlX3RvX2Rpc2s9dHJ1ZQogY2x1c3Rlcl9teXNxbF91c2Vyc19zYXZlX3RvX2Rp\
     c2s9dHJ1ZQogY2x1c3Rlcl9wcm94eXNxbF9zZXJ2ZXJzX3NhdmVfdG9fZGlzaz10cnVlCiBjbHVz\
     dGVyX215c3FsX3F1ZXJ5X3J1bGVzX2RpZmZzX2JlZm9yZV9zeW5jPTEKIGNsdXN0ZXJfbXlzcWxf\
     c2VydmVyc19kaWZmc19iZWZvcmVfc3luYz0xCiBjbHVzdGVyX215c3FsX3VzZXJzX2RpZmZzX2Jl\
     Zm9yZV9zeW5jPTEKIGNsdXN0ZXJfcHJveHlzcWxfc2VydmVyc19kaWZmc19iZWZvcmVfc3luYz0x\
     Cn0KCm15c3FsX3ZhcmlhYmxlcz0KewogbW9uaXRvcl9wYXNzd29yZD0ibW9uaXRvciIKIG1vbml0\
     b3JfZ2FsZXJhX2hlYWx0aGNoZWNrX2ludGVydmFsPTEwMDAKIHRocmVhZHM9MgogbWF4X2Nvbm5l\
     Y3Rpb25zPTIwNDgKIGRlZmF1bHRfcXVlcnlfZGVsYXk9MAogZGVmYXVsdF9xdWVyeV90aW1lb3V0\
     PTEwMDAwCiBwb2xsX3RpbWVvdXQ9MjAwMAogaW50ZXJmYWNlcz0iMC4wLjAuMDozMzA2IgogZGVm\
     YXVsdF9zY2hlbWE9ImluZm9ybWF0aW9uX3NjaGVtYSIKIHN0YWNrc2l6ZT0xMDQ4NTc2CiBjb25u\
     ZWN0X3RpbWVvdXRfc2VydmVyPTEwMDAwCiBtb25pdG9yX2hpc3Rvcnk9NjAwMDAKIG1vbml0b3Jf\
     Y29ubmVjdF9pbnRlcnZhbD0yMDAwMAogbW9uaXRvcl9waW5nX2ludGVydmFsPTEwMDAwCiBwaW5n\
     X3RpbWVvdXRfc2VydmVyPTIwMAogY29tbWFuZHNfc3RhdHM9dHJ1ZQogc2Vzc2lvbnNfc29ydD10\
     cnVlCiBoYXZlX3NzbD10cnVlCiBzc2xfcDJzX2NhPSIvZXRjL3Byb3h5c3FsL3NzbC1pbnRlcm5h\
```

(continues on next page)

(continued from previous page)

```
    bC9jYS5jcnQiCiBzc2xfcDJzX2NlcnQ9Ii9ldGMvcHJveHlzcWwvc3NsLWludGVybmFsL3Rscy5j\
    cnQiCiBzc2xfcDJzX2tleT0iL2V0Yy9wcm94eXNxbC9zc2wtaW50ZXJuYWwvdGxzLmtleSIKIHNz\
    bF9wMnNfY2lwaGVyPSJFQ0RIRS1SU0EtQUVTMTI4LUdDTS1TSEEyNTYiCn0K"
```

When ready, apply it with the following command:

```
$ kubectl create -f deploy/my-proxysql-secret.yaml
```

Note: Do not forget to restart Percona XtraDB Cluster to ensure the cluster has updated the configuration.

18.2 Accessing the ProxySQL Admin Interface

You can use ProxySQL admin interface to configure its settings.

Configuring ProxySQL in this way means connecting to it using the MySQL protocol, and two things are needed to do it:

- the ProxySQL Pod name
- the ProxySQL admin password

You can find out ProxySQL Pod name with the `kubectl get pods` command, which will have the following output:

```
$ kubectl get pods
NAME                                               READY   STATUS    RESTARTS   AGE
cluster1-pxc-node-0                                1/1     Running   0          5m
cluster1-pxc-node-1                                1/1     Running   0          4m
cluster1-pxc-node-2                                1/1     Running   0          2m
cluster1-proxysql-0                                1/1     Running   0          5m
percona-xtradb-cluster-operator-dc67778fd-qtspz    1/1     Running   0          6m
```

The next command will print you the needed admin password:

```
kubectl get secrets $(kubectl get pxc -o jsonpath='{.items[].spec.secretsName}') -o␣
↪template='{{ .data.proxyadmin | base64decode }}'
```

When both Pod name and admin password are known, connect to the ProxySQL as follows, substituting `cluster1-proxysql-0` with the actual Pod name and `admin_password` with the actual password:

```
kubectl exec -it cluster1-proxysql-0 -- mysql -h127.0.0.1 -P6032 -uproxyadmin -padmin_
↪password
```

.

CHAPTER

NINETEEN

TRANSPORT LAYER SECURITY (TLS)

The Percona Distribution for MySQL Operator uses Transport Layer Security (TLS) cryptographic protocol for the following types of communication:

- Internal - communication between Percona XtraDB Cluster instances,
- External - communication between the client application and ProxySQL.

The internal certificate is also used as an authorization method.

TLS security can be configured in several ways. By default, the Operator generates long-term certificates automatically if there are no certificate secrets available. Other options are the following ones:

- The Operator can use a specifically installed *cert-manager*, which will automatically generate and renew short-term TLS certificates,
- Certificates can be generated manually.

You can also use pre-generated certificates available in the `deploy/ssl-secrets.yaml` file for test purposes, but we strongly recommend avoiding their usage on any production system!

The following subsections explain how to configure TLS security with the Operator yourself, as well as how to temporarily disable it if needed.

19.1 Install and use the *cert-manager*

19.1.1 About the *cert-manager*

A cert-manager is a Kubernetes certificate management controller which is widely used to automate the management and issuance of TLS certificates. It is community-driven, and open source.

When you have already installed *cert-manager* and deploy the operator, the operator requests a certificate from the *cert-manager*. The *cert-manager* acts as a self-signed issuer and generates certificates. The Percona Operator self-signed issuer is local to the operator namespace. This self-signed issuer is created because Percona XtraDB Cluster requires all certificates issued by the same CA (Certificate authority).

Self-signed issuer allows you to deploy and use the Percona Operator without creating a clusterissuer separately.

19.1.2 Installation of the *cert-manager*

The steps to install the *cert-manager* are the following:

- Create a namespace,
- Disable resource validations on the cert-manager namespace,
- Install the cert-manager.

The following commands perform all the needed actions:

```
$ kubectl create namespace cert-manager
$ kubectl label namespace cert-manager certmanager.k8s.io/disable-validation=true
$ kubectl_bin apply -f https://github.com/jetstack/cert-manager/releases/download/v0.15.
↪1/cert-manager.yaml
```

After the installation, you can verify the *cert-manager* by running the following command:

```
$ kubectl get pods -n cert-manager
```

The result should display the *cert-manager* and webhook active and running.

19.2 Generate certificates manually

To generate certificates manually, follow these steps:

1. Provision a Certificate Authority (CA) to generate TLS certificates
2. Generate a CA key and certificate file with the server details
3. Create the server TLS certificates using the CA keys, certs, and server details

The set of commands generate certificates with the following attributes:

- `Server-pem` - Certificate
- `Server-key.pem` - the private key
- `ca.pem` - Certificate Authority

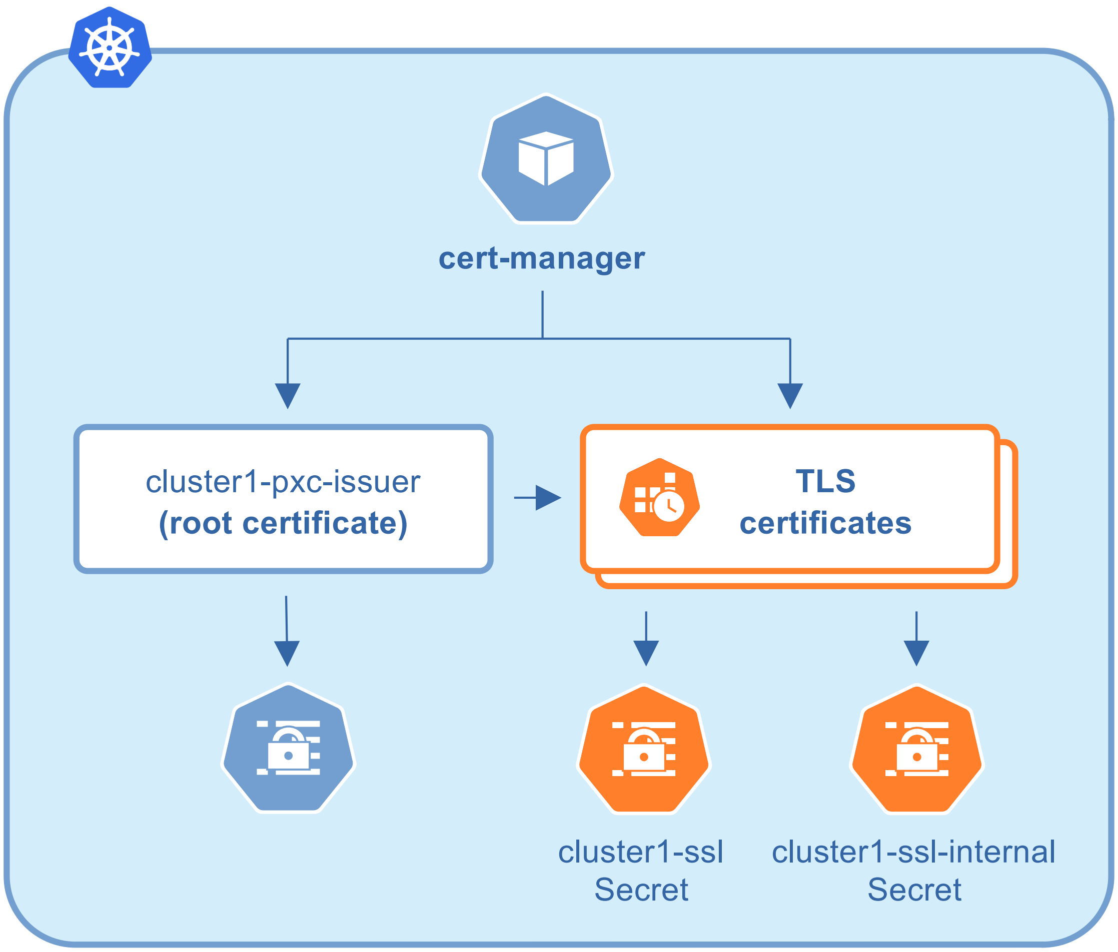
You should generate certificates twice: one set is for external communications, and another set is for internal ones. A secret created for the external use must be added to `cr.yaml/spec/secretsName`. A certificate generated for internal communications must be added to the `cr.yaml/spec/sslInternalSecretName`.

```
$ cat <<EOF | cfssl gencert -initca - | cfssljson -bare ca
{
  "CN": "Root CA",
  "key": {
    "algo": "rsa",
    "size": 2048
  }
}
EOF

$ cat <<EOF | cfssl gencert -ca=ca.pem  -ca-key=ca-key.pem - | cfssljson -bare server
{
  "hosts": [
    "${CLUSTER_NAME}-proxysql",
    "*.${CLUSTER_NAME}-proxysql-unready",
    "*.${CLUSTER_NAME}-pxc"
  ],
  "CN": "${CLUSTER_NAME}-pxc",
  "key": {
    "algo": "rsa",
    "size": 2048
  }
}
EOF

$ kubectl create secret generic my-cluster-ssl --from-file=tls.crt=server.pem --
from-file=tls.key=server-key.pem --from-file=ca.crt=ca.pem --
type=kubernetes.io/tls
```

19.3 Update certificates

If a :ref`cert-manager<tls.certmanager>` is used, it should take care of updating the certificates. If you *generate certificates manually*, you are should take care of updating them in proper time.

TLS certificates issued by cert-manager are short-term ones. Starting from the Operator version 1.9.0 cert-manager issues TLS certificates for 3 months, while root certificate is valid for 3 years. This allows to reissue TLS certificates automatically on schedule and without downtime.



Versions of the Operator prior 1.9.0 have used 3 month root certificate, which caused issues with the automatic TLS certificates update. If that's your case, you can make the Operator update along with the *official instruction*.

Note: If you use the cert-manager version earlier than 1.9.0, and you would like to avoid downtime while updating the certificates after the Operator update to 1.9.0 or newer version, *force the certificates regeneration by a cert-manager*.

19.3.1 Check your certificates for expiration

1. First, check the necessary secrets names (`my-cluster-ssl` and `my-cluster-ssl-internal` by default):

   ```
   $ kubectl get certificate
   ```

   You will have the following response:

   ```
   NAME                    READY   SECRET                    AGE
   cluster1-ssl            True    my-cluster-ssl            49m
   cluster1-ssl-internal   True    my-cluster-ssl-internal   49m
   ```

2. Optionally you can also check that the certificates issuer is up and running:

   ```
   $ kubectl get issuer
   ```

   The response should be as follows:

```
NAME              READY   AGE
cluster1-pxc-ca   True    49m
```

3. Now use the following command to find out the certificates validity dates, substituting Secrets names if necessary:

```
$ {
  kubectl get secret/my-cluster-ssl-internal -o jsonpath='{.data.tls\.crt}' |␣
↪base64 --decode | openssl x509 -inform pem -noout -text | grep "Not After"
  kubectl get secret/my-cluster-ssl -o jsonpath='{.data.ca\.crt}' | base64 --decode␣
↪| openssl x509 -inform pem -noout -text | grep "Not After"
  }
```

The resulting output will be self-explanatory:

```
Not After : Sep 15 11:04:53 2021 GMT
Not After : Sep 15 11:04:53 2021 GMT
```

19.3.2 Update certificates without downtime

If you don't use *cert-manager* and have *created certificates manually*, you can follow the next steps to perform a no-downtime update of these certificates *if they are still valid*.

Note: For already expired certificates, follow *the alternative way*.

Having non-expired certificates, you can roll out new certificates (both CA and TLS) with the Operator as follows.

1. Generate a new CA certificate (`ca.pem`). Optionally you can also generate a new TLS certificate and a key for it, but those can be generated later on step 6.
2. Get the current CA (`ca.pem.old`) and TLS (`tls.pem.old`) certificates and the TLS certificate key (`tls.key.old`):

```
$ kubectl get secret/my-cluster-ssl-internal -o jsonpath='{.data.ca\.crt}' | base64␣
↪--decode > ca.pem.old
$ kubectl get secret/my-cluster-ssl-internal -o jsonpath='{.data.tls\.crt}' |␣
↪base64 --decode > tls.pem.old
$ kubectl get secret/my-cluster-ssl-internal -o jsonpath='{.data.tls\.key}' |␣
↪base64 --decode > tls.key.old
```

3. Combine new and current `ca.pem` into a `ca.pem.combined` file:

```
$ cat ca.pem ca.pem.old >> ca.pem.combined
```

4. Create a new Secrets object with *old* TLS certificate (`tls.pem.old`) and key (`tls.key.old`), but a *new combined* `ca.pem` (`ca.pem.combined`):

```
$ kubectl delete secret/my-cluster-ssl-internal
$ kubectl create secret generic my-cluster-ssl-internal --from-file=tls.crt=tls.pem.
↪old --from-file=tls.key=tls.key.old --from-file=ca.crt=ca.pem.combined --
↪type=kubernetes.io/tls
```

5. The cluster will go through a rolling reconciliation, but it will do it without problems, as every node has old TLS certificate/key, and both new and old CA certificates.

6. If new TLS certificate and key weren't generated on step 1, *do that* now.
7. Create a new Secrets object for the second time: use new TLS certificate (`server.pem` in the example) and its key (`server-key.pem`), and again the combined CA certificate (`ca.pem.combined`):

```
$ kubectl delete secret/my-cluster-ssl-internal
$ kubectl create secret generic my-cluster-ssl-internal --from-file=tls.crt=server.
↪pem --from-file=tls.key=server-key.pem --from-file=ca.crt=ca.pem.combined --
↪type=kubernetes.io/tls
```

8. The cluster will go through a rolling reconciliation, but it will do it without problems, as every node already has a new CA certificate (as a part of the combined CA certificate), and can successfully allow joiners with new TLS certificate to join. Joiner node also has a combined CA certificate, so it can authenticate against older TLS certificate.
9. Create a final Secrets object: use new TLS certificate (`server.pmm`) and its key (`server-key.pem`), and just the new CA certificate (`ca.pem`):

```
$ kubectl delete secret/my-cluster-ssl-internal
$ kubectl create secret generic my-cluster-ssl-internal --from-file=tls.crt=server.
↪pem --from-file=tls.key=server-key.pem --from-file=ca.crt=ca.pem --
↪type=kubernetes.io/tls
```

10. The cluster will go through a rolling reconciliation, but it will do it without problems: the old CA certificate is removed, and every node is already using new TLS certificate and no nodes rely on the old CA certificate any more.

19.3.3 Update certificates with downtime

If your certificates have been already expired (or if you continue to use the Operator version prior to 1.9.0), you should move through the *pause - update Secrets - unpause* route as follows.

1. Pause the cluster *in a standard way*, and make sure it has reached its paused state.
2. If *cert-manager* is used, delete issuer and TLS certificates:

```
$ {
  kubectl delete issuer/cluster1-pxc-ca
  kubectl delete certificate/cluster1-ssl certificate/cluster1-ssl-internal
  }
```

3. Delete Secrets to force the SSL reconciliation:

```
$ kubectl delete secret/my-cluster-ssl secret/my-cluster-ssl-internal
```

4. *Check certificates* to make sure reconciliation have succeeded.
5. Unpause the cluster *in a standard way*, and make sure it has reached its running state.

19.4 Run Percona XtraDB Cluster without TLS

Omitting TLS is also possible, but we recommend that you run your cluster with the TLS protocol enabled.

To disable TLS protocol (e.g. for demonstration purposes) edit the `cr.yaml/spec/allowUnsafeConfigurations` setting to `true` and make sure that there are no certificate secrets available.

CHAPTER

TWENTY

DATA AT REST ENCRYPTION

Full data at rest encryption in Percona XtraDB Cluster is supported by the Operator since version 1.4.0.

Note: Data at rest means inactive data stored as files, database records, etc.

To implement these features, the Operator uses `keyring_vault` plugin, which ships with Percona XtraDB Cluster, and utilizes HashiCorp Vault storage for encryption keys.

20.1 Installing Vault

The following steps will deploy Vault on Kubernetes with the Helm 3 package manager. Other Vault installation methods should also work, so the instruction placed here is not obligatory and is for illustration purposes.

1. Clone the official HashiCorp Vault Helm chart from GitHub:

```
$ git clone -b v0.4.0 https://github.com/hashicorp/vault-helm.git
$ cd vault-helm
```

2. Now use Helm to do the installation:

```
$ helm install vault-service ./
```

3. After the installation, Vault should be first initialized and then unsealed. Initializing Vault is done with the following commands:

```
$ kubectl exec -it pod/vault-service-0 -- vault operator init -key-shares=1 -key-
↪threshold=1 -format=json > /tmp/vault-init
$ unsealKey=$(jq -r ".unseal_keys_b64[]" < /tmp/vault-init)
```

To unseal Vault, execute the following command **for each Pod** of Vault running:

```
$ kubectl exec -it pod/vault-service-0 -- vault operator unseal "$unsealKey"
```

20.2 Configuring Vault

1. First, you should enable secrets within Vault. For this you will need a Vault token. Percona XtraDB Cluster can use any regular token which allows all operations inside the secrets mount point. In the following example we are using the *root token* to be sure the permissions requirement is met, but actually there is no need in root permissions. We don't recommend using the root token on the production system.

   ```
   $ cat /tmp/vault-init | jq -r ".root_token"
   ```

   The output will be like follows:

   ```
   s.VgQvaXl8xGFO1RUxAPbPbsfN
   ```

   Now login to Vault with this token and enable the "pxc-secret" secrets path:

   ```
   $ kubectl exec -it vault-service-0 -- /bin/sh
   $ vault login s.VgQvaXl8xGFO1RUxAPbPbsfN
   $ vault secrets enable --version=1 -path=pxc-secret kv
   ```

   Note: You can also enable audit, which is not mandatory, but useful:

   ```
   $ vault audit enable file file_path=/vault/vault-audit.log
   ```

2. To enable Vault secret within Kubernetes, create and apply the YAML file, as described further.

   2.1. To access the Vault server via HTTP, follow the next YAML file example:

   ```
   apiVersion: v1
   kind: Secret
   metadata:
     name: some-name-vault
   type: Opaque
   stringData:
     keyring_vault.conf: |-
        token = s.VgQvaXl8xGFO1RUxAPbPbsfN
        vault_url = vault-service.vault-service.svc.cluster.local
        secret_mount_point = pxc-secret
   ```

   Note: the `name` key in the above file should be equal to the `spec.vaultSecretName` key from the `deploy/cr.yaml` configuration file.

   2.2. To turn on TLS and access the Vault server via HTTPS, you should do two more things:

   - add one more item to the secret: the contents of the `ca.cert` file with your certificate,
   - store the path to this file in the `vault_ca` key.

   ```
   apiVersion: v1
   kind: Secret
   metadata:
     name: some-name-vault
   type: Opaque
   ```

   (continues on next page)

(continued from previous page)

```
stringData:
  keyring_vault.conf: |-
    token = = s.VgQvaXl8xGFO1RUxAPbPbsfN
    vault_url = https://vault-service.vault-service.svc.cluster.local
    secret_mount_point = pxc-secret
    vault_ca = /etc/mysql/vault-keyring-secret/ca.cert
  ca.cert: |-
    -----BEGIN CERTIFICATE-----
    MIIEczCCA1ugAwIBAgIBADANBgkqhkiG9w0BAQQFAD..AkGA1UEBhMCR0Ix
    EzARBgNVBAgTClNvbWUtU3RhdGUxFDASBgNVBAoTC0..0EgTHRkMTcwNQYD
    7vQMfXdGsRrXNGRGnX+vWDZ3/zWI0joDtCkNnqEpVn..HoX
    -----END CERTIFICATE-----
```

Note: the `name` key in the above file should be equal to the `spec.vaultSecretName` key from the `deploy/cr.yaml` configuration file.

Note: For techincal reasons the `vault_ca` key should either exist or not exist in the YAML file; commented option like `#vault_ca = ...` is not acceptable.

More details on how to install and configure Vault can be found in the official documentation.

20.3 Using the encryption

If using *Percona XtraDB Cluster* 5.7, you should turn encryption on explicitly when you create a table or a tablespace. This can be done by adding the `ENCRYPTION='Y'` part to your SQL statement, like in the following example:

```
CREATE TABLE t1 (c1 INT, PRIMARY KEY pk(c1)) ENCRYPTION='Y';
CREATE TABLESPACE foo ADD DATAFILE 'foo.ibd' ENCRYPTION='Y';
```

Note: See more details on encryption in Percona XtraDB Cluster 5.7 here.

If using *Percona XtraDB Cluster* 8.0, the encryption is turned on by default (in case if Vault is configured).

The following table presents the default values of the correspondent my.cnf configuration options:

Option	Default value
early-plugin-load	keyring_vault.so
keyring_vault_config	/etc/mysql/vault-keyring-secret/keyring_vault.conf
default_table_encryption	ON
table_encryption_privilege_check	ON
innodb_undo_log_encrypt	ON
innodb_redo_log_encrypt	ON
binlog_encryption	ON
binlog_rotate_encryption_master_key_at_startup	ON
innodb_temp_tablespace_encrypt	ON
innodb_parallel_dblwr_encrypt	ON
innodb_encrypt_online_alter_logs	ON
encrypt_tmp_files	ON

Part V

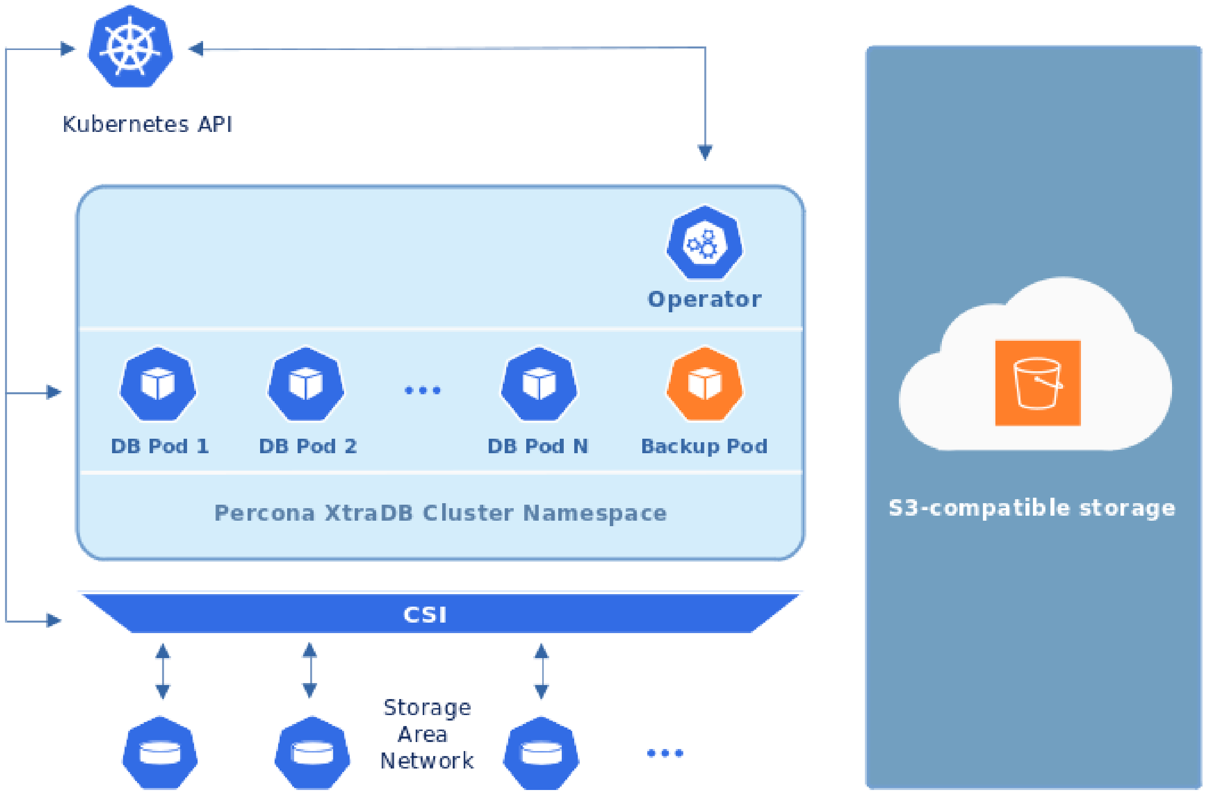
Management

CHAPTER

TWENTYONE

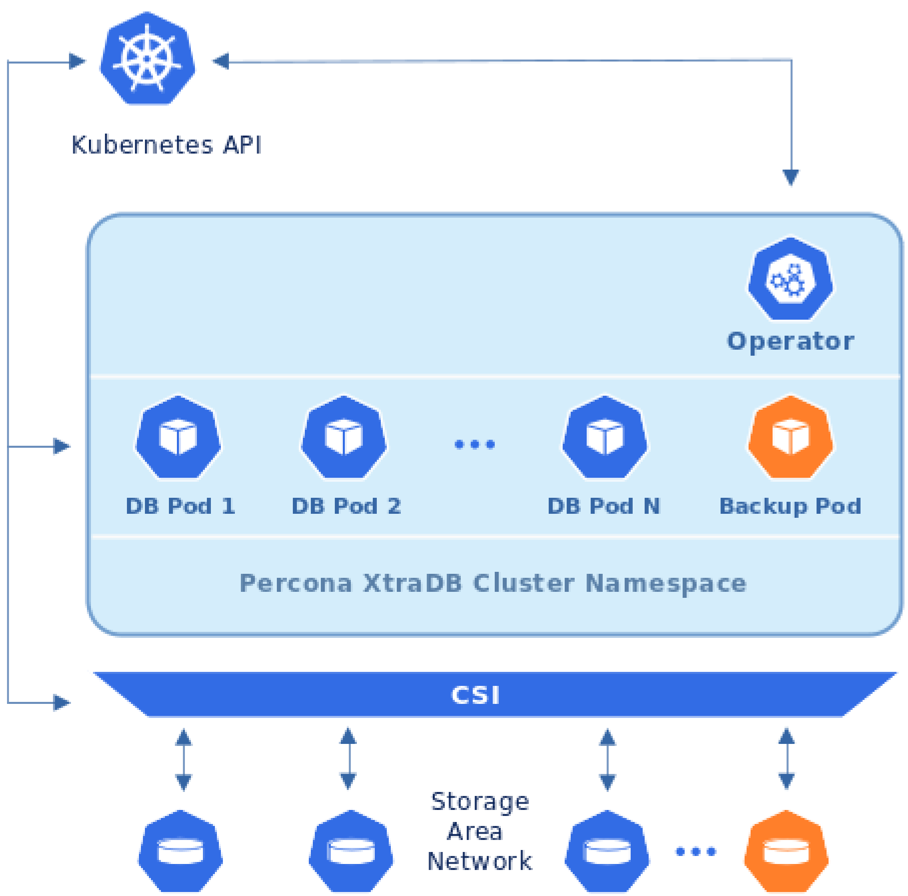
PROVIDING BACKUPS

The Operator usually stores Percona XtraDB Cluster backups on Amazon S3 or S3-compatible storage outside the Kubernetes cluster:



But storing backups on Persistent Volumes inside the Kubernetes cluster is also possible:

The Operator allows doing backups in two ways. *Scheduled backups* are configured in the deploy/cr.yaml file to be executed automatically in proper time. *On-demand backups* can be done manually at any moment.


Kubernetes API
Operator
DB Pod 1
DB Pod 2
DB Pod N
Backup Pod
Percona XtraDB Cluster Namespace
CSI
Storage
Area
Network

21.1 Making scheduled backups

Since backups are stored separately on the Amazon S3, a secret with `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY` should be present on the Kubernetes cluster. The secrets file with these base64-encoded keys should be created: for example `deploy/backup-s3.yaml` file with the following contents:

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: my-cluster-name-backup-s3
type: Opaque
data:
  AWS_ACCESS_KEY_ID: UkVQTEFDRS1XSVRILUFXUy1BQ0NFU1MtS0VZ
  AWS_SECRET_ACCESS_KEY: UkVQTEFDRS1XSVRILUFXUy1TRUNSRVQtS0VZ
```

Note: The following command can be used to get a base64-encoded string from a plain text one: `$ echo -n 'plain-text-string' | base64`

The `name` value is the Kubernetes secret name which will be used further, and `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY` are the keys to access S3 storage (and obviously they should contain proper values to make this access possible). To have effect secrets file should be applied with the appropriate command to create the secret object, e.g. `kubectl apply -f deploy/backup-s3.yaml` (for Kubernetes).

Backups schedule is defined in the `backup` section of the deploy/cr.yaml file. This section contains following subsections:

- `storages` subsection contains data needed to access the S3-compatible cloud to store backups.
- `schedule` subsection allows to actually schedule backups (the schedule is specified in crontab format).

Here is an example of deploy/cr.yaml which uses Amazon S3 storage for backups:

```yaml
...
backup:
  ...
  storages:
    s3-us-west:
      type: s3
      s3:
        bucket: S3-BACKUP-BUCKET-NAME-HERE
        region: us-west-2
        credentialsSecret: my-cluster-name-backup-s3
```

(continues on next page)

(continued from previous page)

```
...
schedule:
 - name: "sat-night-backup"
   schedule: "0 0 * * 6"
   keep: 3
   storageName: s3-us-west
...
```

if you use some S3-compatible storage instead of the original Amazon S3, the endpointUrl is needed in the *s3* subsection which points to the actual cloud used for backups and is specific to the cloud provider. For example, using Google Cloud involves the following endpointUrl:

```
endpointUrl: https://storage.googleapis.com
```

The options within these three subsections are further explained in the *Custom Resource options*.

One option which should be mentioned separately is `credentialsSecret` which is a Kubernetes secret for backups. Value of this key should be the same as the name used to create the secret object (`my-cluster-name-backup-s3` in the last example).

The schedule is specified in crontab format as explained in *Custom Resource options*.

21.2 Making on-demand backup

To make an on-demand backup, the user should first configure the backup storage in the `backup.storages` subsection of the `deploy/cr.yaml` configuration file in a same way it was done for scheduled backups. When the `deploy/cr.yaml` file contains correctly configured storage and is applied with `kubectl` command, use *a special backup configuration YAML file* with the following contents:

- **backup name** in the `metadata.name` key,
- **Percona XtraDB Cluster name** in the `spec.pxcCluster` key,
- **storage name** from `deploy/cr.yaml` in the `spec.storageName` key,
- **S3 backup finalizer** set by the `metadata.finalizers.delete-s3-backup` key (it triggers the actual deletion of backup files from the S3 bucket when there is a manual or scheduled removal of the corresponding backup object).

The example of the backup configuration file is deploy/backup/backup.yaml.

When the backup destination is configured and applied with *kubectl apply -f deploy/cr.yaml* command, the actual backup command is executed:

```
kubectl apply -f deploy/backup/backup.yaml
```

Note: Storing backup settings in a separate file can be replaced by passing its content to the `kubectl apply` command as follows:

```
cat <<EOF | kubectl apply -f-
apiVersion: pxc.percona.com/v1
kind: PerconaXtraDBClusterBackup
metadata:
  finalizers:
```

(continues on next page)

(continued from previous page)

```
    - delete-s3-backup
  name: backup1
spec:
  pxcCluster: cluster1
  storageName: s3-us-west
EOF
```

21.3 Storing binary logs for point-in-time recovery

Point-in-time recovery functionality allows users to roll back the cluster to a specific transaction, time (or even skip a transaction in some cases). Technically, this feature involves continuously saving binary log updates to the backup storage. Point-in-time recovery is off by default and is supported by the Operator only with Percona XtraDB Cluster versions starting from 8.0.21-12.1.

To be used, it requires setting a number of keys in the `pitr` subsection under the `backup` section of the deploy/cr.yaml file:

- `enabled` key should be set to `true`,
- `storageName` key should point to the name of the storage already configured in the `storages` subsection (currently, only s3-compatible storages are supported),
- `timeBetweenUploads` key specifies the number of seconds between running the binlog uploader.

Following example shows how the `pitr` subsection looks like:

```
backup:
  ...
  pitr:
    enabled: true
    storageName: s3-us-west
    timeBetweenUploads: 60
```

Note: It is recommended to have empty bucket/directory which holds binlogs (with no binlogs or files from previous attempts or other clusters) when you enable point-in-time recovery.

Note: Purging binlogs before they are transferred to backup storage will break point-in-time recovery.

21.4 Storing backup on Persistent Volume

Here is an example of the `deploy/cr.yaml` backup section fragment, which configures a private volume for filesystem-type storage:

```
...
backup:
  ...
```

(continues on next page)

(continued from previous page)

```
  storages:
    fs-pvc:
      type: filesystem
      volume:
        persistentVolumeClaim:
          accessModes: [ "ReadWriteOnce" ]
          resources:
            requests:
              storage: 6Gi
  ...
```

Note: Please take into account that 6Gi storage size specified in this example may be insufficient for the real-life setups; consider using tens or hundreds of gigabytes. Also, you can edit this option later, and changes will take effect after applying the updated `deploy/cr.yaml` file with `kubectl`.

21.5 Enabling compression for backups

There is a possibility to enable LZ4 compression for backups.

Note: This feature is available only with Percona XtraDB Cluster 8.0 and not Percona XtraDB Cluster 5.7.

To enable compression, use *pxc.configuration* key in the `deploy/cr.yaml` configuration file to supply Percona XtraDB Cluster nodes with two additional `my.cnf` options under its `[sst]` and `[xtrabackup]` sections as follows:

```
pxc:
  image: percona/percona-xtradb-cluster:8.0.19-10.1
  configuration: |
    ...
    [sst]
    xbstream-opts=--decompress
    [xtrabackup]
    compress=lz4
    ...
```

When enabled, compression will be used for both backups and SST.

21.6 Restore the cluster from a previously saved backup

Backup can be restored not only on the Kubernetes cluster where it was made, but also on any Kubernetes-based environment with the installed Operator.

Note: When restoring to a new Kubernetes-based environment, make sure it has a Secrets object with the same user passwords as in the original cluster. More details about secrets can be found in *System Users*.

Following things are needed to restore a previously saved backup:

- Make sure that the cluster is running.
- Find out correct names for the **backup** and the **cluster**. Available backups can be listed with the following command:

  ```
  kubectl get pxc-backup
  ```

  Note: Obviously, you can make this check only on the same cluster on which you have previously made the backup.

  And the following command will list existing Percona XtraDB Cluster names in the current Kubernetes-based environment:

  ```
  kubectl get pxc
  ```

21.6.1 Restoring without point-in-time recovery

When the correct names for the backup and the cluster are known, backup restoration can be done in the following way.

1. Set appropriate keys in the `deploy/backup/restore.yaml` file.
   - set `spec.pxcCluster` key to the name of the target cluster to restore the backup on,
   - **if you are restoring backup on the *same* Kubernetes-based cluster you have** used to save this backup, set `spec.backupName` key to the name of your backup,
   - if you are restoring backup on the Kubernetes-based cluster *different* from one you have used to save this backup, set `spec.backupSource` subsection instead of `spec.backupName` field to point on the appropriate PVC or S3-compatible storage:

     A. If backup was stored on the PVC volume, `backupSource` should contain the storage name (which should be configured in the main CR) and PVC Name:

     ```
     ...
     backupSource:
       destination: pvc/PVC_VOLUME_NAME
       storageName: pvc
       ...
     ```

     B. If backup was stored on the S3-compatible storage, `backupSource` should contain `destination` key equal to the s3 bucket with a special `s3://` prefix, followed by the necessary S3 configuration keys, same as in `deploy/cr.yaml` file:

     ```
     ...
     backupSource:
       destination: s3://S3-BUCKET-NAME/BACKUP-NAME
       s3:
         credentialsSecret: my-cluster-name-backup-s3
         region: us-west-2
         endpointUrl: https://URL-OF-THE-S3-COMPATIBLE-STORAGE
     ...
     ```

2. After that, the actual restoration process can be started as follows:

```
kubectl apply -f deploy/backup/restore.yaml
```

Note: Storing backup settings in a separate file can be replaced by passing its content to the `kubectl apply` command as follows:

```
cat <<EOF | kubectl apply -f-
apiVersion: "pxc.percona.com/v1"
kind: "PerconaXtraDBClusterRestore"
metadata:
  name: "restore1"
spec:
  pxcCluster: "cluster1"
  backupName: "backup1"
EOF
```

21.6.2 Restoring backup with point-in-time recovery

Note: Disable the point-in-time functionality on the existing cluster before restoring a backup on it, regardless of whether the backup was made with point-in-time recovery or without it.

If the point-in-time recovery feature *was enabled*, you can put additional restoration parameters to the `restore.yaml` file `pitr` section for the most fine-grained restoration.

- `backupSource` key should contain `destination` key equal to the s3 bucket with a special `s3://` prefix, followed by the necessary S3 configuration keys, same as in `deploy/cr.yaml` file: `s3://S3-BUCKET-NAME/BACKUP-NAME`,
- `type` key can be equal to one of the following options, * `date` - roll back to specific date, * `transaction` - roll back to specific transaction, * `latest` - recover to the latest possible transaction,
- `date` key is used with `type=date` option - it contains value in datetime format,
- `gtidSet` key is used with `type=transaction` option - it contains exact GTID or GTIDSet (the restore will not include the transaction with specified GTID, but the one before it),
- if you have necessary backup storage mentioned in the `backup.storages` subsection of the `deploy/cr.yaml` configuration file, you can just set `backupSource.storageName` key in the `deploy/backup/restore.yaml` file to the name of the appropriate storage,
- if there is no necessary backup storage in `deploy/cr.yaml`, set your storage details in the `backupSource.s3` subsection instead of using the `backupSource.storageName` field:

```
...
backupSource:
  s3:
    bucket: S3-BUCKET-NAME
    credentialsSecret: my-cluster-name-backup-s3
    endpointUrl: https://URL-OF-THE-S3-COMPATIBLE-STORAGE
    region: us-west-2
...
```

The resulting `restore.yaml` file may look as follows:

```yaml
apiVersion: pxc.percona.com/v1
kind: PerconaXtraDBClusterRestore
metadata:
  name: restore1
spec:
  pxcCluster: cluster1
  backupName: backup1
  pitr:
    type: date
    date: "2020-12-31 09:37:13"
    backupSource:
      storageName: "s3-us-west"
```

The actual restoration process can be started as follows:

```
kubectl apply -f deploy/backup/restore.yaml
```

Note: Storing backup settings in a separate file can be replaced by passing its content to the `kubectl apply` command as follows:

```
cat <<EOF | kubectl apply -f-
apiVersion: "pxc.percona.com/v1"
kind: "PerconaXtraDBClusterRestore"
metadata:
  name: "restore1"
spec:
  pxcCluster: "cluster1"
  backupName: "backup1"
  pitr:
    type: date
    date: "2020-12-31 09:37:13"
    backupSource:
      storageName: "s3-us-west"
EOF
```

21.7 Delete the unneeded backup

The maximum amount of stored backups is controlled by the *backup.schedule.keep* option (only successful backups are counted). Older backups are automatically deleted, so that amount of stored backups do not exceed this number. Setting `keep=0` or removing this option from `deploy/cr.yaml` disables automatic deletion of backups.

Manual deleting of a previously saved backup requires not more than the backup name. This name can be taken from the list of available backups returned by the following command:

```
kubectl get pxc-backup
```

When the name is known, backup can be deleted as follows:

```
kubectl delete pxc-backup/<backup-name>
```

21.8 Copy backup to a local machine

Make a local copy of a previously saved backup requires not more than the backup name. This name can be taken from the list of available backups returned by the following command:

```
kubectl get pxc-backup
```

When the name is known, backup can be downloaded to the local machine as follows:

```
./deploy/backup/copy-backup.sh <backup-name> path/to/dir
```

For example, this downloaded backup can be restored to the local installation of Percona Server:

```
service mysqld stop
rm -rf /var/lib/mysql/*
cat xtrabackup.stream | xbstream -x -C /var/lib/mysql
xtrabackup --prepare --target-dir=/var/lib/mysql
chown -R mysql:mysql /var/lib/mysql
service mysqld start
```

CHAPTER

TWENTYTWO

PAUSE/RESUME PERCONA XTRADB CLUSTER

There may be external situations when it is needed to shutdown the Percona XtraDB Cluster for a while and then start it back up (some works related to the maintenance of the enterprise infrastructure, etc.).

The `deploy/cr.yaml` file contains a special `spec.pause` key for this. Setting it to `true` gracefully stops the cluster:

```
spec:
  .......
  pause: true
```

Pausing the cluster may take some time, and when the process is over, you will see only the Operator Pod running:

```
$ kubectl get pods
NAME                                              READY   STATUS    RESTARTS   AGE
percona-xtradb-cluster-operator-79966668bd-rswbk  1/1     Running   0          12m
```

To start the cluster after it was shut down just revert the `spec.pause` key to `false`.

Starting the cluster will take time. The process is over when all Pods have reached their Running status:

```
NAME                                              READY   STATUS    RESTARTS   AGE
cluster1-haproxy-0                                2/2     Running   0          6m17s
cluster1-haproxy-1                                2/2     Running   0          4m59s
cluster1-haproxy-2                                2/2     Running   0          4m36s
cluster1-pxc-0                                    3/3     Running   0          6m17s
cluster1-pxc-1                                    3/3     Running   0          5m3s
cluster1-pxc-2                                    3/3     Running   0          3m56s
percona-xtradb-cluster-operator-79966668bd-rswbk  1/1     Running   0          9m54s
```

CHAPTER

TWENTYTHREE

UPDADE PERCONA DISTRIBUTION FOR MYSQL OPERATOR

Starting from version 1.1.0, Percona Distribution for MySQL Operator allows upgrades to newer versions. This includes upgrades of the Operator itself, and upgrades of the Percona XtraDB Cluster.

23.1 Upgrading the Operator

The Operator upgrade includes the following steps.

1. Update the Custom Resource Definition file for the Operator, taking it from the official repository on Github, and do the same for the Role-based access control:

```
$ kubectl apply -f https://raw.githubusercontent.com/percona/percona-xtradb-cluster-
↪operator/v1.9.0/deploy/crd.yaml
$ kubectl apply -f https://raw.githubusercontent.com/percona/percona-xtradb-cluster-
↪operator/v1.9.0/deploy/rbac.yaml
```

2. Now you should apply a patch to your deployment, supplying necessary image name with a newer version tag. This is done with the `kubectl patch deployment` command. You can found proper image name *in the list of certified images*. For example, updating to the `1.9.0` version should look as follows.

```
$ kubectl patch deployment percona-xtradb-cluster-operator \
  -p'{"spec":{"template":{"spec":{"containers":[{"name":"percona-xtradb-cluster-
↪operator","image":"percona/percona-xtradb-cluster-operator:1.9.0"}]}}}}'
```

3. The deployment rollout will be automatically triggered by the applied patch. You can track the rollout process in real time with the `kubectl rollout status` command with the name of your cluster:

```
$ kubectl rollout status deployments percona-xtradb-cluster-operator
```

23.2 Upgrading Percona XtraDB Cluster

23.2.1 Automatic upgrade

Starting from version 1.5.0, the Operator can do fully automatic upgrades to the newer versions of Percona XtraDB Cluster within the method named *Smart Updates*.

To have this upgrade method enabled, make sure that the `updateStrategy` key in the `deploy/cr.yaml` configuration file is set to `SmartUpdate`.

When automatic updates are enabled, the Operator will carry on upgrades according to the following algorithm. It will query a special *Version Service* server at scheduled times to obtain fresh information about version numbers and valid image paths needed for the upgrade. If the current version should be upgraded, the Operator updates the CR to reflect the new image paths and carries on sequential Pods deletion in a safe order, allowing StatefulSet to redeploy the cluster Pods with the new image.

The upgrade details are set in the `upgradeOptions` section of the `deploy/cr.yaml` configuration file. Make the following edits to configure updates:

1. Set the `apply` option to one of the following values:

   - `Recommended` - automatic upgrades will choose the most recent version of software flagged as Recommended (for clusters created from scratch, the Percona XtraDB Cluster 8.0 version will be selected instead of the Percona XtraDB Cluster 5.7 one regardless of the image path; for already existing clusters, the 8.0 vs. 5.7 branch choice will be preserved),
   - `8.0-recommended`, `5.7-recommended` - same as above, but preserves specific major Percona XtraDB Cluster version for newly provisioned clusters (ex. 8.0 will not be automatically used instead of 5.7),
   - `Latest` - automatic upgrades will choose the most recent version of the software available,
   - `8.0-latest`, `5.7-latest` - same as above, but preserves specific major Percona XtraDB Cluster version for newly provisioned clusters (ex. 8.0 will not be automatically used instead of 5.7),
   - *version number* - specify the desired version explicitly (version numbers are specified as `8.0.23-14.1`, `5.7.34-31.51`, etc.),
   - `Never` or `Disabled` - disable automatic upgrades

   Note: When automatic upgrades are disabled by the `apply` option, Smart Update functionality will continue working for changes triggered by other events, such as updating a ConfigMap, rotating a password, or changing resource values.

2. Make sure the `versionServiceEndpoint` key is set to a valid Version Server URL (otherwise Smart Updates will not occur).

   A. You can use the URL of the official Percona's Version Service (default). Set `versionServiceEndpoint` to `https://check.percona.com`.

   B. Alternatively, you can run Version Service inside your cluster. This can be done with the `kubectl` command as follows:

   ```
   $ kubectl run version-service --image=perconalab/version-service --env="SERVE_
   ↪HTTP=true" --port 11000 --expose
   ```

Note: Version Service is never checked if automatic updates are disabled. If automatic updates are enabled, but Version Service URL can not be reached, upgrades will not occur.

3. Use the `schedule` option to specify the update checks time in CRON format.

The following example sets the midnight update checks with the official Percona's Version Service:

```
spec:
  updateStrategy: SmartUpdate
  upgradeOptions:
    apply: Recommended
    versionServiceEndpoint: https://check.percona.com
    schedule: "0 0 * * *"
...
```

23.2.2 Semi-automatic upgrade

Semi-automatic update of Percona XtraDB Cluster should be used with the Operator version 1.5.0 or earlier. For all newer versions, use *automatic update* instead.

1. Edit the `deploy/cr.yaml` file, setting `updateStrategy` key to `RollingUpdate`.
2. Now you should apply a patch to your Custom Resource, setting necessary image names with a newer version tag. Also, you should specify the Operator version for your Percona XtraDB Cluster as a `crVersion` value. This version should be lower or equal to the version of the Operator you currently have in your Kubernetes environment.

Note: Only the incremental update to a nearest minor version of the Operator is supported (for example, update from 1.4.0 to 1.5.0). To update to a newer version, which differs from the current version by more than one, make several incremental updates sequentially.

Patching Custom Resource is done with the `kubectl patch pxc` command. Actual image names can be found *in the list of certified images*. For example, updating to the `1.9.0` version should look as follows, depending on whether you are using Percona XtraDB Cluster 5.7 or 8.0.

A. For Percona XtraDB Cluster 5.7 run the following:

```
kubectl patch pxc cluster1 --type=merge --patch '{
   "spec": {
       "crVersion":"1.9.0",
       "pxc":{ "image": "percona/percona-xtradb-cluster:5.7.34-31.51" },
       "proxysql": { "image": "percona/percona-xtradb-cluster-operator:1.9.0-
↪proxysql" },
       "haproxy":  { "image": "percona/percona-xtradb-cluster-operator:1.9.0-
↪haproxy" },
       "backup":   { "image": "percona/percona-xtradb-cluster-operator:1.9.0-
↪pxc5.7-backup" },
       "logcollector": { "image": "percona/percona-xtradb-cluster-operator:1.9.
↪0-logcollector" },
       "pmm":      { "image": "percona/pmm-client:2.18.0" }
   }}'
```

B. For Percona XtraDB Cluster 8.0 run the following:

```
kubectl patch pxc cluster1 --type=merge --patch '{
   "spec": {
       "crVersion":"1.9.0",
       "pxc":{ "image": "percona/percona-xtradb-cluster:8.0.23-14.1" },
       "proxysql": { "image": "percona/percona-xtradb-cluster-operator:1.9.0-
↪proxysql" },
       "haproxy":  { "image": "percona/percona-xtradb-cluster-operator:1.9.0-
↪haproxy" },
       "backup":   { "image": "percona/percona-xtradb-cluster-operator:1.9.0-
↪pxc8.0-backup" },
       "logcollector": { "image": "percona/percona-xtradb-cluster-operator:1.9.
↪0-logcollector" },
       "pmm":      { "image": "percona/pmm-client:2.18.0" }
   }}'
```

3. The deployment rollout will be automatically triggered by the applied patch. You can track the rollout process in real time with the `kubectl rollout status` command with the name of your cluster:

```
$ kubectl rollout status sts cluster1-pxc
```

23.2.3 Manual upgrade

Manual update of Percona XtraDB Cluster should be used with the Operator version 1.5.0 or earlier. For all newer versions, use *automatic update* instead.

1. Edit the `deploy/cr.yaml` file, setting `updateStrategy` key to `OnDelete`.
2. Now you should apply a patch to your Custom Resource, setting necessary image names with a newer version tag. Also, you should specify the Operator version for your Percona XtraDB Cluster as a `crVersion` value. This version should be lower or equal to the version of the Operator you currently have in your Kubernetes environment.

Note: Only the incremental update to a nearest minor version of the Operator is supported (for example, update from 1.4.0 to 1.5.0). To update to a newer version, which differs from the current version by more than one, make several incremental updates sequentially.

Patching Custom Resource is done with the `kubectl patch pxc` command. Actual image names can be found *in the list of certified images*. For example, updating to the `1.9.0` version should look as follows, depending on whether you are using Percona XtraDB Cluster 5.7 or 8.0.

A. For Percona XtraDB Cluster 5.7 run the following:

```
$ kubectl patch pxc cluster1 --type=merge --patch '{
    "spec": {
       "crVersion":"1.9.0",
       "pxc":{ "image": "percona/percona-xtradb-cluster:5.7.34-31.51" },
       "proxysql": { "image": "percona/percona-xtradb-cluster-operator:1.9.0-
↪proxysql" },
       "haproxy":  { "image": "percona/percona-xtradb-cluster-operator:1.9.0-
↪haproxy" },
       "backup":   { "image": "percona/percona-xtradb-cluster-operator:1.9.0-
↪pxc5.7-backup" },
```

(continues on next page)

(continued from previous page)

```
        "logcollector": { "image": "percona/percona-xtradb-cluster-operator:1.9.
↪0-logcollector" },
        "pmm":       { "image": "percona/pmm-client:2.18.0" }
    }}'
```

B. For Percona XtraDB Cluster 8.0 run the following:

```
$ kubectl patch pxc cluster1 --type=merge --patch '{
    "spec": {
        "crVersion":"1.9.0",
        "pxc":{ "image": "percona/percona-xtradb-cluster:8.0.23-14.1" },
        "proxysql": { "image": "percona/percona-xtradb-cluster-operator:1.9.0-
↪proxysql" },
        "haproxy":  { "image": "percona/percona-xtradb-cluster-operator:1.9.0-
↪haproxy" },
        "backup":   { "image": "percona/percona-xtradb-cluster-operator:1.9.0-
↪pxc8.0-backup" },
        "logcollector": { "image": "percona/percona-xtradb-cluster-operator:1.9.
↪0-logcollector" },
        "pmm":       { "image": "percona/pmm-client:2.18.0" }
    }}'
```

3. The Pod with the newer Percona XtraDB Cluster image will start after you delete it. Delete targeted Pods manually one by one to make them restart in desired order:

   1. Delete the Pod using its name with the command like the following one:

   ```
   $ kubectl delete pod cluster1-pxc-2
   ```

   2. Wait until Pod becomes ready:

   ```
   $ kubectl get pod cluster1-pxc-2
   ```

   The output should be like this:

   ```
   NAME             READY   STATUS    RESTARTS   AGE
   cluster1-pxc-2   1/1     Running   0          3m33s
   ```

4. The update process is successfully finished when all Pods have been restarted.

CHAPTER

TWENTYFOUR

SCALE PERCONA XTRADB CLUSTER ON KUBERNETES AND OPENSHIFT

One of the great advantages brought by Kubernetes and the OpenShift platform is the ease of an application scaling. Scaling an application results in adding or removing the Pods and scheduling them to available Kubernetes nodes.

Size of the cluster is controlled by a *size key* in the *Custom Resource options* configuration. That's why scaling the cluster needs nothing more but changing this option and applying the updated configuration file. This may be done in a specifically saved config, or on the fly, using the following command:

```
$ kubectl scale --replicas=5 pxc/cluster1
```

In this example we have changed the size of the Percona XtraDB Cluster to 5 instances.

24.1 Increase the Persistent Volume Claim size

Kubernetes manages storage with a PersistentVolume (PV), a segment of storage supplied by the administrator, and a PersistentVolumeClaim (PVC), a request for storage from a user. In Kubernetes v1.11 the feature was added to allow a user to increase the size of an existing PVC object. The user cannot shrink the size of an existing PVC object. Certain volume types support, by default, expanding PVCs (details about PVCs and the supported volume types can be found in Kubernetes documentation)

The following are the steps to increase the size:

1. Extract and backup the yaml file for the cluster

   ```
   kubectl get pxc cluster1 -o yaml --export > CR_backup.yaml
   ```

2. Now you should delete the cluster.

   > **Warning:** Make sure that *delete-pxc-pvc* finalizer is not set in your custom resource, **otherwise all cluster data will be lost!**

   You can use the following command to delete the cluster:

   ```
   kubectl delete -f CR_backup.yaml
   ```

3. For each node, edit the yaml to resize the PVC object.

   ```
   kubectl edit pvc datadir-cluster1-pxc-0
   ```

   In the yaml, edit the spec.resources.requests.storage value.

```
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 6Gi
```

Perform the same operation on the other nodes.

```
kubectl edit pvc datadir-cluster1-pxc-1
kubectl edit pvc datadir-cluster1-pxc-2
```

4. In the CR configuration file, use vim or another text editor to edit the PVC size.

```
vim CR_backup.yaml
```

5. Apply the updated configuration to the cluster.

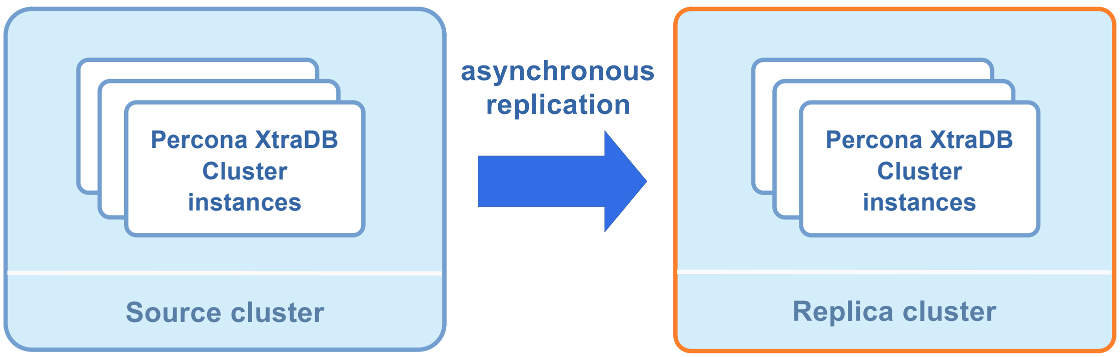
```
kubectl apply -f CR_backup.yaml
```

CHAPTER

TWENTYFIVE

SET UP PERCONA XTRADB CLUSTER CROSS-SITE REPLICATION

The cross-site replication involves configuring one Percona XtraDB Cluster as *Source*, and another Percona XtraDB Cluster as *Replica* to allow an asynchronous replication between them:



The Operator automates configuration of *Source* and *Replica* Percona XtraDB Clusters, but the feature itself is not bound to Kubernetes. Either *Source* or *Replica* can run outside of Kubernetes, be regular MySQL and be out of the Operators' control.

This feature can be useful in several cases: for example, it can simplify migration from on-premises to the cloud with replication, and it can be really helpful in case of the disaster recovery too.

Note: Cross-site replication is based on Automatic Asynchronous Replication Connection Failover. Therefore it requires MySQL 8.0.22+ (Percona XtraDB Cluster 8.0.22+) to work.

Setting up MySQL for asynchronous replication without the Operator is described here and is out of the scope for this document.

Configuring the cross-site replication for the cluster controlled by the Operator is explained in the following subsections.

25.1 Configuring cross-site replication on Source instances

You can configure *Source* instances for cross-site replication with `spec.pxc.replicationChannels` subsection in the `deploy/cr.yaml` configuration file. It is an array of channels, and you should provide the following keys for the channel in your *Source* Percona XtraDB Cluster:

- `pxc.replicationChannels.[].name` key is the name of the channel,
- `pxc.replicationChannels.[].isSource` key should be set to `true`.

Here is an example:

```
spec:
  pxc:
    replicationChannels:
    - name: pxc1_to_pxc2
      isSource: true
```

The cluster will be ready for asynchronous replication when you apply changes as usual:

```
$ kubectl apply -f deploy/cr.yaml
```

25.2 Exposing instances of Percona XtraDB Cluster

You need to expose every Percona XtraDB Cluster Pod of the *Source* cluster to make it possible for the *Replica* cluster to connect. This is done through the `pxc.expose` section in the `deploy/cr.yaml` configuration file as follows.

```
spec:
  pxc:
    expose:
      enabled: true
      type: LoadBalancer
      loadBalancerSourceRanges:
        - 10.0.0.0/8
```

Note: This will create a LoadBalancer per each Percona XtraDB Cluster Pod. Please take into account, that you will need this IP address to be external for cross-region replication in most cases.

25.3 Configuring cross-site replication on Replica instances

You can configure *Replica* instances for cross-site replication with `spec.pxc.replicationChannels` subsection in the `deploy/cr.yaml` configuration file. It is an array of channels, and you should provide the following keys for the channel in your *Replica* Percona XtraDB Cluster:

- `pxc.replicationChannels.[].name` key is the name of the channel,
- `pxc.replicationChannels.[].isSource` key should be set to `false`,
- `pxc.replicationChannels.[].sourcesList` is the list of *Source* cluster names from which Replica should get the data,
- `pxc.replicationChannels.[].sourcesList.[].host` is the host name or IP-address of the Source,

- `pxc.replicationChannels.[].sourcesList.[].port` is the port of the source (`3306` port will be used if nothing specified),
- `pxc.replicationChannels.[].sourcesList.[].weight` is the *weight* of the source (`100` by default).

Here is the example:

```
spec:
  pxc:
    replicationChannels:
    - name: uspxc1_to_pxc2
      isSource: false
      sourcesList:
      - host: pxc1.source.percona.com
        port: 3306
        weight: 100
      - host: pxc2.source.percona.com
      - host: pxc3.source.percona.com
    - name: eu_to_pxc2
      isSource: false
      sourcesList:
      - host: pxc1.source.percona.com
        port: 3306
        weight: 100
      - host: pxc2.source.percona.com
      - host: pxc3.source.percona.com
```

The cluster will be ready for asynchronous replication when you apply changes as usual:

```
$ kubectl apply -f deploy/cr.yaml
```

25.4 System user for replication

Replication channel demands a special *system user* with same credentials on both *Source* and *Replica*. The Operator creates a system-level Percona XtraDB Cluster user named `replication` for this purpose, with credentials stored in a Secret object *along with other system users*.

You can change a password for this user as follows:

```
kubectl patch secret/my-cluster-name-secrets -p '{"data":{"replication": '$(echo -n new_
↪password | base64)'}}'
```

If the cluster is outside of Kubernetes and is not under the Operator's control, the appropriate user with necessary permissions should be created manually.

CHAPTER

TWENTYSIX

MONITORING

Percona Monitoring and Management (PMM) provides an excellent solution to monitor Percona XtraDB Cluster.

Note: Only PMM 2.x versions are supported by the Operator.

PMM is a client/server application. *PMM Client* runs on each node with the database you wish to monitor: it collects needed metrics and sends gathered data to *PMM Server*. As a user, you connect to PMM Server to see database metrics on a number of dashboards.

That's why PMM Server and PMM Client need to be installed separately.

26.1 Installing the PMM Server

PMM Server runs as a *Docker image*, a *virtual appliance*, or on an *AWS instance*. Please refer to the official PMM documentation for the installation instructions.

26.2 Installing the PMM Client

The following steps are needed for the PMM client installation in your Kubernetes-based environment:

1. The PMM client installation is initiated by updating the `pmm` section in the deploy/cr.yaml file.

   - set `pmm.enabled=true`
   - set the `pmm.serverHost` key to your PMM Server hostname,
   - check that the `serverUser` key contains your PMM Server user name (`admin` by default),
   - make sure the `pmmserver` key in the deploy/secrets.yaml secrets file contains the password specified for the PMM Server during its installation

   Note: You use `deploy/secrets.yaml` file to *create* Secrets Object. The file contains all values for each key/value pair in a convenient plain text format. But the resulting Secrets contain passwords stored as base64-encoded strings. If you want to *update* password field, you'll need to encode the value into base64 format. To do this, you can run `echo -n "password" | base64` in your local shell to get valid values. For example, setting the PMM Server user's password to *new_password*` in the `my-cluster-name-secrets` object can be done with the following command:

   ```
   kubectl patch secret/my-cluster-name-secrets -p '{"data":{"pmmserver": '$(echo -
   ↪n new_password | base64)'}}'
   ```

- you can also use `pmm.pxcParams` and `pmm.proxysqlParams` keys to specify additional parameters for pmm-admin add mysql and pmm-admin add mysql commands respectively, if needed.

Note: Please take into account that Operator automatically manages common Percona XtraDB Cluster Service Monitoring parameters mentioned in the officiall PMM documentation, such like username, password, service-name, host, etc. Assigning values to these parameters is not recommended and can negatively affect the functionality of the PMM setup carried out by the Operator.

Apply changes with the `kubectl apply -f deploy/secrets.yaml` command.

When done, apply the edited `deploy/cr.yaml` file:

```
$ kubectl apply -f deploy/cr.yaml
```

2. Check that corresponding Pods are not in a cycle of stopping and restarting. This cycle occurs if there are errors on the previous steps:

```
$ kubectl get pods
$ kubectl logs cluster1-pxc-node-0 -c pmm-client
```

3. Now you can access PMM via *https* in a web browser, with the login/password authentication, and the browser is configured to show Percona XtraDB Cluster metrics.

CHAPTER

TWENTYSEVEN

CRASH RECOVERY

27.1 What does the full cluster crash mean?

A full cluster crash is a situation when all database instances where shut down in random order. Being rebooted after such situation, Pod is continuously restarting, and generates the following errors in the log:

```
It may not be safe to bootstrap the cluster from this node. It was not the last one to␣
↪leave the cluster and may not contain all the updates.
To force cluster bootstrap with this node, edit the grastate.dat file manually and set␣
↪safe_to_bootstrap to 1
```

Note: To avoid this, shutdown your cluster correctly as it is written in *Pause/resume Percona XtraDB Cluster*.

The Percona Distribution for MySQL Operator provides two ways of recovery after a full cluster crash.

The Operator is providing automatic crash recovery (by default) and semi-automatic recovery starting from the version `1.7`. For the previous Operator versions, crash recovery can be done *manually*.

27.2 Automatic Crash Recovery

Crash recovery can be done automatically. This behavior is controlled by the `pxc.autoRecovery` option in the `deploy/cr.yaml` configuration file.

The default value for this option is `true`, which means that automatic recovery is turned on.

If this option is set to `false`, automatic crash recovery is not done, but semi-automatic recovery is still possible.

In this case you need to get the log from pxc container from all Pods using the following command:

```
$ for i in $(seq 0 $(($(kubectl get pxc cluster1 -o jsonpath='{.spec.pxc.size}')-1)));␣
↪do echo "###############cluster1-pxc-$i##############"; kubectl logs cluster1-pxc-$i -
↪c pxc | grep '(seqno):' ; done
```

The output of this command should be similar to the following one:

```
###############cluster1-pxc-0##############
It is cluster1-pxc-0.cluster1-pxc.default.svc.cluster.local node with sequence number␣
↪(seqno): 18
###############cluster1-pxc-1##############
It is cluster1-pxc-1.cluster1-pxc.default.svc.cluster.local node with sequence number␣
↪(seqno): 18
```

(continues on next page)

(continued from previous page)

```
##############cluster1-pxc-2##############
It is cluster1-pxc-2.cluster1-pxc.default.svc.cluster.local node with sequence number␣
↪(seqno): 19
```

Now find the Pod with the largest `seqno` (it is `cluster1-pxc-2` in the above example).

Now execute the following commands to start this instance:

```
$ kubectl exec cluster1-pxc-2 -c pxc -- sh -c 'kill -s USR1 1'
```

27.3 Manual Crash Recovery

> **Warning:** This method includes a lot of operations, and therefore, it is intended for advanced users only!

This method involves the following steps:

- swap the original Percona XtraDB Cluster image with the *debug image*, which does not reboot after the crash, and force all Pods to run it,
- find the Pod with the most recent Percona XtraDB Cluster data, run recovery on it, start `mysqld`, and allow the cluster to be restarted,
- revert all temporary substitutions.

Let's assume that a full crash did occur for the cluster named `cluster1`, which is based on three Percona XtraDB Cluster Pods.

Note: The following commands are written for Percona XtraDB Cluster 8.0. The same steps are also for Percona XtraDB Cluster 5.7 unless specifically indicated otherwise.

1. Check the current Update Strategy with the following command to make sure *Smart Updates* are turned off during the recovery:

   ```
   $ kubectl get pxc cluster1 -o jsonpath='{.spec.updateStrategy}'
   ```

   If the returned value is `SmartUpdate`, please change it to `onDelete` with the following command:

   ```
   $ kubectl patch pxc cluster1 --type=merge --patch '{"spec": {"updateStrategy":
   ↪"OnDelete" }}'
   ```

2. Change the normal PXC image inside the cluster object to the debug image:

   Note: Please make sure the Percona XtraDB Cluster version for the debug image matches the version currently in use in the cluster. You can run the following command to find out which Percona XtraDB Cluster image is in use:

   ```
   $ kubectl get pxc cluster1 -o jsonpath='{.spec.pxc.image}'
   ```

```
$ kubectl patch pxc cluster1 --type="merge" -p '{"spec":{"pxc":{"image":"percona/
↪percona-xtradb-cluster:8.0.23-14.1-debug"}}}'
```

Note: For Percona XtraDB Cluster 5.7 this command should be as follows:

```
$ kubectl patch pxc cluster1 --type="merge" -p '{"spec":{"pxc":{"image":"percona/
↪percona-xtradb-cluster:5.7.34-31.51-debug"}}}'
```

---

3. Restart all Pods:

```
$ for i in $(seq 0 $(($(kubectl get pxc cluster1 -o jsonpath='{.spec.pxc.size}')-
↪1))); do kubectl delete pod cluster1-pxc-$i --force --grace-period=0; done
```

4. Wait until the Pod `0` is ready, and execute the following code (it is required for the Pod liveness check):

```
$ for i in $(seq 0 $(($(kubectl get pxc cluster1 -o jsonpath='{.spec.pxc.size}')-
↪1))); do until [[ $(kubectl get pod cluster1-pxc-$i -o jsonpath='{.status.phase}
↪') == 'Running' ]]; do sleep 10; done; kubectl exec cluster1-pxc-$i -- touch /var/
↪lib/mysql/sst_in_progress; done
```

5. Wait for all Percona XtraDB Cluster Pods to start, and execute the following code to make sure no mysqld processes are running:

```
$ for i in $(seq $(($(kubectl get pxc cluster1 -o jsonpath='{.spec.pxc.size}')-1)));
↪ do pid=$(kubectl exec cluster1-pxc-$i -- ps -C mysqld-ps -o pid=); if [[ -n "$pid
↪" ]]; then kubectl exec cluster1-pxc-$i -- kill -9 $pid; fi;  done
```

6. Wait for all Percona XtraDB Cluster Pods to start, then find the Percona XtraDB Cluster instance with the most recent data - i.e. the one with the highest sequence number (seqno):

```
$ for i in $(seq 0 $(($(kubectl get pxc cluster1 -o jsonpath='{.spec.pxc.size}')-
↪1))); do echo "###############cluster1-pxc-$i##############"; kubectl exec␣
↪cluster1-pxc-$i -- cat /var/lib/mysql/grastate.dat; done
```

The output of this command should be similar to the following one:

```
###############cluster1-pxc-0##############
# GALERA saved state
version: 2.1
uuid:    7e037079-6517-11ea-a558-8e77af893c93
seqno:   18
safe_to_bootstrap: 0
###############cluster1-pxc-1##############
# GALERA saved state
version: 2.1
uuid:    7e037079-6517-11ea-a558-8e77af893c93
seqno:   18
safe_to_bootstrap: 0
###############cluster1-pxc-2##############
# GALERA saved state
version: 2.1
uuid:    7e037079-6517-11ea-a558-8e77af893c93
```

(continues on next page)

(continued from previous page)

```
seqno:   19
safe_to_bootstrap: 0
```

Now find the Pod with the largest `seqno` (it is `cluster1-pxc-2` in the above example).

7. Now execute the following commands *in a separate shell* to start this instance:

```
$ kubectl exec cluster1-pxc-2 -- mysqld --wsrep_recover
$ kubectl exec cluster1-pxc-2 -- sed -i 's/safe_to_bootstrap: 0/safe_to_bootstrap:␣
↪1/g' /var/lib/mysql/grastate.dat
$ kubectl exec cluster1-pxc-2 -- sed -i 's/wsrep_cluster_address=.*/wsrep_cluster_
↪address=gcomm:\/\//g' /etc/mysql/node.cnf
$ kubectl exec cluster1-pxc-2 -- mysqld
```

The `mysqld` process will initialize the database once again, and it will be available for the incoming connections.

8. Go back *to the previous shell* and return the original Percona XtraDB Cluster image because the debug image is no longer needed:

Note: Please make sure the Percona XtraDB Cluster version for the debug image matches the version currently in use in the cluster.

```
$ kubectl patch pxc cluster1 --type="merge" -p '{"spec":{"pxc":{"image":"percona/
↪percona-xtradb-cluster:8.0.23-14.1"}}}'
```

Note: For Percona XtraDB Cluster 5.7 this command should be as follows:

```
$ kubectl patch pxc cluster1 --type="merge" -p '{"spec":{"pxc":{"image":"percona/
↪percona-xtradb-cluster:5.7.34-31.51"}}}'
```

9. Restart all Pods besides the `cluster1-pxc-2` Pod (the recovery donor).

```
$ for i in $(seq 0 $(($(kubectl get pxc cluster1 -o jsonpath='{.spec.pxc.size}')-
↪1))); do until [[ $(kubectl get pod cluster1-pxc-$i -o jsonpath='{.status.phase}
↪') == 'Running' ]]; do sleep 10; done; kubectl exec cluster1-pxc-$i -- rm /var/
↪lib/mysql/sst_in_progress; done
$ kubectl delete pods --force --grace-period=0 cluster1-pxc-0 cluster1-pxc-1
```

10. Wait for the successful startup of the Pods which were deleted during the previous step, and finally remove the `cluster1-pxc-2` Pod:

```
$ kubectl delete pods --force --grace-period=0 cluster1-pxc-2
```

11. After the Pod startup, the cluster is fully recovered.

Note: If you have changed the update strategy on the 1st step, don't forget to revert it back to `SmartUpdate` with the following command:

```
$ kubectl patch pxc cluster1 --type=merge --patch '{"spec": {"updateStrategy":
↪"SmartUpdate" }}'
```

CHAPTER

TWENTYEIGHT

DEBUG

28.1 Cluster-level logging

Cluster-level logging involves collecting logs from all Percona XtraDB Cluster Pods in the cluster to some persistent storage. This feature gives the logs a lifecycle independent of nodes, Pods and containers in which they were collected. Particularly, it ensures that Pod logs from previous failures are available for later review.

Log collector is turned on by the `logcollector.enabled` key in the `deploy/cr.yaml` configuration file (`true` by default).

The Operator collects logs using Fluent Bit Log Processor, which supports many output plugins and has broad forwarding capabilities. If necessary, Fluent Bit filtering and advanced features can be configured via the `logcollector.configuration` key in the `deploy/cr.yaml` configuration file.

Logs are stored for 7 days and then rotated.

Collected logs can be examined using the following command:

```
kubectl logs cluster1-pxc-1 -c logs
```

Note: Technically, logs are stored on the same Persistent Volume, which is used with the corresponding Percona XtraDB Cluster Pod. Therefore collected logs can be found in `DATADIR` (`var/lib/mysql/`).

28.2 Avoid the restart-on-fail loop for Percona XtraDB Cluster containers

The restart-on-fail loop takes place when the container entry point fails (e.g. `mysqld` crashes). In such a situation, Pod is continuously restarting. Continuous restarts prevent to get console access to the container, and so a special approach is needed to make fixes.

You can prevent such infinite boot loop by putting the Percona XtraDB Cluster containers into the infinity loop *without* starting mysqld. This behavior of the container entry point is triggered by the presence of the `/var/lib/mysql/sleep-forever` file.

For example, you can do it for the `pxc` container of an appropriate Percona XtraDB Cluster instance as follows:

```
kubectl exec -it cluster1-pxc-0 -c pxc -- sh -c 'touch /var/lib/mysql/sleep-forever'
```

If `pxc` container can't start, you can use `logs` container instead:

```
kubectl exec -it cluster1-pxc-0 -c logs -- sh -c 'touch /var/lib/mysql/sleep-forever'
```

The instance will restart automatically and run in its usual way as soon as you remove this file (you can do it with a command similar to the one you have used to create the file, just substitute `touch` to `rm` in it).

28.3 Special debug images

For the cases when Pods are failing for some reason or just show abnormal behavior, the Operator can be used with a special *debug images*. Percona XtraDB Cluster debug image has the following specifics:

- it avoids restarting on fail,
- it contains additional tools useful for debugging (sudo, telnet, gdb, etc.),
- it has debug mode enabled for the logs.

There are debug versions for all *Percona XtraDB Cluster images*: they have same names as normal images with a special `-debug` suffix in their version tag: for example, `percona-xtradb-cluster:8.0.23-14.1-debug`.

To use the debug image instead of the normal one, find the needed image name *in the list of certified images* and set it for the proper key in the `deploy/cr.yaml` configuration file. For example, set the following value of the `pxc.image` key to use the Percona XtraDB Cluster debug image:

- `percona/percona-xtradb-cluster:8.0.23-14.1-debug` for Percona XtraDB Cluster 8.0,
- `percona/percona-xtradb-cluster:5.7.34-31.51-debug` for Percona XtraDB Cluster 5.7.

The Pod should be restarted to get the new image.

Note: When the Pod is continuously restarting, you may have to delete it to apply image changes.

Part VI

Reference

CHAPTER

TWENTYNINE

CUSTOM RESOURCE OPTIONS

Percona XtraDB Cluster managed by the Operator configured via the spec section of the deploy/cr.yaml file.

The metadata part of this file contains the following keys:

- `name` (`my-cluster-name` by default) sets the name of your Percona XtraDB Cluster; it should include only URL-compatible characters, not exceed 22 characters, start with an alphabetic character, and end with an alphanumeric character;
- `finalizers.delete-pods-in-order` if present, activates the Finalizer which controls the proper Pods deletion order in case of the cluster deletion event (on by default).
- `finalizers.delete-pxc-pvc`, `delete-proxysql-pvc` if present, activates the Finalizer which deletes Persistent Volume Claims for Percona XtraDB Cluster Pods after the cluster deletion event (off by default).
- `delete-proxysql-pvc` if present, activates the Finalizer which deletes Persistent Volume Claim for ProxySQL Pod after the cluster deletion event (off by default).

The spec part of the deploy/cr.yaml file contains the following sections:

Key	Value type	Default	Description
upgradeOptions	*subdoc*		Percona XtraDB Cluster upgrade options section
pxc	*subdoc*		Percona XtraDB Cluster general section
haproxy	*subdoc*		HAProxy section
proxysql	*subdoc*		ProxySQL section
pmm	*subdoc*		Percona Monitoring and Management section
backup	*subdoc*		Percona XtraDB Cluster backups section
allowUnsafeConfigurations	boolean	`false`	Prevents users from configuring a cluster with unsafe parameters such as starting the cluster with the number of Percona XtdaDB Cluster instances which is less than 3, more than 5, or is an even number, with less than 2 ProxySQL or HAProxy Pods, or without TLS/SSL certificates (if `false`, unsafe parameters will be automatically changed to safe defaults)
enableCRValidationWebhook	boolean	`true`	Enables or disables schema validation before applying `cr.yaml` (works only in *cluster-wide mode* by default due to access restrictions)
pause	boolean	`false`	Pause/resume: setting it to `true` gracefully stops the cluster, and setting it to `false` after shut down starts the cluster back.
secretsName	string	`my-cluster-secrets`	A name for *users secrets*
crVersion	string	`1.9.0`	Version of the Operator the Custom Resource belongs to
vaultSecretName	string	`keyring-secret-vault`	A secret for the HashiCorp Vault to carry on *Data at Rest Encryption*
sslSecretName	string	`my-cluster-ssl`	A secret with TLS certificate generated for *external* communications, see *Transport Layer Security (TLS)* for details
sslInternalSecretName	string	`my-cluster-ssl-internal`	A secret with TLS certificate generated for *internal* communications, see *Transport Layer Security (TLS)* for details
logCollectorSecretName	string	`my-log-collector-secrets`	A secret for the Fluent Bit Log Collector
updateStrategy	string	`SmartUpdate`	A strategy the Operator uses for *upgrades*

29.1 Upgrade Options Section

The upgradeOptions section in the deploy/cr.yaml file contains various configuration options to control Percona XtraDB Cluster upgrades.

Key	upgradeOptions.versionServiceEndpoint
Value	string
Example	`https://check.percona.com`
Description	The Version Service URL used to check versions compatibility for upgrade
Key	upgradeOptions.apply
Value	string
Example	`Disabled`
Description	Specifies how *updates are processed* by the Operator. `Never` or `Disabled` will completely disable automatic upgrades, otherwise it can be set to `Latest` or `Recommended` or to a specific version string of Percona XtraDB Cluster (e.g. `8.0.19-10.1`) that is wished to be version-locked (so that the user can control the version running, but use automatic upgrades to move between them).
Key	upgradeOptions.schedule
Value	string
Example	`0 2 * * *`
Description	Scheduled time to check for updates, specified in the crontab format

29.2 PXC Section

The pxc section in the deploy/cr.yaml file contains general configuration options for the Percona XtraDB Cluster.

Key	pxc.size
Value	int
Example	`3`
Description	The size of the Percona XtraDB cluster must be 3 or 5 for High Availability. other values are allowed if the `spec.allowUnsafeConfigurations` key is set to true.
Key	pxc.image
Value	string
Example	`percona/percona-xtradb-cluster:8.0.23-14.1`
Description	The Docker image of the Percona cluster used (actual image names for Percona XtraDB Cluster 8.0 and Percona XtraDB Cluster 5.7 can be found *in the list of certified images*)
Key	pxc.autoRecovery
Value	boolean
Example	`true`
Description	Turns *Automatic Crash Recovery* on or off
Key	pxc.expose.enabled
Value Type	boolean
Example	`true`

continues on next page

Table 1 – continued from previous page

Description	Enable or disable exposing `Percona XtraDB Cluster nodes with dedicated IP addresses
Key	pxc.expose.type
Value Type	string
Example	`LoadBalancer`
Description	The Kubernetes Service Type used for xposure
Key	pxc.expose.loadBalancerSourceRanges
Value Type	string
Example	`10.0.0.0/8`
Description	The range of client IP addresses from which the load balancer should be reachable (if not set, there is no limitations)
Key	pxc.expose.annotations
Value Type	string
Example	`networking.gke.io/load-balancer-type: "Internal"`
Description	The Kubernetes annotations
Key	pxc.replicationChannels.name
Value Type	string
Example	`pxc1_to_pxc2`
Description	Name of the replication channel for *cross-site replication*
Key	pxc.replicationChannels.isSource
Value Type	boolean
Example	`false`
Description	Should the cluster act as Source (`true`) or Replica (`false`) in *cross-site replication*
Key	pxc.replicationChannels.sourcesList.host
Value Type	string
Example	`10.95.251.101`
Description	For the *cross-site replication* Replica cluster, this key should contain the hostname or IP address of the Source cluster
Key	pxc.replicationChannels.sourcesList.port
Value Type	int
Example	`3306`
Description	For the *cross-site replication* Replica cluster, this key should contain the Source port number
Key	pxc.replicationChannels.sourcesList.weight
Value Type	int
Example	`100`
Description	For the *cross-site replication* Replica cluster, this key should contain the Source cluster weight
Key	pxc.readinessDelaySec
Value	int
Example	`15`
Description	Adds a delay before a run check to verify the application is ready to process traffic
Key	pxc.livenessDelaySec

continues on next page

Table 1 – continued from previous page

Value	int
Example	`300`
Description	Adds a delay before the run check ensures the application is healthy and capable of processing requests
Key	pxc.configuration
Value	string

Example	`|`
	`[mysqld]`
	`wsrep_debug=ON`
	`wsrep-provider_options=gcache.size=1G;gcache.recover=yes`

Description	The `my.cnf` file options to be passed to Percona XtraDB cluster nodes
Key	pxc.imagePullSecrets.name
Value	string
Example	`private-registry-credentials`
Description	The Kubernetes ImagePullSecret
Key	pxc.priorityClassName
Value	string
Example	`high-priority`
Description	The Kubernetes Pod priority class
Key	pxc.schedulerName
Value	string
Example	`mycustom-scheduler`
Description	The Kubernetes Scheduler
Key	pxc.annotations
Value	label
Example	`iam.amazonaws.com/role:  role-arn`
Description	The Kubernetes annotations
Key	pxc.labels
Value	label
Example	`rack:  rack-22`
Description	Labels are key-value pairs attached to objects
Key	pxc.readinessProbes.initialDelaySeconds
Value	int
Example	`15`
Description	Number of seconds to wait before performing the first readiness probe
Key	pxc.readinessProbes.timeoutSeconds
Value	int
Example	`15`
Description	Number of seconds after the container has started before readiness probes are initiated
Key	pxc.readinessProbes.periodSeconds
Value	int

continues on next page

Table 1 – continued from previous page

Example	30
Description	How often (in seconds) to perform the readiness probe
Key	pxc.readinessProbes.successThreshold
Value	int
Example	1
Description	Minimum consecutive successes for the readiness probe to be considered successful after having failed
Key	pxc.readinessProbes.failureThreshold
Value	int
Example	5
Description	When the readiness probe fails, Kubernetes will try this number of times before marking the Pod Unready
Key	pxc.livenessProbes.initialDelaySeconds
Value	int
Example	300
Description	Number of seconds to wait before performing the first liveness probe
Key	pxc.livenessProbes.timeoutSeconds
Value	int
Example	5
Description	Number of seconds after the container has started before liveness probes are initiated
Key	pxc.livenessProbes.periodSeconds
Value	int
Example	10
Description	How often (in seconds) to perform the liveness probe
Key	pxc.livenessProbes.successThreshold
Value	int
Example	1
Description	Minimum consecutive successes for the liveness probe to be considered successful after having failed
Key	pxc.livenessProbes.failureThreshold
Value	int
Example	3
Description	When the liveness probe fails, Kubernetes will try this number of times before restarting the container
Key	pxc.envVarsSecret
Value	string
Example	my-env-var-secrets
Description	A secret with environment variables, see *Define environment variables* for details
Key	pxc.resources.requests.memory
Value	string
Example	1G

continues on next page

Table 1 – continued from previous page

Description	The Kubernetes memory requests for a Percona XtraDB Cluster container
Key	pxc.resources.requests.cpu
Value	string
Example	`600m`
Description	Kubernetes CPU requests for a Percona XtraDB Cluster container
Key	pxc.resources.requests.ephemeral-storage
Value	string
Example	`1G`
Description	Kubernetes Ephemeral Storage requests for a Percona XtraDB Cluster container
Key	pxc.resources.limits.memory
Value	string
Example	`1G`
Description	Kubernetes memory limits for a Percona XtraDB Cluster container
Key	pxc.resources.limits.cpu
Value	string
Example	`1`
Description	Kubernetes CPU limits for a Percona XtraDB Cluster container
Key	pxc.resources.limits.ephemeral-storage
Value	string
Example	`1G`
Description	Kubernetes Ephemeral Storage limits for a Percona XtraDB Cluster container
Key	pxc.nodeSelector
Value	label
Example	`disktype: ssd`
Description	Kubernetes nodeSelector
Key	pxc.affinity.topologyKey
Value	string
Example	`kubernetes.io/hostname`
Description	The Operator topology key node anti-affinity constraint
Key	pxc.affinity.advanced
Value	subdoc
Example	
Description	In cases where the Pods require complex tuning the *advanced* option turns off the `topologyKey` effect. This setting allows the standard Kubernetes affinity constraints of any complexity to be used
Key	pxc.tolerations
Value	subdoc
Example	`node.alpha.kubernetes.io/unreachable`
Description	Kubernetes Pod tolerations
Key	pxc.podDisruptionBudget.maxUnavailable

continues on next page

Table 1 – continued from previous page

Value	int
Example	`1`
Description	The Kubernetes podDisruptionBudget specifies the number of Pods from the set unavailable after the eviction
Key	pxc.podDisruptionBudget.minAvailable
Value	int
Example	`0`
Description	The Kubernetes podDisruptionBudget Pods that must be available after an eviction
Key	pxc.volumeSpec.emptyDir
Value	string
Example	`{}`
Description	The Kubernetes emptyDir volume The directory created on a node and accessible to the Percona XtraDB Cluster Pod containers
Key	pxc.volumeSpec.hostPath.path
Value	string
Example	`/data`
Description	Kubernetes hostPath The volume that mounts a directory from the host node's filesystem into your Pod. The path property is required
Key	pxc.volumeSpec.hostPath.type
Value	string
Example	`Directory`
Description	The Kubernetes hostPath. An optional property for the hostPath
Key	pxc.volumeSpec.persistentVolumeClaim.storageClassName
Value	string
Example	`standard`
Description	Set the Kubernetes storage class to use with the Percona XtraDB Cluster PersistentVolumeClaim
Key	pxc.volumeSpec.persistentVolumeClaim.accessModes
Value	array
Example	`[ReadWriteOnce]`
Description	The Kubernetes PersistentVolumeClaim access modes for the Percona XtraDB cluster
Key	pxc.volumeSpec.resources.requests.storage
Value	string
Example	`6Gi`
Description	The Kubernetes PersistentVolumeClaim size for the Percona XtraDB cluster
Key	pxc.gracePeriod
Value	int
Example	`600`
Description	The Kubernetes grace period when terminating a Pod
Key	pxc.containerSecurityContext
Value	subdoc
Example	`privileged: true`

continues on next page

Table 1 – continued from previous page

Description	A custom Kubernetes Security Context for a Container to be used instead of the default one
Key	pxc.podSecurityContext
Value	subdoc
Example	`fsGroup: 1001` `supplementalGroups: [1001, 1002, 1003]`
Description	A custom Kubernetes Security Context for a Pod to be used instead of the default one
Key	pxc.serviceAccountName
Value	string
Example	`percona-xtradb-cluster-operator-workload`
Description	The Kubernetes Service Account for Percona XtraDB Cluster Pods
Key	pxc.imagePullPolicy
Value	string
Example	`Always`
Description	The policy used to update images
Key	pxc.runtimeClassName
Value Type	string
Example	`image-rc`
Description	Name of the Kubernetes Runtime Class for Percona XtraDB Cluster Pods
Key	pxc.sidecars.image
Value Type	string
Example	`busybox`
Description	Image for the *custom sidecar container* for Percona XtraDB Cluster Pods
Key	pxc.sidecars.command
Value Type	array
Example	`["/bin/sh"]`
Description	Command for the *custom sidecar container* for Percona XtraDB Cluster Pods
Key	pxc.sidecars.args
Value Type	array
Example	`["-c", "while true; do trap 'exit 0' SIGINT SIGTERM SIGQUIT SIGKILL; done;"]`
Description	Command arguments for the *custom sidecar container* for Percona XtraDB Cluster Pods
Key	pxc.sidecars.name
Value Type	string
Example	`my-sidecar-1`
Description	Name of the *custom sidecar container* for Percona XtraDB Cluster Pods
Key	pxc.sidecarResources.requests.memory
Value	string
Example	`1G`
Description	The Kubernetes memory requests for a Percona XtraDB Cluster sidecar container
Key	pxc.sidecarResources.requests.cpu

continues on next page

Table 1 – continued from previous page

Value	string
Example	`500m`
Description	Kubernetes CPU requests for a Percona XtraDB Cluster sidecar container
Key	pxc.sidecarResources.limits.memory
Value	string
Example	`2G`
Description	Kubernetes memory limits for a Percona XtraDB Cluster sidecar container
Key	pxc.sidecarResources.limits.cpu
Value	string
Example	`600m`
Description	Kubernetes CPU limits for a Percona XtraDB Cluster sidecar container

29.3 HAProxy Section

The `haproxy` section in the deploy/cr.yaml file contains configuration options for the HAProxy service.

Key	haproxy.enabled
Value	boolean
Example	`true`
Description	Enables or disables load balancing with HAProxy Services
Key	haproxy.size
Value	int
Example	`2`
Description	The number of the HAProxy Pods to provide load balancing. It should be 2 or more unless the `spec.allowUnsafeConfigurations` key is set to true.
Key	haproxy.image
Value	string
Example	`percona/percona-xtradb-cluster-operator:1.9.0-haproxy`
Description	HAProxy Docker image to use
Key	haproxy.imagePullPolicy
Value	string
Example	`Always`
Description	The policy used to update images
Key	haproxy.imagePullSecrets.name
Value	string
Example	`private-registry-credentials`
Description	The Kubernetes imagePullSecrets for the HAProxy image
Key	haproxy.configuration
Value	string
Example	
Description	The *custom HAProxy configuration file* contents

continues on next page

Table 2 – continued from previous page

Key	haproxy.annotations
Value	label
Example	`iam.amazonaws.com/role: role-arn`
Description	The Kubernetes annotations metadata
Key	haproxy.labels
Value	label
Example	`rack: rack-22`
Description	Labels are key-value pairs attached to objects
Key	haproxy.readinessProbes.initialDelaySeconds
Value	int
Example	`15`
Description	Number of seconds to wait before performing the first readiness probe
Key	haproxy.readinessProbes.timeoutSeconds
Value	int
Example	`1`
Description	Number of seconds after the container has started before readiness probes are initiated
Key	haproxy.readinessProbes.periodSeconds
Value	int
Example	`5`
Description	How often (in seconds) to perform the readiness probe
Key	haproxy.readinessProbes.successThreshold
Value	int
Example	`1`
Description	Minimum consecutive successes for the readiness probe to be considered successful after having failed
Key	haproxy.readinessProbes.failureThreshold
Value	int
Example	`3`
Description	When the readiness probe fails, Kubernetes will try this number of times before marking the Pod Unready
Key	haproxy.serviceType
Value	string
Example	`ClusterIP`
Description	Specifies the type of Kubernetes Service to be used for HAProxy
Key	haproxy.externalTrafficPolicy
Value	string
Example	`Cluster`
Description	Specifies whether Service for HAProxy should route external traffic to cluster-wide or to node-local endpoints (it can influence the load balancing effectiveness)
Key	haproxy.replicasServiceType

continues on next page

Table 2 – continued from previous page

Value	string
Example	`ClusterIP`
Description	Specifies the type of Kubernetes Service to be used for HAProxy replicas
Key	haproxy.replicasExternalTrafficPolicy
Value	string
Example	`Cluster`
Description	Specifies whether Service for HAProxy replicas should route external traffic to cluster-wide or to node-local endpoints (it can influence the load balancing effectiveness)
Key	haproxy.resources.requests.memory
Value	string
Example	`1G`
Description	The Kubernetes memory requests for the main HAProxy container
Key	haproxy.resources.requests.cpu
Value	string
Example	`600m`
Description	Kubernetes CPU requests for the main HAProxy container
Key	haproxy.resources.limits.memory
Value	string
Example	`1G`
Description	Kubernetes memory limits for the main HAProxy container
Key	haproxy.resources.limits.cpu
Value	string
Example	`700m`
Description	Kubernetes CPU limits for the main HAProxy container
Key	haproxy.sidecarResources.requests.memory
Value	string
Example	`1G`
Description	The Kubernetes memory requests for the sidecar HAProxy containers
Key	haproxy.sidecarResources.requests.cpu
Value	string
Example	`500m`
Description	Kubernetes CPU requests for the sidecar HAProxy containers
Key	haproxy.sidecarResources.limits.memory
Value	string
Example	`2G`
Description	Kubernetes memory limits for the sidecar HAProxy containers
Key	haproxy.sidecarResources.limits.cpu
Value	string
Example	`600m`
Description	Kubernetes CPU limits for the sidecar HAProxy containers

continues on next page

Table 2 – continued from previous page

Key	haproxy.envVarsSecret
Value	string
Example	`my-env-var-secrets`
Description	A secret with environment variables, see *Define environment variables* for details
Key	haproxy.priorityClassName
Value	string
Example	`high-priority`
Description	The Kubernetes Pod Priority class for HAProxy
Key	haproxy.schedulerName
Value	string
Example	`mycustom-scheduler`
Description	The Kubernetes Scheduler
Key	haproxy.nodeSelector
Value	label
Example	`disktype: ssd`
Description	Kubernetes nodeSelector
Key	haproxy.affinity.topologyKey
Value	string
Example	`kubernetes.io/hostname`
Description	The Operator topology key node anti-affinity constraint
Key	haproxy.affinity.advanced
Value	subdoc
Example	
Description	If available it makes a topologyKey node affinity constraint to be ignored
Key	haproxy.tolerations
Value	subdoc
Example	`node.alpha.kubernetes.io/unreachable`
Description	Kubernetes Pod tolerations
Key	haproxy.podDisruptionBudget.maxUnavailable
Value	int
Example	`1`
Description	The Kubernetes podDisruptionBudget specifies the number of Pods from the set unavailable after the eviction
Key	haproxy.podDisruptionBudget.minAvailable
Value	int
Example	`0`
Description	The Kubernetes podDisruptionBudget Pods that must be available after an eviction
Key	haproxy.gracePeriod
Value	int
Example	`30`
Description	The Kubernetes grace period when terminating a Pod

continues on next page

Table 2 – continued from previous page

Key	haproxy.loadBalancerSourceRanges
Value	string
Example	`10.0.0.0/8`
Description	The range of client IP addresses from which the load balancer should be reachable (if not set, there is no limitations)
Key	haproxy.serviceAnnotations
Value	string
Example	`service.beta.kubernetes.io/aws-load-balancer-backend-protocol: http`
Description	The Kubernetes annotations metadata for the load balancer Service
Key	haproxy.serviceAccountName
Value	string
Example	`percona-xtradb-cluster-operator-workload`
Description	The Kubernetes Service Account for the HAProxy Pod
Key	haproxy.runtimeClassName
Value Type	string
Example	`image-rc`
Description	Name of the Kubernetes Runtime Class for the HAProxy Pod
Key	haproxy.sidecars.image
Value Type	string
Example	`busybox`
Description	Image for the *custom sidecar container* for the HAProxy Pod
Key	haproxy.sidecars.command
Value Type	array
Example	`["/bin/sh"]`
Description	Command for the *custom sidecar container* for the HAProxy Pod
Key	haproxy.sidecars.args
Value Type	array
Example	`["-c", "while true; do trap 'exit 0' SIGINT SIGTERM SIGQUIT SIGKILL; done;"]`
Description	Command arguments for the *custom sidecar container* for the HAProxy Pod
Key	haproxy.sidecars.name
Value Type	string
Example	`my-sidecar-1`
Description	Name of the *custom sidecar container* for the HAProxy Pod

29.4 ProxySQL Section

The `proxysql` section in the deploy/cr.yaml file contains configuration options for the ProxySQL daemon.

Key	proxysql.enabled
Value	boolean
Example	`false`
Description	Enables or disables load balancing with ProxySQL Services
Key	proxysql.size
Value	int
Example	`2`
Description	The number of the ProxySQL daemons to provide load balancing. It should be 2 or more unless the `spec.allowUnsafeConfigurations` key is set to true.
Key	proxysql.image
Value	string
Example	`percona/percona-xtradb-cluster-operator:1.9.0-proxysql`
Description	ProxySQL Docker image to use
Key	proxysql.imagePullPolicy
Value	string
Example	`Always`
Description	The policy used to update images
Key	proxysql.imagePullSecrets.name
Value	string
Example	`private-registry-credentials`
Description	The Kubernetes imagePullSecrets for the ProxySQL image
Key	proxysql.configuration
Value	string
Example	
Description	The *custom ProxySQL configuration file* contents
Key	proxysql.annotations
Value	label
Example	`iam.amazonaws.com/role:  role-arn`
Description	The Kubernetes annotations metadata
Key	proxysql.labels
Value	label
Example	`rack:  rack-22`
Description	Labels are key-value pairs attached to objects
Key	proxysql.serviceType
Value	string
Example	`ClusterIP`
Description	Specifies the type of Kubernetes Service to be used

continues on next page

Table 3 – continued from previous page

Key	proxysql.externalTrafficPolicy
Value	string
Example	`Cluster`
Description	Specifies whether Service should route external traffic to cluster-wide or node-local endpoints (it can influence the load balancing effectiveness)
Key	proxysql.resources.requests.memory
Value	string
Example	`1G`
Description	The Kubernetes memory requests for the main ProxySQL container
Key	proxysql.resources.requests.cpu
Value	string
Example	`600m`
Description	Kubernetes CPU requests for the main ProxySQL container
Key	proxysql.resources.limits.memory
Value	string
Example	`1G`
Description	Kubernetes memory limits for the main ProxySQL container
Key	proxysql.resources.limits.cpu
Value	string
Example	`700m`
Description	Kubernetes CPU limits for the main ProxySQL container
Key	proxysql.sidecarResources.requests.memory
Value	string
Example	`1G`
Description	The Kubernetes memory requests for the sidecar ProxySQL containers
Key	proxysql.sidecarResources.requests.cpu
Value	string
Example	`500m`
Description	Kubernetes CPU requests for the sidecar ProxySQL containers
Key	proxysql.sidecarResources.limits.memory
Value	string
Example	`2G`
Description	Kubernetes memory limits for the sidecar ProxySQL containers
Key	proxysql.sidecarResources.limits.cpu
Value	string
Example	`600m`
Description	Kubernetes CPU limits for the sidecar ProxySQL containers
Key	proxysql.envVarsSecret
Value	string
Example	`my-env-var-secrets`
Description	A secret with environment variables, see *Define environment variables* for details

continues on next page

Table 3 – continued from previous page

Key	proxysql.priorityClassName
Value	string
Example	`high-priority`
Description	The Kubernetes Pod Priority class for ProxySQL
Key	proxysql.schedulerName
Value	string
Example	`mycustom-scheduler`
Description	The Kubernetes Scheduler
Key	proxysql.nodeSelector
Value	label
Example	`disktype: ssd`
Description	Kubernetes nodeSelector
Key	proxysql.affinity.topologyKey
Value	string
Example	`kubernetes.io/hostname`
Description	The Operator topology key node anti-affinity constraint
Key	proxysql.affinity.advanced
Value	subdoc
Example	
Description	If available it makes a topologyKey node affinity constraint to be ignored
Key	proxysql.tolerations
Value	subdoc
Example	`node.alpha.kubernetes.io/unreachable`
Description	Kubernetes Pod tolerations
Key	proxysql.volumeSpec.emptyDir
Value	string
Example	`{}`
Description	The Kubernetes emptyDir volume The directory created on a node and accessible to the Percona XtraDB Cluster Pod containers
Key	proxysql.volumeSpec.hostPath.path
Value	string
Example	`/data`
Description	Kubernetes hostPath The volume that mounts a directory from the host node's filesystem into your Pod. The path property is required
Key	proxysql.volumeSpec.hostPath.type
Value	string
Example	`Directory`
Description	The Kubernetes hostPath. An optional property for the hostPath
Key	proxysql.volumeSpec.persistentVolumeClaim.storageClassName
Value	string

continues on next page

Table 3 – continued from previous page

Example	`standard`
Description	Set the Kubernetes storage class to use with the Percona XtraDB Cluster PersistentVolumeClaim
Key	proxysql.volumeSpec.persistentVolumeClaim.accessModes
Value	array
Example	`[ReadWriteOnce]`
Description	The Kubernetes PersistentVolumeClaim access modes for the Percona XtraDB cluster
Key	proxysql.volumeSpec.resources.requests.storage
Value	string
Example	`6Gi`
Description	The Kubernetes PersistentVolumeClaim size for the Percona XtraDB cluster
Key	proxysql.podDisruptionBudget.maxUnavailable
Value	int
Example	`1`
Description	The Kubernetes podDisruptionBudget specifies the number of Pods from the set unavailable after the eviction
Key	proxysql.podDisruptionBudget.minAvailable
Value	int
Example	`0`
Description	The Kubernetes podDisruptionBudget Pods that must be available after an eviction
Key	proxysql.gracePeriod
Value	int
Example	`30`
Description	The Kubernetes grace period when terminating a Pod
Key	proxysql.loadBalancerSourceRanges
Value	string
Example	`10.0.0.0/8`
Description	The range of client IP addresses from which the load balancer should be reachable (if not set, there is no limitations)
Key	proxysql.serviceAnnotations
Value	string
Example	`service.beta.kubernetes.io/aws-load-balancer-backend-protocol: http`
Description	The Kubernetes annotations metadata for the load balancer Service
Key	proxysql.serviceAccountName
Value	string
Example	`percona-xtradb-cluster-operator-workload`
Description	The Kubernetes Service Account for the ProxySQL Pod
Key	proxysql.runtimeClassName
Value Type	string
Example	`image-rc`
Description	Name of the Kubernetes Runtime Class for the ProxySQL Pod

continues on next page

Table 3 – continued from previous page

Key	proxysql.sidecars.image
Value Type	string
Example	`busybox`
Description	Image for the *custom sidecar container* for the ProxySQL Pod
Key	proxysql.sidecars.command
Value Type	array
Example	`["/bin/sh"]`
Description	Command for the *custom sidecar container* for the ProxySQL Pod
Key	proxysql.sidecars.args
Value Type	array
Example	`["-c", "while true; do trap 'exit 0' SIGINT SIGTERM SIGQUIT SIGKILL; done;"]`
Description	Command arguments for the *custom sidecar container* for the ProxySQL Pod
Key	proxysql.sidecars.name
Value Type	string
Example	`my-sidecar-1`
Description	Name of the *custom sidecar container* for the ProxySQL Pod

29.5 Log Collector Section

The `logcollector` section in the deploy/cr.yaml file contains configuration options for Fluent Bit Log Collector.

Key	logcollector.enabled
Value	boolean
Example	`true`
Description	Enables or disables *cluster-level logging with Fluent Bit*
Key	logcollector.image
Value	string
Example	`percona/percona-xtradb-cluster-operator:1.6.0-logcollector`
Description	Log Collector Docker image to use
Key	logcollector.configuration
Value	subdoc
Example	
Description	Additional configuration options (see Fluent Bit official documentation for details)
Key	logcollector.resources.requests.memory
Value	string
Example	`200M`
Description	The Kubernetes memory requests for a Log Collector container
Key	logcollector.resources.requests.cpu
Value	string
Example	`500m`
Description	Kubernetes CPU requests for a Log collector container

29.6 PMM Section

The `pmm` section in the deploy/cr.yaml file contains configuration options for Percona Monitoring and Management.

Key	pmm.enabled
Value	boolean
Example	`false`
Description	Enables or disables monitoring Percona XtraDB cluster with PMM
Key	pmm.image
Value	string
Example	`percona/pmm-client:2.18.0`
Description	PMM client Docker image to use
Key	pmm.serverHost
Value	string
Example	`monitoring-service`
Description	Address of the PMM Server to collect data from the cluster
Key	pmm.serverUser

continues on next page

Table 4 – continued from previous page

Value	string
Example	`admin`
Description	The PMM Serve_User. The PMM Server password should be configured using Secrets
Key	pmm.resources.requests.memory
Value	string
Example	`200M`
Description	The Kubernetes memory requests for a PMM container
Key	pmm.resources.requests.cpu
Value	string
Example	`500m`
Description	Kubernetes CPU requests for a PMM container
Key	pmm.pxcParams
Value Type	string
Example	`--disable-tablestats-limit=2000`
Description	Additional parameters which will be passed to the pmm-admin add mysql command for `pxc` Pods
Key	pmm.proxysqlParams
Value Type	string
Example	`--custom-labels=CUSTOM-LABELS`
Description	Additional parameters which will be passed to the pmm-admin add mysql command for `proxysql` Pods

29.7 Backup Section

The `backup` section in the deploy/cr.yaml file contains the following configuration options for the regular Percona XtraDB Cluster backups.

Key	backup.image
Value	string
Example	`percona/percona-xtradb-cluster-operator:1.9.0-backup`
Description	The Percona XtraDB cluster Docker image to use for the backup
Key	backup.imagePullSecrets.name
Value	string
Example	`private-registry-credentials`
Description	The Kubernetes imagePullSecrets for the specified image
Key	backup.storages.<storage-name>.type
Value	string
Example	`s3`
Description	The cloud storage type used for backups. Only `s3` and `filesystem` types are supported
Key	backup.storages.<storage-name>.s3.credentialsSecret
Value	string
Example	`my-cluster-name-backup-s3`

continues on next page

Table 5 – continued from previous page

Description	The Kubernetes secret for backups. It should contain `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY` keys.
Key	backup.storages.<storage-name>.s3.bucket
Value	string
Example	
Description	The Amazon S3 bucket name for backups
Key	backup.storages.s3.<storage-name>.region
Value	string
Example	`us-east-1`
Description	The AWS region to use. Please note **this option is mandatory** for Amazon and all S3-compatible storages
Key	backup.storages.s3.<storage-name>.endpointUrl
Value	string
Example	
Description	The endpoint URL of the S3-compatible storage to be used (not needed for the original Amazon S3 cloud)
Key	backup.storages.<storage-name>.persistentVolumeClaim.type
Value	string
Example	`filesystem`
Description	The persistent volume claim storage type
Key	backup.storages.<storage-name>.persistentVolumeClaim.storageClassName
Value	string
Example	`standard`
Description	Set the Kubernetes Storage Class to use with the Percona XtraDB Cluster backups PersistentVolumeClaims for the `filesystem` storage type
Key	backup.storages.<storage-name>.volume.persistentVolumeClaim.accessModes
Value	array
Example	`[ReadWriteOne]`
Description	The Kubernetes PersistentVolume access modes
Key	backup.storages.<storage-name>.volume.persistentVolumeClaim.resources.requests.storage
Value	string
Example	`6Gi`
Description	Storage size for the PersistentVolume
Key	backup.storages.<storage-name>.annotations
Value	label
Example	`iam.amazonaws.com/role: role-arn`
Description	The Kubernetes annotations
Key	backup.storages.<storage-name>.labels
Value	label
Example	`rack: rack-22`
Description	Labels are key-value pairs attached to objects

continues on next page

Table 5 – continued from previous page

Key	backup.storages.<storage-name>.resources.requests.memory
Value	string
Example	`1G`
Description	The Kubernetes memory requests for a Percona XtraDB Cluster container
Key	backup.storages.<storage-name>.resources.requests.cpu
Value	string
Example	`600m`
Description	Kubernetes CPU requests for a Percona XtraDB Cluster container
Key	backup.storages.<storage-name>.resources.limits.memory
Value	string
Example	`1G`
Description	Kubernetes memory limits for a Percona XtraDB Cluster container
Key	backup.storages.<storage-name>.nodeSelector
Value	label
Example	`disktype: ssd`
Description	Kubernetes nodeSelector
Key	backup.storages.<storage-name>.affinity.nodeAffinity
Value	subdoc
Example	
Description	The Operator node affinity constraint
Key	backup.storages.<storage-name>.tolerations
Value	subdoc
Example	`backupWorker`
Description	Kubernetes Pod tolerations
Key	backup.storages.<storage-name>.priorityClassName
Value	string
Example	`high-priority`
Description	The Kubernetes Pod priority class
Key	backup.storages.<storage-name>.schedulerName
Value	string
Example	`mycustom-scheduler`
Description	The Kubernetes Scheduler
Key	backup.storages.<storage-name>.containerSecurityContext
Value	subdoc
Example	`privileged: true`
Description	A custom Kubernetes Security Context for a Container to be used instead of the default one
Key	backup.storages.<storage-name>.podSecurityContext
Value	subdoc
Example	`fsGroup: 1001` `supplementalGroups: [1001, 1002, 1003]`

continues on next page

Table 5 – continued from previous page

Description	A custom Kubernetes Security Context for a Pod to be used instead of the default one
Key	backup.schedule.name
Value	string
Example	`sat-night-backup`
Description	The backup name
Key	backup.schedule.schedule
Value	string
Example	`0 0 * * 6`
Description	Scheduled time to make a backup specified in the crontab format
Key	backup.schedule.keep
Value	int
Example	`3`
Description	The amount of most recent backups to store. Older backups are automatically deleted. Set `keep` to zero or completely remove it to disable automatic deletion of backups
Key	backup.schedule.storageName
Value	string
Example	`s3-us-west`
Description	The name of the storage for the backups configured in the `storages` or `fs-pvc` subsection
Key	backup.pitr.enabled
Value	boolean
Example	`false`
Description	Enables or disables *point-in-time-recovery functionality*
Key	backup.pitr.storageName
Value	string
Example	`s3-us-west`
Description	The name of the storage for the backups configured in the `storages` subsection, which will be reused to store binlog for point-in-time-recovery
Key	backup.pitr.timeBetweenUploads
Value	int
Example	`60`
Description	Seconds between running the binlog uploader

CHAPTER

THIRTY

PERCONA CERTIFIED IMAGES

Following table presents Percona's certified docker images to be used with the Percona Distribution for MySQL Operator:

Image	Digest
percona/percona-xtradb-cluster-operator:1.9.0	3783173df8b833028184de686cc98890179a249c7ed5cde8391cdfd751cf990d
percona/percona-xtradb-cluster-operator:1.9.0-haproxy	84f12a19fb179e56862dc1bf8812cad8a328a1dbd05b0f740233618d2c6c4ffc
percona/percona-xtradb-cluster-operator:1.9.0-proxysql	6af34e7261c3afe4cfb28f9c5a32d48d90f2ddd0d0c75e0cc33aba68236f11b5
percona/percona-xtradb-cluster-operator:1.9.0-pxc8.0-backup	8082666a8cdda2b1232885b86b27c92967312449e44971f8f6bed1874ae56b9a
percona/percona-xtradb-cluster-operator:1.9.0-pxc5.7-backup	b75049d3b6b870891a9604fb0e9393c449ed45fbb7064d63bd187470a7535998
percona/percona-xtradb-cluster-operator:1.9.0-logcollector	efb0b84fd5b5231c041579284baaa74a849c1bb0bc7659432925497f88e5bffa
percona/pmm-client:2.18.0	4fd309bf3d9336e9bf853ec5e9b578562c4a69cbc577bf1c2511a3b135922830
percona/percona-xtradb-cluster:8.0.23-14.1	8109f7ca4fc465ba862c08021df12e77b65d384395078e31e270d14b77810d79
percona/percona-xtradb-cluster:8.0.22-13.1	1295af1153c1d02e9d40131eb0945b53f7f371796913e64116bf2caa77dc186d
percona/percona-xtradb-cluster:8.0.21-12.1	d95cf39a58f09759408a00b519fe0d0b19c1b28332ece94349dd5e9cdbda017e
percona/percona-xtradb-cluster:8.0.20-11.2	feda5612db18da824e971891d6084465aa9cdc9918c18001cd95ba30916da78b
percona/percona-xtradb-cluster:8.0.20-11.1	54b1b2f5153b78b05d651034d4603a13e685cbb9b45bfa09a39864fa3f169349
percona/percona-xtradb-cluster:8.0.19-10.1	1058ae8eded735ebdf664807aad7187942fc9a1170b3fd0369574cb61206b63a
percona/percona-xtradb-cluster:5.7.34-31.51	f8d51d7932b9bb1a5a896c7ae440256230eb69b55798ff37397aabfd58b80ccb
percona/percona-xtradb-cluster:5.7.33-31.49	f0a4bbb0ec5adff2a2d3e88194b3dcac479266ca29da028f0dfb22f55449ac17
percona/percona-xtradb-cluster:5.7.32-31.47	7b095019ad354c336494248d6080685022e2ed46e3b53fc103b25cd12c95952b
percona/percona-xtradb-cluster:5.7.31-31.45.2	0decf85c7c7afacc438f5fe355dc8320ea7ffc7018ca2cb6bda3ac0c526ae172
percona/percona-xtradb-cluster:5.7.31-31.45	3852cef43cc0c6aa791463ba6279e59dcdac3a4fb1a5616c745c1b3c68041dc2
percona/percona-xtradb-cluster:5.7.30-31.43	b03a060e9261b37288a2153c78f86dcfc53367c36e1bcdcae046dd2d0b0721af
percona/percona-xtradb-cluster:5.7.29-31.43	85fb479de073770280ae601cf3ec22dc5c8cca4c8b0dc893b09503767338e6f9
percona/percona-xtradb-cluster:5.7.28-31.41.2	fccd6525aaeedb5e436e9534e2a63aebcf743c043526dd05dba8519ebddc8b30

CHAPTER

THIRTYONE

PERCONA DISTRIBUTION FOR MYSQL OPERATOR API DOCUMENTATION

Percona Distribution for MySQL Operator provides an aggregation-layer extension for the Kubernetes API. Please refer to the official Kubernetes API documentation on the API access and usage details. The following subsections describe the Percona XtraDB Cluster API provided by the Operator.

31.1 Prerequisites

1. Create the namespace name you will use, if not exist:

```
kubectl create namespace my-namespace-name
```

Trying to create an already-existing namespace will show you a self-explanatory error message. Also, you can use the `defalut` namespace.

Note: In this document `default` namespace is used in all examples. Substitute `default` with your namespace name if you use a different one.

2. Prepare

```
# set correct API address
KUBE_CLUSTER=$(kubectl config view --minify -o jsonpath='{.clusters[0].name}')
API_SERVER=$(kubectl config view -o jsonpath="{.clusters[?(@.name==\"$KUBE_CLUSTER\
↪")].cluster.server}" | sed -e 's#https://##')
```

(continues on next page)

(continued from previous page)

```
# create service account and get token
kubectl apply -f deploy/crd.yaml -f deploy/rbac.yaml -n default
KUBE_TOKEN=$(kubectl get secret $(kubectl get serviceaccount percona-xtradb-cluster-
↪operator -o jsonpath='{.secrets[0].name}' -n default) -o jsonpath='{.data.token}'␣
↪-n default | base64 --decode )
```

31.2 Create new Percona XtraDB Cluster

Description:

```
The command to create a new Percona XtraDB Cluster with all its resources
```

Kubectl Command:

```
kubectl apply -f percona-xtradb-cluster-operator/deploy/cr.yaml
```

URL:

```
https://$API_SERVER/apis/pxc.percona.com/v1-9-0/namespaces/default/perconaxtradbclusters
```

Authentication:

```
Authorization: Bearer $KUBE_TOKEN
```

cURL Request:

```
curl -k -v -XPOST "https://$API_SERVER/apis/pxc.percona.com/v1-9-0/namespaces/default/
↪perconaxtradbclusters" \
            -H "Content-Type: application/json" \
            -H "Accept: application/json" \
            -H "Authorization: Bearer $KUBE_TOKEN" \
            -d "@cluster.json"
```

Request Body (cluster.json):

JSON:

```
{
   "apiVersion":"pxc.percona.com/v1-5-0",
   "kind":"PerconaXtraDBCluster",
   "metadata":{
      "name":"cluster1",
      "finalizers":[
         "delete-pxc-pods-in-order"
      ]
   },
   "spec":{
      "secretsName":"my-cluster-secrets",
      "vaultSecretName":"keyring-secret-vault",
      "sslSecretName":"my-cluster-ssl",
```

(continues on next page)

(continued from previous page)

```
      "sslInternalSecretName":"my-cluster-ssl-internal",
      "allowUnsafeConfigurations":true,
      "pxc":{
         "size":3,
         "image":"percona/percona-xtradb-cluster:8.0.19-10.1",
         "resources":{
            "requests":null
         },
         "affinity":{
            "antiAffinityTopologyKey":"none"
         },
         "podDisruptionBudget":{
            "maxUnavailable":1
         },
         "volumeSpec":{
            "persistentVolumeClaim":{
               "resources":{
                  "requests":{
                     "storage":"6Gi"
                  }
               }
            }
         },
         "gracePeriod":600
      },
      "proxysql":{
         "enabled":true,
         "size":3,
         "image":"percona/percona-xtradb-cluster-operator:1.5.0-proxysql",
         "resources":{
            "requests":null
         },
         "affinity":{
            "antiAffinityTopologyKey":"none"
         },
         "volumeSpec":{
            "persistentVolumeClaim":{
               "resources":{
                  "requests":{
                     "storage":"2Gi"
                  }
               }
            }
         },
         "podDisruptionBudget":{
            "maxUnavailable":1
         },
         "gracePeriod":30
      },
      "pmm":{
         "enabled":false,
         "image":"percona/percona-xtradb-cluster-operator:1.5.0-pmm",
```

(continues on next page)

(continued from previous page)

```
            "serverHost":"monitoring-service",
            "serverUser":"pmm"
         },
         "backup":{
            "image":"percona/percona-xtradb-cluster-operator:1.5.0-pxc8.0-backup",
            "serviceAccountName":"percona-xtradb-cluster-operator",
            "storages":{
               "s3-us-west":{
                  "type":"s3",
                  "s3":{
                     "bucket":"S3-BACKUP-BUCKET-NAME-HERE",
                     "credentialsSecret":"my-cluster-name-backup-s3",
                     "region":"us-west-2"
                  }
               },
               "fs-pvc":{
                  "type":"filesystem",
                  "volume":{
                     "persistentVolumeClaim":{
                        "accessModes":[
                           "ReadWriteOnce"
                        ],
                        "resources":{
                           "requests":{
                              "storage":"6Gi"
                           }
                        }
                     }
                  }
               }
            },
            "schedule":[
               {
                  "name":"sat-night-backup",
                  "schedule":"0 0 * * 6",
                  "keep":3,
                  "storageName":"s3-us-west"
               },
               {
                  "name":"daily-backup",
                  "schedule":"0 0 * * *",
                  "keep":5,
                  "storageName":"fs-pvc"
               }
            ]
         }
      }
}
```

Inputs:

Metadata:

1. Name (String, min-length: 1) : `contains name of cluster`

2. Finalizers (list of string, Default: ["delete-pxc-pods-in-order"]) `contains steps to do when deleting the cluster`

Spec:

1. secretsName (String, min-length: 1) : `contains name of secret to create for the cluster`
2. vaultSecretName (String, min-length: 1) : `contains name of vault secret to create for the cluster`
3. sslInternalSecretName (String, min-length: 1) : `contains name of ssl secret to create for the cluster`
4. allowUnsafeConfigurations (Boolean, Default: false) : `allow unsafe configurations to run`

pxc:

1. Size (Int , min-value: 1, default, 3) : `number of Percona XtraDB Cluster nodes to create`
2. Image (String, min-length: 1) : `contains image name to use for Percona XtraDB Cluster nodes`
3. volumeSpec : storage (SizeString, default: "6Gi") : `contains the size for the storage volume of Percona XtraDB Cluster nodes`
4. gracePeriod (Int, default: 600, min-value: 0) : `contains the time to wait for Percona XtraDB Cluster node to shutdown in milliseconds`

proxysql:

1. Enabled (Boolean, default: true) : `enabled or disables proxysql`

pmm:

1. serverHost (String, min-length: 1) : `serivce name for monitoring`
2. serverUser (String, min-length: 1) : `name of pmm user`
3. image (String, min-length: 1) : `name of pmm image`

backup:

1. Storages (Object) : `contains the storage destinations to save the backups in`
2. schedule:
   1. name (String, min-length: 1) : `name of backup job`
   2. schedule (String, Cron format: `"* * * * *"`) : `contains cron schedule format for when to run cron jobs`
   3. keep (Int, min-value = 1) : `number of backups to keep`
   4. storageName (String, min-length: 1) : `name of storage object to use`

Response:

JSON

```
{
   "apiVersion":"pxc.percona.com/v1-5-0",
   "kind":"PerconaXtraDBCluster",
   "metadata":{
      "creationTimestamp":"2020-05-27T22:23:58Z",
      "finalizers":[
```

(continues on next page)

(continued from previous page)


```
         "delete-pxc-pods-in-order"
      ],
      "generation":1,
      "managedFields":[
         {
            "apiVersion":"pxc.percona.com/v1-5-0",
            "fieldsType":"FieldsV1",
            "fieldsV1":{
               "f:metadata":{
                  "f:finalizers":{

                  }
               },
               "f:spec":{
                  ".":{

                  },
                  "f:allowUnsafeConfigurations":{

                  },
                  "f:backup":{
                     ".":{

                     },
                     "f:image":{

                     },
                     "f:schedule":{

                     },
                     "f:serviceAccountName":{

                     },
                     "f:storages":{
                        ".":{

                        },
                        "f:fs-pvc":{
                           ".":{

                           },
                           "f:type":{

                           },
                           "f:volume":{
                              ".":{

                              },
                              "f:persistentVolumeClaim":{
                                 ".":{

                                 },
```


(continues on next page)

(continued from previous page)


```
                                    "f:accessModes":{

                                    },
                                    "f:resources":{
                                       ".":{

                                       },
                                       "f:requests":{
                                          ".":{

                                          },
                                          "f:storage":{

                                          }
                                       }
                                    }
                                 }
                              }
                           },
                           "f:s3-us-west":{
                              ".":{

                              },
                              "f:s3":{
                                 ".":{

                                 },
                                 "f:bucket":{

                                 },
                                 "f:credentialsSecret":{

                                 },
                                 "f:region":{

                                 }
                              },
                              "f:type":{

                              }
                           }
                        }
                     },
                     "f:pmm":{
                        ".":{

                        },
                        "f:enabled":{

                        },
                        "f:image":{
```


(continues on next page)

(continued from previous page)


```
                    },
                    "f:serverHost":{

                    },
                    "f:serverUser":{

                    }
                 },
                 "f:proxysql":{
                    ".":{

                    },
                    "f:affinity":{
                       ".":{

                       },
                       "f:antiAffinityTopologyKey":{

                       }
                    },
                    "f:enabled":{

                    },
                    "f:gracePeriod":{

                    },
                    "f:image":{

                    },
                    "f:podDisruptionBudget":{
                       ".":{

                       },
                       "f:maxUnavailable":{

                       }
                    },
                    "f:resources":{
                       ".":{

                       },
                       "f:requests":{

                       }
                    },
                    "f:size":{

                    },
                    "f:volumeSpec":{
                       ".":{

                       },
```


(continues on next page)

(continued from previous page)


```
                  "f:persistentVolumeClaim":{
                     ".":{

                     },
                     "f:resources":{
                        ".":{

                        },
                        "f:requests":{
                           ".":{

                           },
                           "f:storage":{

                           }
                        }
                     }
                  }
               }
            },
            "f:pxc":{
               ".":{

               },
               "f:affinity":{
                  ".":{

                  },
                  "f:antiAffinityTopologyKey":{

                  }
               },
               "f:gracePeriod":{

               },
               "f:image":{

               },
               "f:podDisruptionBudget":{
                  ".":{

                  },
                  "f:maxUnavailable":{

                  }
               },
               "f:resources":{
                  ".":{

                  },
                  "f:requests":{
```


(continues on next page)

(continued from previous page)


```
                    }
                },
                "f:size":{

                },
                "f:volumeSpec":{
                    ".":{

                    },
                    "f:persistentVolumeClaim":{
                        ".":{

                        },
                        "f:resources":{
                            ".":{

                            },
                            "f:requests":{
                                ".":{

                                },
                                "f:storage":{

                                }
                            }
                        }
                    }
                }
            },
            "f:secretsName":{

            },
            "f:sslInternalSecretName":{

            },
            "f:sslSecretName":{

            },
            "f:vaultSecretName":{

            }
        }
    },
    "manager":"kubectl",
    "operation":"Update",
    "time":"2020-05-27T22:23:58Z"
  }
],
"name":"cluster1",
"namespace":"default",
"resourceVersion":"8694",
"selfLink":"/apis/pxc.percona.com/v1-5-0/namespaces/default/perconaxtradbclusters/
↪cluster1",
```


(continues on next page)

(continued from previous page)


```
      "uid":"e9115e2a-49df-4ebf-9dab-fa5a550208d3"
   },
   "spec":{
      "allowUnsafeConfigurations":false,
      "backup":{
         "image":"percona/percona-xtradb-cluster-operator:1.5.0-pxc8.0-backup",
         "schedule":[
            {
               "keep":3,
               "name":"sat-night-backup",
               "schedule":"0 0 * * 6",
               "storageName":"s3-us-west"
            },
            {
               "keep":5,
               "name":"daily-backup",
               "schedule":"0 0 * * *",
               "storageName":"fs-pvc"
            }
         ],
         "serviceAccountName":"percona-xtradb-cluster-operator",
         "storages":{
            "fs-pvc":{
               "type":"filesystem",
               "volume":{
                  "persistentVolumeClaim":{
                     "accessModes":[
                        "ReadWriteOnce"
                     ],
                     "resources":{
                        "requests":{
                           "storage":"6Gi"
                        }
                     }
                  }
               }
            },
            "s3-us-west":{
               "s3":{
                  "bucket":"S3-BACKUP-BUCKET-NAME-HERE",
                  "credentialsSecret":"my-cluster-name-backup-s3",
                  "region":"us-west-2"
               },
               "type":"s3"
            }
         }
      },
      "pmm":{
         "enabled":false,
         "image":"percona/percona-xtradb-cluster-operator:1.5.0-pmm",
         "serverHost":"monitoring-service",
         "serverUser":"pmm"
```


(continues on next page)

(continued from previous page)

```
      },
      "proxysql":{
         "affinity":{
            "antiAffinityTopologyKey":"none"
         },
         "enabled":true,
         "gracePeriod":30,
         "image":"percona/percona-xtradb-cluster-operator:1.5.0-proxysql",
         "podDisruptionBudget":{
            "maxUnavailable":1
         },
         "resources":{
            "requests":null
         },
         "size":3,
         "volumeSpec":{
            "persistentVolumeClaim":{
               "resources":{
                  "requests":{
                     "storage":"2Gi"
                  }
               }
            }
         }
      },
      "pxc":{
         "affinity":{
            "antiAffinityTopologyKey":"none"
         },
         "gracePeriod":600,
         "image":"percona/percona-xtradb-cluster:8.0.19-10.1",
         "podDisruptionBudget":{
            "maxUnavailable":1
         },
         "resources":{
            "requests":null
         },
         "size":3,
         "volumeSpec":{
            "persistentVolumeClaim":{
               "resources":{
                  "requests":{
                     "storage":"6Gi"
                  }
               }
            }
         }
      },
      "secretsName":"my-cluster-secrets",
      "sslInternalSecretName":"my-cluster-ssl-internal",
      "sslSecretName":"my-cluster-ssl",
      "vaultSecretName":"keyring-secret-vault"
```

(continues on next page)

(continued from previous page)

```
    }
}
```

31.3 List Percona XtraDB Clusters

Description:

```
Lists all Percona XtraDB Clusters that exist in your kubernetes cluster
```

Kubectl Command:

```
kubectl get pxc
```

URL:

```
https://$API_SERVER/apis/pxc.percona.com/v1/namespaces/default/perconaxtradbclusters?
↪limit=500
```

Authentication:

```
Authorization: Bearer $KUBE_TOKEN
```

cURL Request:

```
curl -k -v -XGET "https://$API_SERVER/apis/pxc.percona.com/v1/namespaces/default/
↪perconaxtradbclusters?limit=500" \
            -H "Accept: application/json;as=Table;v=v1;g=meta.k8s.io,application/json;
↪as=Table;v=v1beta1;g=meta.k8s.io,application/json" \
            -H "Authorization: Bearer $KUBE_TOKEN"
```

Request Body:

```
None
```

Response:

JSON:


```
{
   "kind":"Table",
   "apiVersion":"meta.k8s.io/v1",
   "metadata":{
      "selfLink":"/apis/pxc.percona.com/v1/namespaces/default/perconaxtradbclusters",
      "resourceVersion":"10528"
   },
   "columnDefinitions":[
      {
         "name":"Name",
         "type":"string",
         "format":"name",
         "description":"Name must be unique within a namespace. Is required when␣
↪creating resources, although some resources may allow a client to request the␣
↪generation of an appropriate name automatically. Name is primarily intended for␣
↪creation idempotence and configuration definition. Cannot be updated. More info: http:/
↪/kubernetes.io/docs/user-guide/identifiers#names",
```


(continues on next page)

(continued from previous page)


```
         "priority":0
      },
      {
         "name":"Endpoint",
         "type":"string",
         "format":"",
         "description":"Custom resource definition column (in JSONPath format): .status.
↪host",
         "priority":0
      },
      {
         "name":"Status",
         "type":"string",
         "format":"",
         "description":"Custom resource definition column (in JSONPath format): .status.
↪state",
         "priority":0
      },
      {
         "name":"PXC",
         "type":"string",
         "format":"",
         "description":"Ready pxc nodes",
         "priority":0
      },
      {
         "name":"proxysql",
         "type":"string",
         "format":"",
         "description":"Ready pxc nodes",
         "priority":0
      },
      {
         "name":"Age",
         "type":"date",
         "format":"",
         "description":"Custom resource definition column (in JSONPath format): .
↪metadata.creationTimestamp",
         "priority":0
      }
   ],
   "rows":[
      {
         "cells":[
            "cluster1",
            "cluster1-proxysql.default",
            "ready",
            "3",
            "3",
            "8m37s"
         ],
         "object":{
```


(continues on next page)

(continued from previous page)


```
            "kind":"PartialObjectMetadata",
            "apiVersion":"meta.k8s.io/v1",
            "metadata":{
               "name":"cluster1",
               "namespace":"default",
               "selfLink":"/apis/pxc.percona.com/v1/namespaces/default/
↪perconaxtradbclusters/cluster1",
               "uid":"e9115e2a-49df-4ebf-9dab-fa5a550208d3",
               "resourceVersion":"10517",
               "generation":1,
               "creationTimestamp":"2020-05-27T22:23:58Z",
               "finalizers":[
                  "delete-pxc-pods-in-order"
               ],
               "managedFields":[
                  {
                     "manager":"kubectl",
                     "operation":"Update",
                     "apiVersion":"pxc.percona.com/v1-5-0",
                     "time":"2020-05-27T22:23:58Z",
                     "fieldsType":"FieldsV1",
                     "fieldsV1":{
                        "f:metadata":{
                           "f:finalizers":{

                           }
                        },
                        "f:spec":{
                           ".":{

                           },
                           "f:allowUnsafeConfigurations":{

                           },
                           "f:backup":{
                              ".":{

                              },
                              "f:image":{

                              },
                              "f:schedule":{

                              },
                              "f:serviceAccountName":{

                              },
                              "f:storages":{
                                 ".":{

                                 },
                                 "f:fs-pvc":{
```


(continues on next page)

(continued from previous page)


```
   ".":{

   },
   "f:type":{

   },
   "f:volume":{
      ".":{

      },
      "f:persistentVolumeClaim":{
         ".":{

         },
         "f:accessModes":{

         },
         "f:resources":{
            ".":{

            },
            "f:requests":{
               ".":{

               },
               "f:storage":{

               }
            }
         }
      }
   }
},
"f:s3-us-west":{
   ".":{

   },
   "f:s3":{
      ".":{

      },
      "f:bucket":{

      },
      "f:credentialsSecret":{

      },
      "f:region":{

      }
   },
   "f:type":{
```


(continues on next page)

(continued from previous page)


```
                                    }
                                 }
                              }
                           },
                           "f:pmm":{
                              ".":{

                              },
                              "f:image":{

                              },
                              "f:serverHost":{

                              },
                              "f:serverUser":{

                              }
                           },
                           "f:proxysql":{
                              ".":{

                              },
                              "f:affinity":{
                                 ".":{

                                 },
                                 "f:antiAffinityTopologyKey":{

                                 }
                              },
                              "f:enabled":{

                              },
                              "f:gracePeriod":{

                              },
                              "f:image":{

                              },
                              "f:podDisruptionBudget":{
                                 ".":{

                                 },
                                 "f:maxUnavailable":{

                                 }
                              },
                              "f:resources":{

                              },
                              "f:size":{
```


(continues on next page)

(continued from previous page)


```
},
"f:volumeSpec":{
   ".":{

   },
   "f:persistentVolumeClaim":{
      ".":{

      },
      "f:resources":{
         ".":{

         },
         "f:requests":{
            ".":{

            },
            "f:storage":{

            }
         }
      }
   }
}
},
"f:pxc":{
   ".":{

   },
   "f:affinity":{
      ".":{

      },
      "f:antiAffinityTopologyKey":{

      }
   },
   "f:gracePeriod":{

   },
   "f:image":{

   },
   "f:podDisruptionBudget":{
      ".":{

      },
      "f:maxUnavailable":{

      }
   },
```


(continues on next page)

(continued from previous page)


```
                        "f:resources":{

                        },
                        "f:size":{

                        },
                        "f:volumeSpec":{
                           ".":{

                           },
                           "f:persistentVolumeClaim":{
                              ".":{

                              },
                              "f:resources":{
                                 ".":{

                                 },
                                 "f:requests":{
                                    ".":{

                                    },
                                    "f:storage":{

                                    }
                                 }
                              }
                           }
                        }
                     },
                     "f:secretsName":{

                     },
                     "f:sslInternalSecretName":{

                     },
                     "f:sslSecretName":{

                     },
                     "f:vaultSecretName":{

                     }
                  }
               }
            },
            {
               "manager":"percona-xtradb-cluster-operator",
               "operation":"Update",
               "apiVersion":"pxc.percona.com/v1",
               "time":"2020-05-27T22:32:31Z",
               "fieldsType":"FieldsV1",
               "fieldsV1":{
```


(continues on next page)

(continued from previous page)


```
                    "f:spec":{
                       "f:backup":{
                          "f:storages":{
                             "f:fs-pvc":{
                                "f:podSecurityContext":{
                                   ".":{

                                   },
                                   "f:fsGroup":{

                                   },
                                   "f:supplementalGroups":{

                                   }
                                },
                                "f:s3":{
                                   ".":{

                                   },
                                   "f:bucket":{

                                   },
                                   "f:credentialsSecret":{

                                   }
                                }
                             },
                             "f:s3-us-west":{
                                "f:podSecurityContext":{
                                   ".":{

                                   },
                                   "f:fsGroup":{

                                   },
                                   "f:supplementalGroups":{

                                   }
                                }
                             }
                          }
                       },
                       "f:pmm":{
                          "f:resources":{

                          }
                       },
                       "f:proxysql":{
                          "f:podSecurityContext":{
                             ".":{

                             },
```


(continues on next page)

(continued from previous page)


```
         "f:fsGroup":{

         },
         "f:supplementalGroups":{

         }
      },
      "f:sslInternalSecretName":{

      },
      "f:sslSecretName":{

      },
      "f:volumeSpec":{
         "f:persistentVolumeClaim":{
            "f:accessModes":{

            }
         }
      }
   },
   "f:pxc":{
      "f:podSecurityContext":{
         ".":{

         },
         "f:fsGroup":{

         },
         "f:supplementalGroups":{

         }
      },
      "f:sslInternalSecretName":{

      },
      "f:sslSecretName":{

      },
      "f:vaultSecretName":{

      },
      "f:volumeSpec":{
         "f:persistentVolumeClaim":{
            "f:accessModes":{

            }
         }
      }
   }
},
"f:status":{
```


(continues on next page)

(continued from previous page)


```
                           ".":{

                           },
                           "f:conditions":{

                           },
                           "f:host":{

                           },
                           "f:observedGeneration":{

                           },
                           "f:proxysql":{
                              ".":{

                              },
                              "f:ready":{

                              },
                              "f:size":{

                              },
                              "f:status":{

                              }
                           },
                           "f:pxc":{
                              ".":{

                              },
                              "f:ready":{

                              },
                              "f:size":{

                              },
                              "f:status":{

                              }
                           },
                           "f:state":{

                           }
                        }
                     }
                  }
               ]
            }
         }
      }
   ]
}
```

31.4 Get status of Percona XtraDB Cluster

Description:

```
Gets all information about the specified Percona XtraDB Cluster
```

Kubectl Command:

```
kubectl get pxc/cluster1 -o json
```

URL:

```
https://$API_SERVER/apis/pxc.percona.com/v1/namespaces/default/perconaxtradbclusters/
↪cluster1
```

Authentication:

```
Authorization: Bearer $KUBE_TOKEN
```

cURL Request:

```
curl -k -v -XGET "https://$API_SERVER/apis/pxc.percona.com/v1/namespaces/default/
↪perconaxtradbclusters/cluster1" \
            -H "Accept: application/json" \
            -H "Authorization: Bearer $KUBE_TOKEN"
```

Request Body:

```
None
```

Response:

JSON:


```
{
   "apiVersion":"pxc.percona.com/v1",
   "kind":"PerconaXtraDBCluster",
   "metadata":{
      "annotations":{
         "kubectl.kubernetes.io/last-applied-configuration":"{\"apiVersion\":\"pxc.
↪percona.com/v1\",\"kind\":\"PerconaXtraDBCluster\",\"metadata\":{\"annotations\":{},\
↪"creationTimestamp\":\"2020-05-27T22:23:58Z\",\"finalizers\":[\"delete-pxc-pods-in-
↪order\"],\"generation\":1,\"managedFields\":[{\"apiVersion\":\"pxc.percona.com/v1-5-0\
↪",\"fieldsType\":\"FieldsV1\",\"fieldsV1\":{\"f:metadata\":{\"f:finalizers\":{}},\
↪"f:spec\":{\".\":{},\"f:allowUnsafeConfigurations\":{},\"f:backup\":{\".\":{},\
↪"f:image\":{},\"f:schedule\":{},\"f:serviceAccountName\":{},\"f:storages\":{\".\":{},\
↪"f:fs-pvc\":{\".\":{},\"f:type\":{},\"f:volume\":{\".\":{},\"f:persistentVolumeClaim\":
↪{\".\":{},\"f:accessModes\":{},\"f:resources\":{\".\":{},\"f:requests\":{\".\":{},\
↪"f:storage\":{}}}}}},\"f:s3-us-west\":{\".\":{},\"f:s3\":{\".\":{},\"f:bucket\":{},\
↪"f:credentialsSecret\":{},\"f:region\":{}},\"f:type\":{}}}},\"f:pmm\":{\".\":{},\
↪"f:image\":{},\"f:serverHost\":{},\"f:serverUser\":{}},\"f:proxysql\":{\".\":{},\
↪"f:affinity\":{\".\":{},\"f:antiAffinityTopologyKey\":{}},\"f:enabled\":{},\
↪"f:gracePeriod\":{},\"f:image\":{},\"f:podDisruptionBudget\":{\".\":{},\
↪"f:maxUnavailable\":{}},\"f:resources\":{},\"f:size\":{},\"f:volumeSpec\":{\".\":{},\
↪"f:persistentVolumeClaim\":{\".\":{},\"f:resources\":{\".\":{},\"f:requests\":{\".\":{}
↪,\"f:storage\":{}}}}}}},\"f:pxc\":{\".\":{},\"f:affinity\":{\".\":{},\
↪"f:antiAffinityTopologyKey\":{}},\"f:gracePeriod\":{},\"f:image\":{},\
↪"f:podDisruptionBudget\":{\".\":{},\"f:maxUnavailable\":{}},\"f:resources\":{},\
↪"f:size\":{},\"f:volumeSpec\":{\".\":{},\"f:persistentVolumeClaim\":{\".\":{},\
↪"f:resources\":{\".\":{},\"f:requests\":{\".\":{},\"f:storage\":{}}}}}},\
↪"f:secretsName\":{},\"f:sslInternalSecretName\":{},\"f:sslSecretName\":{},\
↪"f:vaultSecretName\":{}}},\"manager\":\"kubectl\",\"operation\":\"Update\",\"time\":\
```


(continues on next page)

(continued from previous page)


```
      },
      "creationTimestamp":"2020-05-27T22:23:58Z",
      "finalizers":[
         "delete-pxc-pods-in-order"
      ],
      "generation":6,
      "managedFields":[
         {
            "apiVersion":"pxc.percona.com/v1-5-0",
            "fieldsType":"FieldsV1",
            "fieldsV1":{
               "f:metadata":{
                  "f:finalizers":{

                  }
               },
               "f:spec":{
                  ".":{

                  },
                  "f:allowUnsafeConfigurations":{

                  },
                  "f:backup":{
                     ".":{

                     },
                     "f:schedule":{

                     },
                     "f:serviceAccountName":{

                     },
                     "f:storages":{
                        ".":{

                        },
                        "f:fs-pvc":{
                           ".":{

                           },
                           "f:type":{

                           },
                           "f:volume":{
                              ".":{

                              },
                              "f:persistentVolumeClaim":{
                                 ".":{

                                 },
```


(continues on next page)

(continued from previous page)


```
                              "f:accessModes":{

                              },
                              "f:resources":{
                                 ".":{

                                 },
                                 "f:requests":{
                                    ".":{

                                    },
                                    "f:storage":{

                                    }
                                 }
                              }
                           }
                        }
                     },
                     "f:s3-us-west":{
                        ".":{

                        },
                        "f:s3":{
                           ".":{

                           },
                           "f:bucket":{

                           },
                           "f:credentialsSecret":{

                           },
                           "f:region":{

                           }
                        },
                        "f:type":{

                        }
                     }
                  }
               },
               "f:pmm":{
                  ".":{

                  },
                  "f:image":{

                  },
                  "f:serverHost":{
```


(continues on next page)

(continued from previous page)


```
                },
                "f:serverUser":{

                }
            },
            "f:proxysql":{
                ".":{

                },
                "f:affinity":{
                    ".":{

                    },
                    "f:antiAffinityTopologyKey":{

                    }
                },
                "f:enabled":{

                },
                "f:gracePeriod":{

                },
                "f:image":{

                },
                "f:podDisruptionBudget":{
                    ".":{

                    },
                    "f:maxUnavailable":{

                    }
                },
                "f:resources":{

                },
                "f:volumeSpec":{
                    ".":{

                    },
                    "f:persistentVolumeClaim":{
                        ".":{

                        },
                        "f:resources":{
                            ".":{

                            },
                            "f:requests":{
                                ".":{
```


(continues on next page)

(continued from previous page)


```
                              },
                              "f:storage":{

                              }
                           }
                        }
                     }
                  }
               },
               "f:pxc":{
                  ".":{

                  },
                  "f:affinity":{
                     ".":{

                     },
                     "f:antiAffinityTopologyKey":{

                     }
                  },
                  "f:gracePeriod":{

                  },
                  "f:podDisruptionBudget":{
                     ".":{

                     },
                     "f:maxUnavailable":{

                     }
                  },
                  "f:resources":{

                  },
                  "f:volumeSpec":{
                     ".":{

                     },
                     "f:persistentVolumeClaim":{
                        ".":{

                        },
                        "f:resources":{
                           ".":{

                           },
                           "f:requests":{
                              ".":{

                              },
                              "f:storage":{
```


(continues on next page)

(continued from previous page)


```
                        }
                     }
                  }
               }
            }
         },
         "f:secretsName":{

         },
         "f:sslInternalSecretName":{

         },
         "f:sslSecretName":{

         },
         "f:vaultSecretName":{

         }
      }
   },
   "manager":"kubectl",
   "operation":"Update",
   "time":"2020-05-27T22:23:58Z"
},
{
   "apiVersion":"pxc.percona.com/v1",
   "fieldsType":"FieldsV1",
   "fieldsV1":{
      "f:metadata":{
         "f:annotations":{
            ".":{

            },
            "f:kubectl.kubernetes.io/last-applied-configuration":{

            }
         }
      },
      "f:spec":{
         "f:backup":{
            "f:image":{

            }
         },
         "f:proxysql":{
            "f:size":{

            }
         },
         "f:pxc":{
            "f:image":{
```


(continues on next page)

(continued from previous page)


```
                    },
                    "f:size":{

                    }
                 }
              }
           },
           "manager":"kubectl",
           "operation":"Update",
           "time":"2020-05-27T23:38:49Z"
        },
        {
           "apiVersion":"pxc.percona.com/v1",
           "fieldsType":"FieldsV1",
           "fieldsV1":{
              "f:spec":{
                 "f:backup":{
                    "f:storages":{
                       "f:fs-pvc":{
                          "f:podSecurityContext":{
                             ".":{

                             },
                             "f:fsGroup":{

                             },
                             "f:supplementalGroups":{

                             }
                          },
                          "f:s3":{
                             ".":{

                             },
                             "f:bucket":{

                             },
                             "f:credentialsSecret":{

                             }
                          }
                       },
                       "f:s3-us-west":{
                          "f:podSecurityContext":{
                             ".":{

                             },
                             "f:fsGroup":{

                             },
                             "f:supplementalGroups":{
```


(continues on next page)

(continued from previous page)


```
                    }
                 }
              }
           }
        },
        "f:pmm":{
           "f:resources":{

           }
        },
        "f:proxysql":{
           "f:podSecurityContext":{
              ".":{

              },
              "f:fsGroup":{

              },
              "f:supplementalGroups":{

              }
           },
           "f:sslInternalSecretName":{

           },
           "f:sslSecretName":{

           },
           "f:volumeSpec":{
              "f:persistentVolumeClaim":{
                 "f:accessModes":{

                 }
              }
           }
        },
        "f:pxc":{
           "f:podSecurityContext":{
              ".":{

              },
              "f:fsGroup":{

              },
              "f:supplementalGroups":{

              }
           },
           "f:sslInternalSecretName":{

           },
```


(continues on next page)

(continued from previous page)


```
                "f:sslSecretName":{

                },
                "f:vaultSecretName":{

                },
                "f:volumeSpec":{
                   "f:persistentVolumeClaim":{
                      "f:accessModes":{

                      }
                   }
                }
             }
          },
          "f:status":{
             ".":{

             },
             "f:conditions":{

             },
             "f:host":{

             },
             "f:message":{

             },
             "f:observedGeneration":{

             },
             "f:proxysql":{
                ".":{

                },
                "f:ready":{

                },
                "f:size":{

                },
                "f:status":{

                }
             },
             "f:pxc":{
                ".":{

                },
                "f:message":{

                },
```


(continues on next page)

(continued from previous page)


```
                     "f:ready":{

                     },
                     "f:size":{

                     },
                     "f:status":{

                     }
                  },
                  "f:state":{

                  }
               }
            },
            "manager":"percona-xtradb-cluster-operator",
            "operation":"Update",
            "time":"2020-05-28T10:42:00Z"
         }
      ],
      "name":"cluster1",
      "namespace":"default",
      "resourceVersion":"35660",
      "selfLink":"/apis/pxc.percona.com/v1/namespaces/default/perconaxtradbclusters/
↪cluster1",
      "uid":"e9115e2a-49df-4ebf-9dab-fa5a550208d3"
   },
   "spec":{
      "allowUnsafeConfigurations":true,
      "backup":{
         "image":"percona/percona-xtradb-cluster-operator:1.5.0-pxc8.0-debug-backup",
         "schedule":[
            {
               "keep":3,
               "name":"sat-night-backup",
               "schedule":"0 0 * * 6",
               "storageName":"s3-us-west"
            },
            {
               "keep":5,
               "name":"daily-backup",
               "schedule":"0 0 * * *",
               "storageName":"fs-pvc"
            }
         ],
         "serviceAccountName":"percona-xtradb-cluster-operator",
         "storages":{
            "fs-pvc":{
               "type":"filesystem",
               "volume":{
                  "persistentVolumeClaim":{
                     "accessModes":[
```


(continues on next page)

(continued from previous page)

```
                      "ReadWriteOnce"
                   ],
                   "resources":{
                      "requests":{
                         "storage":"6Gi"
                      }
                   }
                }
             }
          },
          "s3-us-west":{
             "s3":{
                "bucket":"S3-BACKUP-BUCKET-NAME-HERE",
                "credentialsSecret":"my-cluster-name-backup-s3",
                "region":"us-west-2"
             },
             "type":"s3"
          }
       }
    },
    "pmm":{
       "enabled":false,
       "image":"percona/percona-xtradb-cluster-operator:1.5.0-pmm",
       "serverHost":"monitoring-service",
       "serverUser":"pmm"
    },
    "proxysql":{
       "affinity":{
          "antiAffinityTopologyKey":"none"
       },
       "enabled":true,
       "gracePeriod":30,
       "image":"percona/percona-xtradb-cluster-operator:1.5.0-proxysql",
       "podDisruptionBudget":{
          "maxUnavailable":1
       },
       "resources":{

       },
       "size":3,
       "volumeSpec":{
          "persistentVolumeClaim":{
             "resources":{
                "requests":{
                   "storage":"2Gi"
                }
             }
          }
       }
    },
    "pxc":{
       "affinity":{
```

(continues on next page)

(continued from previous page)


```
            "antiAffinityTopologyKey":"none"
         },
         "gracePeriod":600,
         "image":"percona/percona-xtradb-cluster-operator:1.5.0-pxc8.0-debug",
         "podDisruptionBudget":{
            "maxUnavailable":1
         },
         "resources":{

         },
         "size":3,
         "volumeSpec":{
            "persistentVolumeClaim":{
               "resources":{
                  "requests":{
                     "storage":"6Gi"
                  }
               }
            }
         }
      },
      "secretsName":"my-cluster-secrets",
      "sslInternalSecretName":"my-cluster-ssl-internal",
      "sslSecretName":"my-cluster-ssl",
      "vaultSecretName":"keyring-secret-vault"
   },
   "status":{
      "conditions":[
         {
            "lastTransitionTime":"2020-05-27T22:25:43Z",
            "status":"True",
            "type":"Ready"
         },
         {
            "lastTransitionTime":"2020-05-27T23:06:48Z",
            "status":"True",
            "type":"Initializing"
         },
         {
            "lastTransitionTime":"2020-05-27T23:08:58Z",
            "message":"ProxySQL upgrade error: context deadline exceeded",
            "reason":"ErrorReconcile",
            "status":"True",
            "type":"Error"
         },
         {
            "lastTransitionTime":"2020-05-27T23:08:59Z",
            "status":"True",
            "type":"Initializing"
         },
         {
            "lastTransitionTime":"2020-05-27T23:29:59Z",
```


(continues on next page)

(continued from previous page)


```
        "status":"True",
        "type":"Ready"
    },
    {
        "lastTransitionTime":"2020-05-27T23:30:04Z",
        "status":"True",
        "type":"Initializing"
    },
    {
        "lastTransitionTime":"2020-05-27T23:35:27Z",
        "status":"True",
        "type":"Ready"
    },
    {
        "lastTransitionTime":"2020-05-27T23:35:42Z",
        "status":"True",
        "type":"Initializing"
    },
    {
        "lastTransitionTime":"2020-05-27T23:47:00Z",
        "status":"True",
        "type":"Ready"
    },
    {
        "lastTransitionTime":"2020-05-27T23:47:05Z",
        "status":"True",
        "type":"Initializing"
    },
    {
        "lastTransitionTime":"2020-05-28T09:58:25Z",
        "status":"True",
        "type":"Ready"
    },
    {
        "lastTransitionTime":"2020-05-28T09:58:31Z",
        "status":"True",
        "type":"Initializing"
    },
    {
        "lastTransitionTime":"2020-05-28T10:03:54Z",
        "status":"True",
        "type":"Ready"
    },
    {
        "lastTransitionTime":"2020-05-28T10:04:14Z",
        "status":"True",
        "type":"Initializing"
    },
    {
        "lastTransitionTime":"2020-05-28T10:15:28Z",
        "status":"True",
        "type":"Ready"
```


(continues on next page)

(continued from previous page)


```
        },
        {
           "lastTransitionTime":"2020-05-28T10:15:38Z",
           "status":"True",
           "type":"Initializing"
        },
        {
           "lastTransitionTime":"2020-05-28T10:26:56Z",
           "status":"True",
           "type":"Ready"
        },
        {
           "lastTransitionTime":"2020-05-28T10:27:01Z",
           "status":"True",
           "type":"Initializing"
        },
        {
           "lastTransitionTime":"2020-05-28T10:38:28Z",
           "status":"True",
           "type":"Ready"
        },
        {
           "lastTransitionTime":"2020-05-28T10:38:33Z",
           "status":"True",
           "type":"Initializing"
        }
     ],
     "host":"cluster1-proxysql.default",
     "message":[
        "PXC: pxc: back-off 5m0s restarting failed container=pxc pod=cluster1-pxc-1_
↪default(5b9b16e6-d0f8-4c97-a2d0-294feb9d014b); pxc: back-off 5m0s restarting failed␣
↪container=pxc pod=cluster1-pxc-2_default(b8ebedd7-42f0-440b-aa5e-509d28926a5e); pxc:␣
↪back-off 5m0s restarting failed container=pxc pod=cluster1-pxc-4_default(2dce12f2-9ebc-
↪419c-a92a-9cec68912004); "
     ],
     "observedGeneration":6,
     "proxysql":{
        "ready":3,
        "size":3,
        "status":"ready"
     },
     "pxc":{
        "message":"pxc: back-off 5m0s restarting failed container=pxc pod=cluster1-pxc-
↪1_default(5b9b16e6-d0f8-4c97-a2d0-294feb9d014b); pxc: back-off 5m0s restarting failed␣
↪container=pxc pod=cluster1-pxc-2_default(b8ebedd7-42f0-440b-aa5e-509d28926a5e); pxc:␣
↪back-off 5m0s restarting failed container=pxc pod=cluster1-pxc-4_default(2dce12f2-9ebc-
↪419c-a92a-9cec68912004); ",
        "ready":2,
        "size":3,
        "status":"initializing"
     },
     "state":"initializing"
```


(continues on next page)

(continued from previous page)

```
    }
}
```

31.5 Scale up/down Percona XtraDB Cluster

Description:

```
Increase or decrease the size of the Percona XtraDB Cluster nodes to fit the
current high availability needs
```

Kubectl Command:

```
kubectl patch pxc cluster1 --type=merge --patch '{
"spec": {"pxc":{ "size": "5" }
}}'
```

URL:

```
https://$API_SERVER/apis/pxc.percona.com/v1/namespaces/default/perconaxtradbclusters/
↪cluster1
```

Authentication:

```
Authorization: Bearer $KUBE_TOKEN
```

cURL Request:

```
curl -k -v -XPATCH "https://$API_SERVER/apis/pxc.percona.com/v1/namespaces/default/
↪perconaxtradbclusters/cluster1" \
            -H "Authorization: Bearer $KUBE_TOKEN" \
            -H "Content-Type: application/merge-patch+json"
            -H "Accept: application/json" \
            -d '{
                  "spec": {"pxc":{ "size": "5" }
                  }}'
```

Request Body:

JSON:

```
{
"spec": {"pxc":{ "size": "5" }
}}
```

Input:

spec:

pxc

1. size (Int or String, Defaults: 3): `Specifiy the size of the Percona XtraDB Cluster to scale up or down to`

Response:

JSON:


```
{
   "apiVersion":"pxc.percona.com/v1",
   "kind":"PerconaXtraDBCluster",
   "metadata":{
      "annotations":{
         "kubectl.kubernetes.io/last-applied-configuration":"{\"apiVersion\":\"pxc.
↪percona.com/v1-5-0\",\"kind\":\"PerconaXtraDBCluster\",\"metadata\":{\"annotations\":{}
↪,\"finalizers\":[\"delete-pxc-pods-in-order\"],\"name\":\"cluster1\",\"namespace\":\
↪"default\"},\"spec\":{\"allowUnsafeConfigurations\":true,\"backup\":{\"image\":\
↪"percona/percona-xtradb-cluster-operator:1.5.0-pxc8.0-backup\",\"schedule\":[{\"keep\
↪":3,\"name\":\"sat-night-backup\",\"schedule\":\"0 0 * * 6\",\"storageName\":\"s3-us-
↪west\"},{\"keep\":5,\"name\":\"daily-backup\",\"schedule\":\"0 0 * * *\",\"storageName\
↪":\"fs-pvc\"}],\"serviceAccountName\":\"percona-xtradb-cluster-operator\",\"storages\":
↪{\"fs-pvc\":{\"type\":\"filesystem\",\"volume\":{\"persistentVolumeClaim\":{\
↪"accessModes\":[\"ReadWriteOnce\"],\"resources\":{\"requests\":{\"storage\":\"6Gi\"}}}}
↪},\"s3-us-west\":{\"s3\":{\"bucket\":\"S3-BACKUP-BUCKET-NAME-HERE\",\
↪"credentialsSecret\":\"my-cluster-name-backup-s3\",\"region\":\"us-west-2\"},\"type\":\
↪"s3\"}}},\"pmm\":{\"enabled\":false,\"image\":\"percona/percona-xtradb-cluster-
↪operator:1.5.0-pmm\",\"serverHost\":\"monitoring-service\",\"serverUser\":\"pmm\"},\
↪"proxysql\":{\"affinity\":{\"antiAffinityTopologyKey\":\"none\"},\"enabled\":true,\
↪"gracePeriod\":30,\"image\":\"percona/percona-xtradb-cluster-operator:1.5.0-proxysql\",
↪\"podDisruptionBudget\":{\"maxUnavailable\":1},\"resources\":{\"requests\":null},\
↪"size\":3,\"volumeSpec\":{\"persistentVolumeClaim\":{\"resources\":{\"requests\":{\
↪"storage\":\"2Gi\"}}}}},\"pxc\":{\"affinity\":{\"antiAffinityTopologyKey\":\"none\"},\
↪"gracePeriod\":600,\"image\":\"percona/percona-xtradb-cluster:8.0.19-10.1\",\
↪"podDisruptionBudget\":{\"maxUnavailable\":1},\"resources\":{\"requests\":null},\"size\
↪":3,\"volumeSpec\":{\"persistentVolumeClaim\":{\"resources\":{\"requests\":{\"storage\
↪":\"6Gi\"}}}}},\"secretsName\":\"my-cluster-secrets\",\"sslInternalSecretName\":\"my-
↪cluster-ssl-internal\",\"sslSecretName\":\"my-cluster-ssl\",\"updateStrategy\":\
↪"RollingUpdate\",\"vaultSecretName\":\"keyring-secret-vault\"}}\n"
      },
      "creationTimestamp":"2020-06-01T16:50:05Z",
      "finalizers":[
         "delete-pxc-pods-in-order"
      ],
      "generation":4,
      "managedFields":[
         {
            "apiVersion":"pxc.percona.com/v1-5-0",
            "fieldsType":"FieldsV1",
            "fieldsV1":{
               "f:metadata":{
                  "f:annotations":{
                     ".":{

                     },
                     "f:kubectl.kubernetes.io/last-applied-configuration":{

                     }
                  },
```


(continues on next page)

(continued from previous page)


```
                   "f:finalizers":{

                   }
                },
                "f:spec":{
                   ".":{

                   },
                   "f:allowUnsafeConfigurations":{

                   },
                   "f:backup":{
                      ".":{

                      },
                      "f:image":{

                      },
                      "f:schedule":{

                      },
                      "f:serviceAccountName":{

                      },
                      "f:storages":{
                         ".":{

                         },
                         "f:fs-pvc":{
                            ".":{

                            },
                            "f:type":{

                            },
                            "f:volume":{
                               ".":{

                               },
                               "f:persistentVolumeClaim":{
                                  ".":{

                                  },
                                  "f:accessModes":{

                                  },
                                  "f:resources":{
                                     ".":{

                                     },
                                     "f:requests":{
                                        ".":{
```


(continues on next page)

(continued from previous page)


```
                                },
                                "f:storage":{

                                }
                            }
                        }
                    }
                }
            },
            "f:s3-us-west":{
                ".":{

                },
                "f:s3":{
                    ".":{

                    },
                    "f:bucket":{

                    },
                    "f:credentialsSecret":{

                    },
                    "f:region":{

                    }
                },
                "f:type":{

                }
            }
        }
    },
    "f:pmm":{
        ".":{

        },
        "f:image":{

        },
        "f:serverHost":{

        },
        "f:serverUser":{

        }
    },
    "f:proxysql":{
        ".":{

        },
```


(continues on next page)

(continued from previous page)

```
                    "f:affinity":{
                       ".":{

                       },
                       "f:antiAffinityTopologyKey":{

                       }
                    },
                    "f:enabled":{

                    },
                    "f:gracePeriod":{

                    },
                    "f:image":{

                    },
                    "f:podDisruptionBudget":{
                       ".":{

                       },
                       "f:maxUnavailable":{

                       }
                    },
                    "f:resources":{

                    },
                    "f:size":{

                    },
                    "f:volumeSpec":{
                       ".":{

                       },
                       "f:persistentVolumeClaim":{
                          ".":{

                          },
                          "f:resources":{
                             ".":{

                             },
                             "f:requests":{
                                ".":{

                                },
                                "f:storage":{

                                }
                             }
                          }
```

(continues on next page)

(continued from previous page)


```
                    }
                 }
              },
              "f:pxc":{
                 ".":{

                 },
                 "f:affinity":{
                    ".":{

                    },
                    "f:antiAffinityTopologyKey":{

                    }
                 },
                 "f:gracePeriod":{

                 },
                 "f:podDisruptionBudget":{
                    ".":{

                    },
                    "f:maxUnavailable":{

                    }
                 },
                 "f:resources":{

                 },
                 "f:volumeSpec":{
                    ".":{

                    },
                    "f:persistentVolumeClaim":{
                       ".":{

                       },
                       "f:resources":{
                          ".":{

                          },
                          "f:requests":{
                             ".":{

                             },
                             "f:storage":{

                             }
                          }
                       }
                    }
                 }
```


(continues on next page)

(continued from previous page)


```
                },
                "f:secretsName":{

                },
                "f:sslInternalSecretName":{

                },
                "f:sslSecretName":{

                },
                "f:updateStrategy":{

                },
                "f:vaultSecretName":{

                }
             }
          },
          "manager":"kubectl",
          "operation":"Update",
          "time":"2020-06-01T16:52:30Z"
       },
       {
          "apiVersion":"pxc.percona.com/v1",
          "fieldsType":"FieldsV1",
          "fieldsV1":{
             "f:spec":{
                "f:backup":{
                   "f:storages":{
                      "f:fs-pvc":{
                         "f:podSecurityContext":{
                            ".":{

                            },
                            "f:fsGroup":{

                            },
                            "f:supplementalGroups":{

                            }
                         },
                         "f:s3":{
                            ".":{

                            },
                            "f:bucket":{

                            },
                            "f:credentialsSecret":{

                            }
                         }
```


(continues on next page)

(continued from previous page)


```
                 },
                 "f:s3-us-west":{
                    "f:podSecurityContext":{
                       ".":{

                       },
                       "f:fsGroup":{

                       },
                       "f:supplementalGroups":{

                       }
                    }
                 }
              }
           },
           "f:pmm":{
              "f:resources":{

              }
           },
           "f:proxysql":{
              "f:podSecurityContext":{
                 ".":{

                 },
                 "f:fsGroup":{

                 },
                 "f:supplementalGroups":{

                 }
              },
              "f:sslInternalSecretName":{

              },
              "f:sslSecretName":{

              },
              "f:volumeSpec":{
                 "f:persistentVolumeClaim":{
                    "f:accessModes":{

                    }
                 }
              }
           },
           "f:pxc":{
              "f:podSecurityContext":{
                 ".":{

                 },
```


(continues on next page)

(continued from previous page)


```
                "f:fsGroup":{

                },
                "f:supplementalGroups":{

                }
             },
             "f:sslInternalSecretName":{

             },
             "f:sslSecretName":{

             },
             "f:vaultSecretName":{

             },
             "f:volumeSpec":{
                "f:persistentVolumeClaim":{
                   "f:accessModes":{

                   }
                }
             }
          }
       },
       "f:status":{
          ".":{

          },
          "f:conditions":{

          },
          "f:host":{

          },
          "f:observedGeneration":{

          },
          "f:proxysql":{
             ".":{

             },
             "f:ready":{

             },
             "f:size":{

             },
             "f:status":{

             }
          },
```


(continues on next page)

(continued from previous page)


```
                  "f:pxc":{
                     ".":{

                     },
                     "f:ready":{

                     },
                     "f:size":{

                     },
                     "f:status":{

                     }
                  },
                  "f:state":{

                  }
               }
            },
            "manager":"percona-xtradb-cluster-operator",
            "operation":"Update",
            "time":"2020-06-03T15:32:11Z"
         },
         {
            "apiVersion":"pxc.percona.com/v1",
            "fieldsType":"FieldsV1",
            "fieldsV1":{
               "f:spec":{
                  "f:pxc":{
                     "f:image":{

                     },
                     "f:size":{

                     }
                  }
               }
            },
            "manager":"kubectl",
            "operation":"Update",
            "time":"2020-06-03T15:32:14Z"
         }
      ],
      "name":"cluster1",
      "namespace":"default",
      "resourceVersion":"129605",
      "selfLink":"/apis/pxc.percona.com/v1/namespaces/default/perconaxtradbclusters/
↪cluster1",
      "uid":"15e5e7d6-10b2-46cf-85d0-d3fdea3412ca"
   },
   "spec":{
      "allowUnsafeConfigurations":true,
```


(continues on next page)

(continued from previous page)

```
      "backup":{
         "image":"percona/percona-xtradb-cluster-operator:1.5.0-pxc8.0-backup",
         "schedule":[
            {
               "keep":3,
               "name":"sat-night-backup",
               "schedule":"0 0 * * 6",
               "storageName":"s3-us-west"
            },
            {
               "keep":5,
               "name":"daily-backup",
               "schedule":"0 0 * * *",
               "storageName":"fs-pvc"
            }
         ],
         "serviceAccountName":"percona-xtradb-cluster-operator",
         "storages":{
            "fs-pvc":{
               "type":"filesystem",
               "volume":{
                  "persistentVolumeClaim":{
                     "accessModes":[
                        "ReadWriteOnce"
                     ],
                     "resources":{
                        "requests":{
                           "storage":"6Gi"
                        }
                     }
                  }
               }
            },
            "s3-us-west":{
               "s3":{
                  "bucket":"S3-BACKUP-BUCKET-NAME-HERE",
                  "credentialsSecret":"my-cluster-name-backup-s3",
                  "region":"us-west-2"
               },
               "type":"s3"
            }
         }
      },
      "pmm":{
         "enabled":false,
         "image":"percona/percona-xtradb-cluster-operator:1.5.0-pmm",
         "serverHost":"monitoring-service",
         "serverUser":"pmm"
      },
      "proxysql":{
         "affinity":{
            "antiAffinityTopologyKey":"none"
```

(continues on next page)

(continued from previous page)

```
        },
        "enabled":true,
        "gracePeriod":30,
        "image":"percona/percona-xtradb-cluster-operator:1.5.0-proxysql",
        "podDisruptionBudget":{
           "maxUnavailable":1
        },
        "resources":{
           "requests":null
        },
        "size":3,
        "volumeSpec":{
           "persistentVolumeClaim":{
              "resources":{
                 "requests":{
                    "storage":"2Gi"
                 }
              }
           }
        }
     },
     "pxc":{
        "affinity":{
           "antiAffinityTopologyKey":"none"
        },
        "gracePeriod":600,
        "image":"percona/percona-xtradb-cluster:5.7.30-31.43",
        "podDisruptionBudget":{
           "maxUnavailable":1
        },
        "resources":{
           "requests":null
        },
        "size":"5",
        "volumeSpec":{
           "persistentVolumeClaim":{
              "resources":{
                 "requests":{
                    "storage":"6Gi"
                 }
              }
           }
        }
     },
     "secretsName":"my-cluster-secrets",
     "sslInternalSecretName":"my-cluster-ssl-internal",
     "sslSecretName":"my-cluster-ssl",
     "updateStrategy":"RollingUpdate",
     "vaultSecretName":"keyring-secret-vault"
  },
  "status":{
     "conditions":[
```

(continues on next page)

(continued from previous page)


```
        {
           "lastTransitionTime":"2020-06-01T16:50:37Z",
           "message":"create newStatefulSetNode: StatefulSet.apps \"cluster1-pxc\" is␣
↪invalid: spec.updateStrategy: Invalid value: apps.StatefulSetUpdateStrategy{Type:\
↪"SmartUpdate\", RollingUpdate:(*apps.RollingUpdateStatefulSetStrategy)(nil)}: must be
↪'RollingUpdate' or 'OnDelete'",
           "reason":"ErrorReconcile",
           "status":"True",
           "type":"Error"
        },
        {
           "lastTransitionTime":"2020-06-01T16:52:31Z",
           "status":"True",
           "type":"Initializing"
        },
        {
           "lastTransitionTime":"2020-06-01T16:55:59Z",
           "status":"True",
           "type":"Ready"
        },
        {
           "lastTransitionTime":"2020-06-01T17:19:15Z",
           "status":"True",
           "type":"Initializing"
        }
     ],
     "host":"cluster1-proxysql.default",
     "observedGeneration":3,
     "proxysql":{
        "ready":3,
        "size":3,
        "status":"ready"
     },
     "pxc":{
        "ready":1,
        "size":3,
        "status":"initializing"
     },
     "state":"initializing"
  }
}
```

31.6 Update Percona XtraDB Cluster image

Description:

```
Change the image of Percona XtraDB Cluster containers inside the cluster
```

Kubectl Command:

```
kubectl patch pxc cluster1 --type=merge --patch '{
"spec": {"pxc":{ "image": "percona/percona-xtradb-cluster:5.7.30-31.43" }
}}'
```

URL:

```
https://$API_SERVER/apis/pxc.percona.com/v1/namespaces/default/perconaxtradbclusters/
↪cluster1
```

Authentication:

```
Authorization: Bearer $KUBE_TOKEN
```

cURL Request:

```
curl -k -v -XPATCH "https://$API_SERVER/apis/pxc.percona.com/v1/namespaces/default/
↪perconaxtradbclusters/cluster1" \
            -H "Authorization: Bearer $KUBE_TOKEN" \
            -H "Accept: application/json" \
            -H "Content-Type: application/merge-patch+json"
            -d '{
              "spec": {"pxc":{ "image": "percona/percona-xtradb-cluster:5.7.30-31.43" }
              }}'
```

Request Body:

JSON:

```
{
"spec": {"pxc":{ "image": "percona/percona-xtradb-cluster:5.7.30-31.43" }
}}
```

Input:

spec:

pxc:

1. image (String, min-length: 1) : `name of the image to update for Percona XtraDB Cluster`

Response:

JSON:

```
{
   "apiVersion":"pxc.percona.com/v1",
   "kind":"PerconaXtraDBCluster",
   "metadata":{
```

(continues on next page)

(continued from previous page)

```
      "annotations":{
         "kubectl.kubernetes.io/last-applied-configuration":"{\"apiVersion\":\"pxc.
↪percona.com/v1-5-0\",\"kind\":\"PerconaXtraDBCluster\",\"metadata\":{\"annotations\":{}
↪,\"finalizers\":[\"delete-pxc-pods-in-order\"],\"name\":\"cluster1\",\"namespace\":\
↪"default\"},\"spec\":{\"allowUnsafeConfigurations\":true,\"backup\":{\"image\":\
↪"percona/percona-xtradb-cluster-operator:1.5.0-pxc8.0-backup\",\"schedule\":[{\"keep\
↪":3,\"name\":\"sat-night-backup\",\"schedule\":\"0 0 * * 6\",\"storageName\":\"s3-us-
↪west\"},{\"keep\":5,\"name\":\"daily-backup\",\"schedule\":\"0 0 * * *\",\"storageName\
↪":\"fs-pvc\"}],\"serviceAccountName\":\"percona-xtradb-cluster-operator\",\"storages\":
↪{\"fs-pvc\":{\"type\":\"filesystem\",\"volume\":{\"persistentVolumeClaim\":{\
↪"accessModes\":[\"ReadWriteOnce\"],\"resources\":{\"requests\":{\"storage\":\"6Gi\"}}}}
↪},\"s3-us-west\":{\"s3\":{\"bucket\":\"S3-BACKUP-BUCKET-NAME-HERE\",\
↪"credentialsSecret\":\"my-cluster-name-backup-s3\",\"region\":\"us-west-2\"},\"type\":\
↪"s3\"}}},\"pmm\":{\"enabled\":false,\"image\":\"percona/percona-xtradb-cluster-
↪operator:1.5.0-pmm\",\"serverHost\":\"monitoring-service\",\"serverUser\":\"pmm\"},\
↪"proxysql\":{\"affinity\":{\"antiAffinityTopologyKey\":\"none\"},\"enabled\":true,\
↪"gracePeriod\":30,\"image\":\"percona/percona-xtradb-cluster-operator:1.5.0-proxysql\",
↪\"podDisruptionBudget\":{\"maxUnavailable\":1},\"resources\":{\"requests\":null},\
↪"size\":3,\"volumeSpec\":{\"persistentVolumeClaim\":{\"resources\":{\"requests\":{\
↪"storage\":\"2Gi\"}}}}},\"pxc\":{\"affinity\":{\"antiAffinityTopologyKey\":\"none\"},\
↪"gracePeriod\":600,\"image\":\"percona/percona-xtradb-cluster:8.0.19-10.1\",\
↪"podDisruptionBudget\":{\"maxUnavailable\":1},\"resources\":{\"requests\":null},\"size\
↪":3,\"volumeSpec\":{\"persistentVolumeClaim\":{\"resources\":{\"requests\":{\"storage\
↪":\"6Gi\"}}}}},\"secretsName\":\"my-cluster-secrets\",\"sslInternalSecretName\":\"my-
↪cluster-ssl-internal\",\"sslSecretName\":\"my-cluster-ssl\",\"updateStrategy\":\
↪"RollingUpdate\",\"vaultSecretName\":\"keyring-secret-vault\"}}\n"
      },
      "creationTimestamp":"2020-06-01T16:50:05Z",
      "finalizers":[
         "delete-pxc-pods-in-order"
      ],
      "generation":3,
      "managedFields":[
         {
            "apiVersion":"pxc.percona.com/v1-5-0",
            "fieldsType":"FieldsV1",
            "fieldsV1":{
               "f:metadata":{
                  "f:annotations":{
                     ".":{

                     },
                     "f:kubectl.kubernetes.io/last-applied-configuration":{

                     }
                  },
                  "f:finalizers":{

                  }
               },
               "f:spec":{
                  ".":{
```

(continues on next page)

(continued from previous page)


```
                  },
                  "f:allowUnsafeConfigurations":{

                  },
                  "f:backup":{
                     ".":{

                     },
                     "f:image":{

                     },
                     "f:schedule":{

                     },
                     "f:serviceAccountName":{

                     },
                     "f:storages":{
                        ".":{

                        },
                        "f:fs-pvc":{
                           ".":{

                           },
                           "f:type":{

                           },
                           "f:volume":{
                              ".":{

                              },
                              "f:persistentVolumeClaim":{
                                 ".":{

                                 },
                                 "f:accessModes":{

                                 },
                                 "f:resources":{
                                    ".":{

                                    },
                                    "f:requests":{
                                       ".":{

                                       },
                                       "f:storage":{

                                       }
                                    }
```


(continues on next page)

(continued from previous page)


```
                    }
                 }
              }
           },
           "f:s3-us-west":{
              ".":{

              },
              "f:s3":{
                 ".":{

                 },
                 "f:bucket":{

                 },
                 "f:credentialsSecret":{

                 },
                 "f:region":{

                 }
              },
              "f:type":{

              }
           }
        }
     },
     "f:pmm":{
        ".":{

        },
        "f:image":{

        },
        "f:serverHost":{

        },
        "f:serverUser":{

        }
     },
     "f:proxysql":{
        ".":{

        },
        "f:affinity":{
           ".":{

           },
           "f:antiAffinityTopologyKey":{
```


(continues on next page)

(continued from previous page)

```
            }
         },
         "f:enabled":{

         },
         "f:gracePeriod":{

         },
         "f:image":{

         },
         "f:podDisruptionBudget":{
            ".":{

            },
            "f:maxUnavailable":{

            }
         },
         "f:resources":{

         },
         "f:size":{

         },
         "f:volumeSpec":{
            ".":{

            },
            "f:persistentVolumeClaim":{
               ".":{

               },
               "f:resources":{
                  ".":{

                  },
                  "f:requests":{
                     ".":{

                     },
                     "f:storage":{

                     }
                  }
               }
            }
         }
      },
      "f:pxc":{
         ".":{
```

(continues on next page)

(continued from previous page)


```
                    },
                    "f:affinity":{
                       ".":{

                       },
                       "f:antiAffinityTopologyKey":{

                       }
                    },
                    "f:gracePeriod":{

                    },
                    "f:podDisruptionBudget":{
                       ".":{

                       },
                       "f:maxUnavailable":{

                       }
                    },
                    "f:resources":{

                    },
                    "f:size":{

                    },
                    "f:volumeSpec":{
                       ".":{

                       },
                       "f:persistentVolumeClaim":{
                          ".":{

                          },
                          "f:resources":{
                             ".":{

                             },
                             "f:requests":{
                                ".":{

                                },
                                "f:storage":{

                                }
                             }
                          }
                       }
                    }
                 },
                 "f:secretsName":{
```


(continues on next page)

(continued from previous page)


```
            },
            "f:sslInternalSecretName":{

            },
            "f:sslSecretName":{

            },
            "f:updateStrategy":{

            },
            "f:vaultSecretName":{

            }
         }
      },
      "manager":"kubectl",
      "operation":"Update",
      "time":"2020-06-01T16:52:30Z"
   },
   {
      "apiVersion":"pxc.percona.com/v1",
      "fieldsType":"FieldsV1",
      "fieldsV1":{
         "f:spec":{
            "f:pxc":{
               "f:image":{

               }
            }
         }
      },
      "manager":"kubectl",
      "operation":"Update",
      "time":"2020-06-01T17:18:58Z"
   },
   {
      "apiVersion":"pxc.percona.com/v1",
      "fieldsType":"FieldsV1",
      "fieldsV1":{
         "f:spec":{
            "f:backup":{
               "f:storages":{
                  "f:fs-pvc":{
                     "f:podSecurityContext":{
                        ".":{

                        },
                        "f:fsGroup":{

                        },
                        "f:supplementalGroups":{
```


(continues on next page)

(continued from previous page)


```
                              }
                           },
                           "f:s3":{
                              ".":{

                              },
                              "f:bucket":{

                              },
                              "f:credentialsSecret":{

                              }
                           }
                        },
                        "f:s3-us-west":{
                           "f:podSecurityContext":{
                              ".":{

                              },
                              "f:fsGroup":{

                              },
                              "f:supplementalGroups":{

                              }
                           }
                        }
                     }
                  },
                  "f:pmm":{
                     "f:resources":{

                     }
                  },
                  "f:proxysql":{
                     "f:podSecurityContext":{
                        ".":{

                        },
                        "f:fsGroup":{

                        },
                        "f:supplementalGroups":{

                        }
                     },
                     "f:sslInternalSecretName":{

                     },
                     "f:sslSecretName":{

                     },
```


(continues on next page)

(continued from previous page)


```
                  "f:volumeSpec":{
                     "f:persistentVolumeClaim":{
                        "f:accessModes":{

                        }
                     }
                  }
               },
               "f:pxc":{
                  "f:podSecurityContext":{
                     ".":{

                     },
                     "f:fsGroup":{

                     },
                     "f:supplementalGroups":{

                     }
                  },
                  "f:sslInternalSecretName":{

                  },
                  "f:sslSecretName":{

                  },
                  "f:vaultSecretName":{

                  },
                  "f:volumeSpec":{
                     "f:persistentVolumeClaim":{
                        "f:accessModes":{

                        }
                     }
                  }
               }
            },
            "f:status":{
               ".":{

               },
               "f:conditions":{

               },
               "f:host":{

               },
               "f:message":{

               },
               "f:observedGeneration":{
```


(continues on next page)

(continued from previous page)


```
                  },
                  "f:proxysql":{
                     ".":{

                     },
                     "f:ready":{

                     },
                     "f:size":{

                     },
                     "f:status":{

                     }
                  },
                  "f:pxc":{
                     ".":{

                     },
                     "f:message":{

                     },
                     "f:ready":{

                     },
                     "f:size":{

                     },
                     "f:status":{

                     }
                  },
                  "f:state":{

                  }
               }
            },
            "manager":"percona-xtradb-cluster-operator",
            "operation":"Update",
            "time":"2020-06-01T17:21:36Z"
         }
      ],
      "name":"cluster1",
      "namespace":"default",
      "resourceVersion":"41149",
      "selfLink":"/apis/pxc.percona.com/v1/namespaces/default/perconaxtradbclusters/
↪cluster1",
      "uid":"15e5e7d6-10b2-46cf-85d0-d3fdea3412ca"
   },
   "spec":{
      "allowUnsafeConfigurations":true,
```


(continues on next page)

(continued from previous page)

```
      "backup":{
         "image":"percona/percona-xtradb-cluster-operator:1.5.0-pxc8.0-backup",
         "schedule":[
            {
               "keep":3,
               "name":"sat-night-backup",
               "schedule":"0 0 * * 6",
               "storageName":"s3-us-west"
            },
            {
               "keep":5,
               "name":"daily-backup",
               "schedule":"0 0 * * *",
               "storageName":"fs-pvc"
            }
         ],
         "serviceAccountName":"percona-xtradb-cluster-operator",
         "storages":{
            "fs-pvc":{
               "type":"filesystem",
               "volume":{
                  "persistentVolumeClaim":{
                     "accessModes":[
                        "ReadWriteOnce"
                     ],
                     "resources":{
                        "requests":{
                           "storage":"6Gi"
                        }
                     }
                  }
               }
            },
            "s3-us-west":{
               "s3":{
                  "bucket":"S3-BACKUP-BUCKET-NAME-HERE",
                  "credentialsSecret":"my-cluster-name-backup-s3",
                  "region":"us-west-2"
               },
               "type":"s3"
            }
         }
      },
      "pmm":{
         "enabled":false,
         "image":"percona/percona-xtradb-cluster-operator:1.5.0-pmm",
         "serverHost":"monitoring-service",
         "serverUser":"pmm"
      },
      "proxysql":{
         "affinity":{
            "antiAffinityTopologyKey":"none"
```

(continues on next page)

(continued from previous page)


```
        },
        "enabled":true,
        "gracePeriod":30,
        "image":"percona/percona-xtradb-cluster-operator:1.5.0-proxysql",
        "podDisruptionBudget":{
           "maxUnavailable":1
        },
        "resources":{
           "requests":null
        },
        "size":3,
        "volumeSpec":{
           "persistentVolumeClaim":{
              "resources":{
                 "requests":{
                    "storage":"2Gi"
                 }
              }
           }
        }
     },
     "pxc":{
        "affinity":{
           "antiAffinityTopologyKey":"none"
        },
        "gracePeriod":600,
        "image":"percona/percona-xtradb-cluster:5.7.30-31.43",
        "podDisruptionBudget":{
           "maxUnavailable":1
        },
        "resources":{
           "requests":null
        },
        "size":3,
        "volumeSpec":{
           "persistentVolumeClaim":{
              "resources":{
                 "requests":{
                    "storage":"6Gi"
                 }
              }
           }
        }
     },
     "secretsName":"my-cluster-secrets",
     "sslInternalSecretName":"my-cluster-ssl-internal",
     "sslSecretName":"my-cluster-ssl",
     "updateStrategy":"RollingUpdate",
     "vaultSecretName":"keyring-secret-vault"
  },
  "status":{
     "conditions":[
```


(continues on next page)

(continued from previous page)

```
        {
           "lastTransitionTime":"2020-06-01T16:50:37Z",
           "message":"create newStatefulSetNode: StatefulSet.apps \"cluster1-pxc\" is␣
↪invalid: spec.updateStrategy: Invalid value: apps.StatefulSetUpdateStrategy{Type:\
↪"SmartUpdate\", RollingUpdate:(*apps.RollingUpdateStatefulSetStrategy)(nil)}: must be
↪'RollingUpdate' or 'OnDelete'",
           "reason":"ErrorReconcile",
           "status":"True",
           "type":"Error"
        },
        {
           "lastTransitionTime":"2020-06-01T16:52:31Z",
           "status":"True",
           "type":"Initializing"
        },
        {
           "lastTransitionTime":"2020-06-01T16:55:59Z",
           "status":"True",
           "type":"Ready"
        },
        {
           "lastTransitionTime":"2020-06-01T17:19:15Z",
           "status":"True",
           "type":"Initializing"
        }
     ],
     "host":"cluster1-proxysql.default",
     "message":[
        "PXC: pxc: back-off 40s restarting failed container=pxc pod=cluster1-pxc-2_
↪default(87cdf1a8-0fb3-4bc0-b50d-f66a0a73c087); "
     ],
     "observedGeneration":3,
     "proxysql":{
        "ready":3,
        "size":3,
        "status":"ready"
     },
     "pxc":{
        "message":"pxc: back-off 40s restarting failed container=pxc pod=cluster1-pxc-2_
↪default(87cdf1a8-0fb3-4bc0-b50d-f66a0a73c087); ",
        "ready":2,
        "size":3,
        "status":"initializing"
     },
     "state":"initializing"
  }
}
```

31.7 Pass custom my.cnf during the creation of Percona XtraDB Cluster

Description:

```
Create a custom config map containing the contents of the file my.cnf to be
passed on to the Percona XtraDB Cluster containers when they are created
```

Kubectl Command:

```
kubectl create configmap cluster1-pxc3 --from-file=my.cnf
```

my.cnf (Contains mysql configuration):

```
[mysqld]
max_connections=250
```

URL:

```
https://$API_SERVER/api/v1/namespaces/default/configmaps
```

Authentication:

```
Authorization: Bearer $KUBE_TOKEN
```

cURL Request:

```
curl -k -v -XPOST "https://$API_SERVER/api/v1/namespaces/default/configmaps" \
            -H "Accept: application/json" \
            -H "Authorization: Bearer $KUBE_TOKEN" \
            -d '{"apiVersion":"v1","data":{"my.cnf":"[mysqld]\nmax_connections=250\n"},
↪"kind":"ConfigMap","metadata":{"creationTimestamp":null,"name":"cluster1-pxc3"}}' \
            -H "Content-Type: application/json"
```

Request Body:

JSON:


```
{
   "apiVersion":"v1",
   "data":{
      "my.cnf":"[mysqld]\nmax_connections=250\n"
   },
   "kind":"ConfigMap",
   "metadata":{
      "creationTimestamp":null,
      "name":"cluster1-pxc3"
   }
}
```


Input:

1. data (Object {filename : contents(String, min-length:0)}): `contains filenames to create in config map and its contents`
2. metadata: name(String, min-length: 1) : `contains name of the configmap`

3. kind (String): `type of object to create`

Response:

JSON:


```
{
   "kind":"ConfigMap",
   "apiVersion":"v1",
   "metadata":{
      "name":"cluster1-pxc3",
      "namespace":"default",
      "selfLink":"/api/v1/namespaces/default/configmaps/cluster1-pxc3",
      "uid":"d92c7196-f399-4e20-abc7-b5de62c0691b",
      "resourceVersion":"85258",
      "creationTimestamp":"2020-05-28T14:19:41Z",
      "managedFields":[
         {
            "manager":"kubectl",
            "operation":"Update",
            "apiVersion":"v1",
            "time":"2020-05-28T14:19:41Z",
            "fieldsType":"FieldsV1",
            "fieldsV1":{
               "f:data":{
                  ".":{

                  },
                  "f:my.cnf":{

                  }
               }
            }
         }
      ]
   },
   "data":{
      "my.cnf":""
   }
}
```


31.8 Backup Percona XtraDB Cluster

Description:

```
Takes a backup of the Percona XtraDB Cluster containers data to be able to
recover from disasters or make a roll-back later
```

Kubectl Command:

```
kubectl apply -f percona-xtradb-cluster-operator/deploy/backup/backup.yaml
```

URL:

```
https://$API_SERVER/apis/pxc.percona.com/v1/namespaces/default/
↪perconaxtradbclusterbackups
```

Authentication:

```
Authorization: Bearer $KUBE_TOKEN
```

cURL Request:

```
curl -k -v -XPOST "https://$API_SERVER/apis/pxc.percona.com/v1/namespaces/default/
↪perconaxtradbclusterbackups" \
            -H "Accept: application/json" \
            -H "Content-Type: application/json" \
            -d "@backup.json" -H "Authorization: Bearer $KUBE_TOKEN"
```

Request Body (backup.json):

JSON:


```
{
   "apiVersion":"pxc.percona.com/v1",
   "kind":"PerconaXtraDBClusterBackup",
   "metadata":{
      "name":"backup1"
   },
   "spec":{
      "pxcCluster":"cluster1",
      "storageName":"fs-pvc"
   }
}
```


Input:

1. **metadata**:

   name(String, min-length:1) : `name of backup to create`

2. **spec**:

   1. pxcCluster(String, min-length:1) : `name of Percona XtraDB Cluster`
   2. storageName(String, min-length:1) : `name of storage claim to use`

Response:

JSON:


```
{
   "apiVersion":"pxc.percona.com/v1",
   "kind":"PerconaXtraDBClusterBackup",
   "metadata":{
      "creationTimestamp":"2020-05-27T23:56:33Z",
      "generation":1,
      "managedFields":[
         {
            "apiVersion":"pxc.percona.com/v1",
            "fieldsType":"FieldsV1",
```


(continues on next page)

(continued from previous page)


```
            "fieldsV1":{
               "f:spec":{
                  ".":{

                  },
                  "f:pxcCluster":{

                  },
                  "f:storageName":{

                  }
               }
            },
            "manager":"kubectl",
            "operation":"Update",
            "time":"2020-05-27T23:56:33Z"
         }
      ],
      "name":"backup1",
      "namespace":"default",
      "resourceVersion":"26024",
      "selfLink":"/apis/pxc.percona.com/v1/namespaces/default/
↪perconaxtradbclusterbackups/backup1",
      "uid":"95a354b1-e25b-40c3-8be4-388acca055fe"
   },
   "spec":{
      "pxcCluster":"cluster1",
      "storageName":"fs-pvc"
   }
}
```


31.9 Restore Percona XtraDB Cluster

Description:

```
Restores Percona XtraDB Cluster data to an earlier version to recover from a
problem or to make a roll-back
```

Kubectl Command:

```
kubectl apply -f percona-xtradb-cluster-operator/deploy/backup/restore.yaml
```

URL:

```
https://$API_SERVER/apis/pxc.percona.com/v1/namespaces/default/
↪perconaxtradbclusterrestores
```

Authentication:

```
Authorization: Bearer $KUBE_TOKEN
```

cURL Request:

```
curl -k -v -XPOST "https://$API_SERVER/apis/pxc.percona.com/v1/namespaces/default/
↪perconaxtradbclusterrestores" \
            -H "Accept: application/json" \
            -H "Content-Type: application/json" \
            -d "@restore.json" \
            -H "Authorization: Bearer $KUBE_TOKEN"
```

Request Body (restore.json):

JSON:


```
{
   "apiVersion":"pxc.percona.com/v1",
   "kind":"PerconaXtraDBClusterRestore",
   "metadata":{
      "name":"restore1"
   },
   "spec":{
      "pxcCluster":"cluster1",
      "backupName":"backup1"
   }
}
```


Input:

1. **metadata**:

   name(String, min-length:1): `name of restore to create`

2. **spec**:

   1. pxcCluster(String, min-length:1) : `name of Percona XtraDB Cluster`
   2. backupName(String, min-length:1) : `name of backup to restore from`

Response:

JSON:


```
{
   "apiVersion":"pxc.percona.com/v1",
   "kind":"PerconaXtraDBClusterRestore",
   "metadata":{
      "creationTimestamp":"2020-05-27T23:59:41Z",
      "generation":1,
      "managedFields":[
         {
            "apiVersion":"pxc.percona.com/v1",
            "fieldsType":"FieldsV1",
            "fieldsV1":{
               "f:spec":{
                  ".":{

                  },
                  "f:backupName":{
```


(continues on next page)

(continued from previous page)


```
                  },
                  "f:pxcCluster":{

                  }
               }
            },
            "manager":"kubectl",
            "operation":"Update",
            "time":"2020-05-27T23:59:41Z"
         }
      ],
      "name":"restore1",
      "namespace":"default",
      "resourceVersion":"26682",
      "selfLink":"/apis/pxc.percona.com/v1/namespaces/default/
↪perconaxtradbclusterrestores/restore1",
      "uid":"770c3471-be17-46fb-b0a6-e706685ab2fc"
   },
   "spec":{
      "backupName":"backup1",
      "pxcCluster":"cluster1"
   }
}
```

CHAPTER

THIRTYTWO

FREQUENTLY ASKED QUESTIONS

32.1 Why do we need to follow "the Kubernetes way" when Kubernetes was never intended to run databases?

As it is well known, the Kubernetes approach is targeted at stateless applications but provides ways to store state (in Persistent Volumes, etc.) if the application needs it. Generally, a stateless mode of operation is supposed to provide better safety, sustainability, and scalability, it makes the already-deployed components interchangeable. You can find more about substantial benefits brought by Kubernetes to databases in this blog post.

The architecture of state-centric applications (like databases) should be composed in a right way to avoid crashes, data loss, or data inconsistencies during hardware failure. Percona Distribution for MySQL Operator provides out-of-the-box functionality to automate provisioning and management of highly available MySQL database clusters on Kubernetes.

32.2 How can I contact the developers?

The best place to discuss Percona Distribution for MySQL Operator with developers and other community members is the community forum.

If you would like to report a bug, use the Percona Distribution for MySQL Operator project in JIRA.

32.3 What is the difference between the Operator quickstart and advanced installation ways?

As you have noticed, the installation section of docs contains both quickstart and advanced installation guides.

The quickstart guide is simpler. It has fewer installation steps in favor of predefined default choices. Particularly, in advanced installation guides, you separately apply the Custom Resource Definition and Role-based Access Control configuration files with possible edits in them. At the same time, quickstart guides rely on the all-inclusive bundle configuration.

At another point, quickstart guides are related to specific platforms you are going to use (Minikube, Google Kubernetes Engine, etc.) and therefore include some additional steps needed for these platforms.

Generally, rely on the quickstart guide if you are a beginner user of the specific platform and/or you are new to the Percona Distribution for MySQL Operator as a whole.

32.4 Which versions of MySQL the Percona Distribution for MySQL Operator supports?

Percona Distribution for MySQL Operator provides a ready-to-use installation of the MySQL-based Percona XtraDB Cluster inside your Kubernetes installation. It works with both MySQL 8.0 and 5.7 branches, and the exact version is determined by the Docker image in use.

Percona-certified Docker images used by the Operator are listed here. As you can see, both Percona XtraDB Cluster 8.0 and 5.7 are supported with the following recommended versions: 8.0.23-14.1 and 5.7.34-31.51. Three major numbers in the XtraDB Cluster version refer to the version of Percona Server in use. More details on the exact Percona Server version can be found in the release notes (8.0, 5.7).

32.5 How HAProxy is better than ProxySQL?

Percona Distribution for MySQL Operator supports both HAProxy and ProxySQL as a load balancer. HAProxy is turned on by default, but both solutions are similar in terms of their configuration and operation under the control of the Operator.

Still, they have technical differences. HAProxy is a general and widely used high availability, load balancing, and proxying solution for TCP and HTTP-based applications. ProxySQL provides similar functionality but is specific to MySQL clusters. As an SQL-aware solution, it is able to provide more tight internal integration with MySQL instances.

Both projects do a really good job with Percona Distribution for MySQL Operator. The proxy choice should depend mostly on application-specific workload (including object-relational mapping), performance requirements, advanced routing and caching needs with one or another project, components already in use in the current infrastructure, and any other specific needs of the application.

32.6 How can I add custom sidecar containers to my cluster?

The Operator allows you to deploy additional (so-called *sidecar*) containers to the Pod. You can use this feature to run debugging tools, some specific monitoring solutions, etc. Add such sidecar container to the `deploy/cr.yaml` configuration file, specifying its name and image, and possibly a command to run:

```
spec:
  pxc:
    ....
    sidecars:
    - image: busybox
      command: ["/bin/sh"]
      args: ["-c", "while true; do echo echo $(date -u) 'test' >> /dev/null; sleep 5;␣
↪done"]
      name: my-sidecar-1
    ....
```

You can add `sidecars` subsection to `pxc`, `haproxy`, and `proxysql` sections.

Note: Custom sidecar containers can easily access other components of your cluster. Therefore they should be used carefully and by experienced users only.

32.7 How to get core dumps in case of the Percona XtraDB Cluster crash

In the Percona XtraDB Cluster crash case, gathering all possible information for enhanced diagnostics to be shared with Percona Support helps to solve an issue faster. One of such helpful artifacts is core dump.

Percona XtraDB Cluster can create core dumps on crush using libcoredumper. The Operator has this feature turned on by default. Core dumps are saved to `DATADIR` (`var/lib/mysql/`). You can find appropriate core files in the following way (substitute `some-name-pxc-1` with the name of your Pod):

```
kubectl exec some-name-pxc-1 -c pxc -it -- sh -c 'ls -alh /var/lib/mysql/ | grep core'
-rw------- 1 mysql mysql 1.3G Jan 15 09:30 core.20210015093005
```

When identified, the appropriate core dump can be downloaded as follows:

```
kubectl cp some-name-pxc-1:/var/lib/mysql/core.20210015093005  /tmp/core.20210015093005
```

Note: It is useful to provide Build ID and Server Version in addition to core dump when Creating a support ticket. Both can be found from logs:

```
kubectl logs some-name-pxc-1 -c logs

[1] init-deploy-949.some-name-pxc-1.mysqld-error.log: [1610702394.259356066, {"log"=>
↪"09:19:54 UTC - mysqld got signal 11 ;"}]
[2] init-deploy-949.some-name-pxc-1.mysqld-error.log: [1610702394.259356829, {"log"=>
↪"Most likely, you have hit a bug, but this error can also be caused by malfunctioning␣
↪hardware."}]
```

(continues on next page)

(continued from previous page)

```
[3] init-deploy-949.some-name-pxc-1.mysqld-error.log: [1610702394.259457282, {"log"=>
↪"Build ID: 5a2199b1784b967a713a3bde8d996dc517c41adb"}]
[4] init-deploy-949.some-name-pxc-1.mysqld-error.log: [1610702394.259465692, {"log"=>
↪"Server Version: 8.0.21-12.1 Percona XtraDB Cluster (GPL), Release rel12, Revision␣
↪4d973e2, WSREP version 26.4.3, wsrep_26.4.3"}]
.....
```

32.8 How to choose between HAProxy and ProxySQL when configuring the cluster?

You can configure the Operator to use one of two different proxies, HAProxy (the default choice) and ProxySQL. Both solutions are fully supported by the Operator, but they have some differences in the architecture, which can make one of them more suitable then the other one in some use cases.

The main difference is that HAProxy operates in TCP mode as an OSI level 4 proxy, while ProxySQL implements OSI level 7 proxy, and thus can provide some additional functionality like read/write split, firewalling and caching.

From the other side, utilizing HAProxy for the service is the easier way to go, and getting use of the ProxySQL level 7 specifics requires good understanding of Kubernetes and ProxySQL.

See more detailed functionality and performance comparison of using the Operator with both solutions in this blog post.

CHAPTER

THIRTYTHREE

PERCONA DISTRIBUTION FOR MYSQL OPERATOR 1.9.0 RELEASE NOTES

33.1 *Percona Distribution for MySQL Operator* 1.9.0

Date August 9, 2021

Installation For installation please refer to the documentation page

33.1.1 Release Highlights

- Starting from this release, the Operator changes its official name to **Percona Distribution for MySQL Operator**. This new name emphasizes gradual changes which incorporated a collection of Percona's solutions to run and operate Percona Server for MySQL and Percona XtraDB Cluster, available separately as Percona Distribution for MySQL.
- Now you can see HAProxy metrics in your favorite Percona Monitoring and Management (PMM) dashboards automatically.
- The *cross-site replication* feature allows an asynchronous replication between two Percona XtraDB Clusters, including scenarios when one of the clusters is outside of the Kubernetes environment. The feature is intended for the following use cases:
  - provide migrations of your Percona XtraDB Cluster to Kubernetes or vice versa,
  - migrate regular MySQL database to Percona XtraDB Cluster under the Operator control, or carry on backward migration,
  - enable disaster recovery capability for your cluster deployment.

33.1.2 New Features

- K8SPXC-657: Use Secrets to store custom configuration with sensitive data for *Percona XtraDB Cluster*, *HAProxy*, and *ProxySQL* Pods
- K8SPXC-308: Implement Percona XtraDB Cluster *asynchronous replication* within the Operator
- K8SPXC-688: Define *environment variables* in the Custom Resource to provide containers with additional customizations

33.1.3 Improvements

- K8SPXC-673: HAProxy Pods now come with Percona Monitoring and Management integration and support
- K8SPXC-791: Allow *stopping the restart-on-fail loop* for Percona XtraDB Cluster and Log Collector Pods without special debug images
- K8SPXC-764: Unblock backups even if just a single instance is available by setting the `allowUnsafeConfigurations` flag to true
- K8SPXC-765: Automatically delete custom configuration ConfigMaps if the variable in Custom Resource was unset (Thanks to Oleksandr Levchenkov for contributing)
- K8SPXC-734: Simplify manual recovery by automatically getting Percona XtraDB Cluster namespace in the pxc container entrypoint script (Thanks to Michael Lin for contributing)
- K8SPXC-656: imagePullPolicy is now set for init container as well to avoid pulling and simplifying deployments in air-gapped environments (Thanks to Herberto Graça for contributing)
- K8SPXC-511: Secret object containing system users passwords is now deleted along with the Cluster if `delete-pxc-pvc` finalizer is enabled (Thanks to Matthias Baur for contributing)
- K8SPXC-772: All Service objects now have Percona XtraDB Cluster labels attached to them to enable label selector usage
- K8SPXC-731: It is now possible to see the overall progress of the provisioning of Percona XtraDB Cluster resources and dependent components in Custom Resource status
- K8SPXC-730: Percona XtraDB Cluster resource statuses in Custom Resource output (e.g. returned by `kubectl get pxc` command) have been improved and now provide more precise reporting
- K8SPXC-697: Add namespace support in the `copy-backup` script
- K8SPXC-321, K8SPXC-556, K8SPXC-568: Restrict the minimal number of ProxySQL and HAProxy Pods and the maximal number of Percona XtraDB Cluster Pods if the unsafe flag is not set
- K8SPXC-554: Reduced the number of various etcd and k8s object updates from the Operator to minimize the pressure on the Kubernetes cluster
- K8SPXC-421: It is now possible to use X Plugin with Percona XtraDB Cluster Pods

33.1.4 Known Issues and Limitations

- K8SPXC-835: ProxySQL will fail to start on a Replica Percona XtraDB Cluster for cross-site replication in this release

33.1.5 Bugs Fixed

- K8SPXC-757: Fixed a bug where manual crash recovery interfered with auto recovery functionality even with the `auto_recovery` flag set to false
- K8SPXC-706: TLS certificates *renewal by a cert-manager was failing* (Thanks to Jeff Andrews for reporting this issue)
- K8SPXC-785: Fixed a bug where backup to S3 was producing false-positive error messages even if backup was successful
- K8SPXC-642: Fixed a bug where PodDisruptionBudget was blocking the upgrade of HAProxy (Thanks to Davi S Evangelista for reporting this issue)

- K8SPXC-585: Fixed a bug where the Operator got stuck if the wrong user credentials were set in the Secret object (Thanks to Sergiy Prykhodko for reporting this issue)
- K8SPXC-756: Fixed a bug where the Operator was scheduling backups even when the cluster was paused (Thanks to Dmytro for reporting this issue)
- K8SPXC-813: Fixed a bug where backup restore didn't return error on incorrect AWS credentials
- K8SPXC-805: Fixed a bug that made pxc-backups object deletion hang if the Operator couldn't list objects from the S3 bucket (e.g. due to wrong S3 credentials)
- K8SPXC-787: Fixed the "initializing" status of ready clusters caused by the xtrabackup user password change
- K8SPXC-775: Fixed a bug where errors in custom mysqld config settings were not detected by the Operator if the config was modified after the initial cluster was created
- K8SPXC-767: Fixed a bug where on-demand backup hung up if created while the cluster was in the "initializing" state
- K8SPXC-726: Fixed a bug where the `delete-s3-backup` finalizer prevented deleting a backup stored on Persistent Volume
- K8SPXC-682: Fixed auto-tuning feature setting wrong `innodb_buffer_pool_size` value in some cases

33.2 *Percona Kubernetes Operator for Percona XtraDB Cluster* 1.8.0

Date April 26, 2021

Installation Installing Percona Kubernetes Operator for Percona XtraDB Cluster

33.2.1 Release Highlights

- It is now possible to use `kubectl scale` command to scale Percona XtraDB Cluster horizontally (add or remove Replica Set instances). You can also use Horizontal Pod Autoscaler which will scale your database cluster based on various metrics, such as CPU utilization.
- Support for *custom sidecar containers*. The Operator makes it possible now to deploy additional (sidecar) containers to the Pod. This feature can be useful to run debugging tools or some specific monitoring solutions, etc. Sidecar containers can be added to *pxc*, *haproxy*, and *proxysql* sections of the `deploy/cr.yaml` configuration file.

33.2.2 New Features

- K8SPXC-528: Support for *custom sidecar containers* to extend the Operator capabilities
- K8SPXC-647: Allow the cluster *scale in and scale out* with the `kubectl scale` command or Horizontal Pod Autoscaler
- K8SPXC-643: Operator can now automatically recover Percona XtraDB Cluster after the network partitioning

33.2.3 Improvements

- K8SPXC-442: The Operator can now automatically remove old backups from S3 storage if the retention period is set (thanks to Davi S Evangelista for reporting this issue)
- K8SPXC-697: Add namespace support in the *script used to copy backups* from remote storage to a local machine
- K8SPXC-627: Point-in-time recovery uploader now chooses the Pod with the oldest binary log in the cluster to ensure log consistency
- K8SPXC-618: Add debug symbols from the percona-xtradb-cluster-server-debuginfo package to the Percona XtraDB Cluster debug docker image to simplify troubleshooting
- K8SPXC-599: It is now possible to *recover* databases up to a specific transaction with the Point-in-time Recovery feature. Previously the user could only recover to specific date and time
- K8SPXC-598: Point-in-time recovery feature now works with compressed backups
- K8SPXC-536: It is now possible to explicitly set the version of Percona XtraDB Cluster for newly provisioned clusters. Before that, all new clusters were started with the latest PXC version if Version Service was enabled
- K8SPXC-522: Add support for the `runtimeClassName` Kubernetes feature for selecting the container runtime
- K8SPXC-519, K8SPXC-558, and K8SPXC-637: Various improvements of Operator log messages

33.2.4 Known Issues and Limitations

- K8SPXC-701: Scheduled backups are not compatible with Kubernetes 1.20 in cluster-wide mode.

33.2.5 Bugs Fixed

- K8SPXC-654: Use MySQL administrative port for Kubernetes liveness/readiness probes to avoid false positive failures
- K8SPXC-614, K8SPXC-619, K8SPXC-545, K8SPXC-641, K8SPXC-576: Fix multiple bugs due to which changes of various objects in `deploy/cr.yaml` were not applied to the running cluster (thanks to Sergiy Prykhodko for reporting some of these issues)
- K8SPXC-596: Fix a bug due to which liveness probe for `pxc` container could cause zombie processes
- K8SPXC-632: Fix a bug preventing point-in-time recovery when multiple clusters were uploading binary logs to a single S3 bucket
- K8SPXC-573: Fix a bug that prevented using special characters in XtraBackup password (thanks to Gertjan Bijl for reporting this issue)
- K8SPXC-571: Fix a bug where Percona XtraDB Cluster went into a desynced state at backup job crash (Thanks to Dimitrij Hilt for reporting this issue)
- K8SPXC-430: Galera Arbitrator used for backups does not break the cluster anymore in various cases
- K8SPXC-684: Fix a bug due to which point-in-time recovery backup didn't allow specifying the `endpointUrl` for Amazon S3 storage
- K8SPXC-681: Fix operator crash which occurred when non-existing storage name was specified for point-in-time recovery
- K8SPXC-638: Fix unneeded delay in showing logs with the `kubectl logs` command for the logs container
- K8SPXC-609: Fix frequent HAProxy service NodePort updates which were causing issues with load balancers

- K8SPXC-542: Fix a bug due to which backups were taken only for one cluster out of many controlled by one Operator
- CLOUD-611: Stop using the already deprecated runtime/scheme package (Thanks to Jerome Küttner for reporting this issue)

33.3 *Percona Kubernetes Operator for Percona XtraDB Cluster* 1.7.0

Date February 2, 2021

Installation Installing Percona Kubernetes Operator for Percona XtraDB Cluster

33.3.1 New Features

- K8SPXC-530: Add support for *point-in-time recovery*
- K8SPXC-564: PXC cluster will now recover automatically from a full crash when Pods are stuck in CrashLoopBackOff status
- K8SPXC-497: Official support for *Percona Monitoring and Management (PMM) v.2*

Note: Monitoring with PMM v.1 configured according to the unofficial instruction will not work after the upgrade. Please switch to PMM v.2.

33.3.2 Improvements

- K8SPXC-485: *Percona XtraDB Cluster Pod logs are now stored on Persistent Volumes*. Users can debug the issues even after the Pod restart
- K8SPXC-389: User can now change ServiceType for HAProxy replicas Kubernetes service
- K8SPXC-546: Reduce the number of ConfigMap object updates from the Operator to improve performance of the Kubernetes cluster
- K8SPXC-553: Change default configuration of ProxySQL to WRITERS_ARE_READERS=yes so Percona XtraDB Cluster continues operating with a single node left
- K8SPXC-512: User can now limit cluster-wide Operator access to specific namespaces (Thanks to user mgar for contribution)
- K8SPXC-490: Improve error message when not enough memory is set for auto-tuning
- K8SPXC-312: Add schema validation for Custom Resource. Now `cr.yaml` is validated by a WebHook for syntax typos before being applied. It works only in cluster-wide mode due to access restrictions
- K8SPXC-510: Percona XtraDB Cluster operator can now be deployed through RedHat Marketplace
- K8SPXC-543: Check HAProxy custom configuration for syntax errors before applying it to avoid Pod getting stuck in CrashLoopBackOff status (Thanks to user pservit for reporting this issue)

33.3.3 Bugs Fixed

- K8SPXC-544: Add a liveness probe for HAProxy so it is not stuck and automatically restarted when crashed (Thanks to user pservit for reporting this issue)
- K8SPXC-500: Fix a bug that prevented creating a backup in cluster-wide mode if default cr.yaml is used (Thanks to user michael.lin1 for reporting this issue)
- K8SPXC-491: Fix a bug due to which compressed backups didn't work with the Operator (Thanks to user dejw for reporting this issue)
- K8SPXC-570: Fix a bug causing backups to fail with some S3-compatible storages (Thanks to user dimitrij for reporting this issue)
- K8SPXC-517: Fix a bug causing Operator crash if Custom Resource backup section is missing (Thanks to user deamonmv for reporting this issue)
- K8SPXC-253: Fix a bug preventing rolling out Custom Resource changes (Thanks to user bitsbeats for reporting this issue)
- K8SPXC-552: Fix a bug when HAProxy secrets cannot be updated by the user
- K8SPXC-551: Fix a bug due to which cluster was not initialized when the password had an end of line symbol in `secret.yaml`
- K8SPXC-526: Fix a bug due to which not all clusters managed by the Operator were upgraded by the automatic update
- K8SPXC-523: Fix a bug putting cluster into unhealthy status after the clustercheck secret changed
- K8SPXC-521: Fix automatic upgrade job repeatedly looking for an already removed cluster
- K8SPXC-520: Fix Smart update in cluster-wide mode adding version service check job repeatedly instead of doing it only once
- K8SPXC-463: Fix a bug due to which wsrep_recovery log was unavailable after the Pod restart
- K8SPXC-424: Fix a bug due to which HAProxy health-check spammed in logs, making them hardly unreadable
- K8SPXC-379: Fix a bug due to which the Operator user credentials were not added into internal secrets when upgrading from 1.4.0 (Thanks to user pservit for reporting this issue)

33.4 *Percona Kubernetes Operator for Percona XtraDB Cluster* 1.6.0

Date October 9, 2020

Installation Installing Percona Kubernetes Operator for Percona XtraDB Cluster

33.4.1 New Features

- K8SPXC-394: Support of *"cluster-wide" mode* for Percona XtraDB Cluster Operator
- K8SPXC-416: *Support of the proxy-protocol* in HAProxy (to use this feature, you should have a Percona XtraDB Cluster image version `8.0.21` or newer)
- K8SPXC-429: A possibility to *restore backups to a new Kubernetes-based environment* with no existing Percona XtraDB Cluster Custom Resource
- K8SPXC-343: Helm chart *officially provided with the Operator*

33.4.2 Improvements

- K8SPXC-144: Allow *adding ProxySQL configuration options*
- K8SPXC-398: New `crVersion key` in `deploy/cr.yaml` to indicate the API version that the Custom Resource corresponds to (thanks to user mike.saah for contribution)
- K8SPXC-474: The init container now has the same resource requests as the main container of a correspondent Pod (thanks to user yann.leenhardt for contribution)
- K8SPXC-372: Support new versions of cert-manager by the Operator (thanks to user rf_enigm for contribution)
- K8SPXC-317: Possibility to configure the `imagePullPolicy` Operator option (thanks to user imranrazakhan for contribution)
- K8SPXC-462: Add readiness probe for HAProxy
- K8SPXC-411: Extend cert-manager configuration to add additional domains (multiple SAN) to a certificate
- K8SPXC-375: Improve HAProxy behavior in case of switching writer node to a new one and back
- K8SPXC-368: Autoupdate system users by changing the appropriate Secret name

33.4.3 Known Issues and Limitations

- OpenShift 3.11 requires additional configuration for the correct HAProxy operation: the feature gate `PodShareProcessNamespace` should be set to `true`. If getting it enabled is not possible, we recommend using ProxySQL instead of HAProxy with OpenShift 3.11. Other OpenShift and Kubernetes versions are not affected.
- K8SPXC-491: Compressed backups are not compatible with the Operator 1.6.0 (`percona/percona-xtradb-cluster-operator:1.5.0-pxc8.0-backup` or `percona/percona-xtradb-cluster-operator:1.5.0-pxc5.7-backup` image can be used as a workaround if needed).

33.4.4 Bugs Fixed

- K8SPXC-431: HAProxy unable to start on OpenShift with the default `cr.yaml` file
- K8SPXC-408: Insufficient MAX_USER_CONNECTIONS=10 for ProxySQL monitor user (increased to 100)
- K8SPXC-391: HAProxy and PMM cannot be enabled at the same time (thanks to user rf_enigm for reporting this issue)
- K8SPXC-406: Second node (XXX-pxc-1) always selected as a donor (thanks to user pservit for reporting this issue)
- K8SPXC-390: Crash on missing HAProxy PodDisruptionBudget
- K8SPXC-355: Counterintuitive YYYY-DD-MM dates in the S3 backup folder names (thanks to user graham-web for contribution)
- K8SPXC-305: ProxySQL not working in case of passwords with a `%` symbol in the Secrets object (thanks to user ben.wilson for reporting this issue)
- K8SPXC-278: ProxySQL never getting ready status in some environments after the cluster launch due to the `proxysql-monit` Pod crash (thanks to user lots0logs for contribution)
- K8SPXC-274: The 1.2.0 -> 1.3.0 -> 1.4.0 upgrade path not working (thanks to user martin.atroo for reporting this issue)

- K8SPXC-476: SmartUpdate failing to fetch version from Version Service in case of incorrectly formatted Percona XtraDB Cluster patch version higher than the last known one
- K8SPXC-454: After the cluster creation, pxc-0 Pod restarting due to Operator not waiting for cert-manager to issue requested certificates (thanks to user mike.saah for reporting this issue)
- K8SPXC-450: TLS annotations causing unnecessary HAProxy Pod restarts
- K8SPXC-443 and K8SPXC-456: The outdated version service endpoint URL (fix with preserving backward compatibility)
- K8SPXC-435: MySQL root password visible through `kubectl logs`
- K8SPXC-426: mysqld recovery logs not logged to file and not available through `kubectl logs`
- K8SPXC-423: HAProxy not refreshing IP addresses even when the node gets a different address
- K8SPXC-419: Percona XtraDB Cluster incremental state transfers not taken into account by readiness/liveness checks
- K8SPXC-418: HAProxy not routing traffic for 1 donor, 2 joiners
- K8SPXC-417: Cert-manager not compatible with Kubernetes versions below v1.15 due to unnecessarily high API version demand
- K8SPXC-384: Debug images were not fully functional for the latest version of the Operator because of having no infinity loop
- K8SPXC-383: DNS warnings in PXC Pods when using HAProxy
- K8SPXC-364: Smart Updates showing empty "from" versions for non-PXC objects in logs
- K8SPXC-379: The Operator user credentials not added into internal secrets when upgrading from 1.4.0 (thanks to user pservit for reporting this issue)

33.5 *Percona Kubernetes Operator for Percona XtraDB Cluster* 1.5.0

Date July 21, 2020

Installation Installing Percona Kubernetes Operator for Percona XtraDB Cluster

33.5.1 New Features

- K8SPXC-298: Automatic synchronization of MySQL users with ProxySQL
- K8SPXC-294: HAProxy Support
- K8SPXC-284: Fully automated minor version updates (Smart Update)
- K8SPXC-257: Update Reader members before Writer member at cluster upgrades
- K8SPXC-256: Support multiple PXC minor versions by the Operator

33.5.2 Improvements

- K8SPXC-290: Extend usable backup schedule syntax to include lists of values
- K8SPXC-309: Quickstart Guide on Google Kubernetes Engine (GKE) - link
- K8SPXC-288: Quickstart Guide on Amazon Elastic Kubernetes Service (EKS) - link
- K8SPXC-280: Support XtraBackup compression
- K8SPXC-279: Use SYSTEM_USER privilege for system users on PXC 8.0
- K8SPXC-277: Install GDB in PXC images
- K8SPXC-276: Pod-0 should be selected as Writer if possible
- K8SPXC-252: Automatically manage system users for MySQL and ProxySQL on password rotation via Secret
- K8SPXC-242: Improve internal backup implementation for better stability with PXC 8.0
- CLOUD-404: Support of loadBalancerSourceRanges for LoadBalancer Services
- CLOUD-556: Kubernetes 1.17 added to the list of supported platforms

33.5.3 Bugs Fixed

- K8SPXC-327: CrashloopBackOff if PXC 8.0 Pod restarts in the middle of SST
- K8SPXC-291: PXC Restore failure with "The node was low on resource: ephemeral-storage" error (Thanks to user rjeka for reporting this issue)
- K8SPXC-270: Restore job wiping data from the original backup's cluster when restoring to another cluster in the same namespace
- K8SPXC-352: Backup cronjob not scheduled in some Kubernetes environments (Thanks to user msavchenko for reporting this issue)
- K8SPXC-275: Outdated documentation on the Operator updates (Thanks to user martin.atroo for reporting this issue)
- K8SPXC-347: XtraBackup failure after uploading a backup, causing the backup process restart in some cases (Thanks to user connde for reporting this issue)
- K8SPXC-373: Pod not cleaning up the SST tmp dir on start
- K8SPXC-326: Changes in TLS Secrets not triggering PXC restart if AllowUnsafeConfig enabled
- K8SPXC-323: Missing `tar` utility in the PXC node docker image
- CLOUD-531: Wrong usage of `strings.TrimLeft` when processing apiVersion
- CLOUD-474: Cluster creation not failing if wrong resources are set

33.6 *Percona Kubernetes Operator for Percona XtraDB Cluster* 1.4.0

Date April 29, 2020

Installation Installing Percona Kubernetes Operator for Percona XtraDB Cluster

33.6.1 New Features

- K8SPXC-172: Full data-at-rest encryption available in PXC 8.0 is now supported by the Operator. This feature is implemented with the help of the `keyring_vault` plugin which ships with PXC 8.0. By utilizing Vault we enable our customers to follow best practices with encryption in their environment.
- K8SPXC-125: Percona XtraDB Cluster 8.0 is now supported
- K8SPXC-95: Amazon Elastic Container Service for Kubernetes (EKS) was added to the list of the officially supported platforms
- The OpenShift Container Platform 4.3 is now supported

33.6.2 Improvements

- K8SPXC-262: The Operator allows setting ephemeral-storage requests and limits on all Pods
- K8SPXC-221: The Operator now updates observedGeneration status message to allow better monitoring of the cluster rollout or backup/restore process
- K8SPXC-213: A special *PXC debug image* is now available. It avoids restarting on fail and contains additional tools useful for debugging
- K8SPXC-100: The Operator now implements the crash tolerance on the one member crash. The implementation is based on starting Pods with `mysqld --wsrep_recover` command if there was no graceful shutdown

33.6.3 Bugs Fixed

- K8SPXC-153: S3 protocol credentials were not masked in logs during the PXC backup & restore process
- K8SPXC-222: The Operator got caught in reconciliation error in case of the erroneous/absent API version in the `deploy/cr.yaml` file
- K8SPXC-261: ProxySQL logs were showing the root password
- K8SPXC-220: The inability to update or delete existing CRD was possible because of too large records in etcd, resulting in "request is too large" errors. Only 20 last status changes are now stored in etcd to avoid this problem.
- K8SPXC-52: The Operator produced an unclear error message in case of fail caused by the absent or malformed pxc section in the `deploy/cr.yaml` file
- K8SPXC-269: The `copy-backup.sh` script didn's work correctly in case of an existing secret with the AWS_ACCESS_KEY_ID/AWS_SECRET_ACCESS_KEY credentials and prevented users from copying backups (e.g. to a local machine)
- K8SPXC-263: The `kubectl get pxc` command was unable to show the correct ProxySQL external endpoint
- K8SPXC-219: PXC Helm charts were incompatible with the version 3 of the Helm package manager
- K8SPXC-40: The cluster was unable to reach "ready" status in case if `ProxySQL.Enabled` field was set to `false`

- K8SPXC-34: Change of the `proxysql.servicetype` filed was not detected by the Operator and thus had no effect

33.7 *Percona Kubernetes Operator for Percona XtraDB Cluster* 1.3.0

Percona announces the *Percona Kubernetes Operator for Percona XtraDB Cluster* 1.3.0 release on January 6, 2020. This release is now the current GA release in the 1.3 series. Install the Kubernetes Operator for Percona XtraDB Cluster by following the instructions.

The Percona Kubernetes Operator for Percona XtraDB Cluster automates the lifecycle and provides a consistent Percona XtraDB Cluster instance. The Operator can be used to create a Percona XtraDB Cluster, or scale an existing Cluster and contains the necessary Kubernetes settings.

The Operator simplifies the deployment and management of the Percona XtraDB Cluster in Kubernetes-based environments. It extends the Kubernetes API with a new custom resource for deploying, configuring and managing the application through the whole life cycle.

The Operator source code is available in our Github repository. All of Percona's software is open-source and free.

New features and improvements:

- CLOUD-412: Auto-Tuning of the MySQL Parameters based on Pod memory resources was implemented in the case of Percona XtraDB Cluster Pod limits (or at least Pod requests) specified in the cr.yaml file.
- CLOUD-411: Now the user can adjust securityContext, replacing the automatically generated securityContext with the customized one.
- CLOUD-394: The Percona XtraDB Cluster, ProxySQL, and backup images size decrease by 40-60% was achieved by removing unnecessary dependencies and modules to reduce the cluster deployment time.
- CLOUD-390: Helm chart for Percona Monitoring and Management (PMM) 2.0 has been provided.
- CLOUD-383: Affinity constraints and tolerations were added to the backup Pod
- CLOUD-430: Image URL in the CronJob Pod template is automatically updated when the Operator detects changed backup image URL

Fixed bugs:

- CLOUD-462: Resource requests/limits were set not for all containers in a ProxySQL Pod
- CLOUD-437: Percona Monitoring and Management Client was taking resources definition from the Percona XtraDB Cluster despite having much lower need in resources, particularly lower memory footprint.
- CLOUD-434: Restoring Percona XtraDB Cluster was failing on the OpenShift platform with customized security settings
- CLOUD-399: The iputils package was added to the backup docker image to provide backup jobs with the ping command for a better network connection handling
- CLOUD-393: The Operator generated various StatefulSets in the first reconciliation cycle and in all subsequent reconciliation cycles, causing Kubernetes to trigger an unnecessary ProxySQL restart once during the cluster creation.
- CLOUD-376: A long-running SST caused the liveness probe check to fail it's grace period timeout, resulting in an unrecoverable failure
- CLOUD-243: Using *MYSQL_ROOT_PASSWORD* with special characters in a ProxySQL docker image was breaking the entrypoint initialization process

Percona XtraDB Cluster is an open source, cost-effective and robust clustering solution for businesses. It integrates Percona Server for MySQL with the Galera replication library to produce a highly-available and scalable MySQL® cluster complete with synchronous multi-primary replication, zero data loss and automatic node provisioning using Percona XtraBackup.

Help us improve our software quality by reporting any bugs you encounter using our bug tracking system.

33.8 *Percona Kubernetes Operator for Percona XtraDB Cluster* 1.2.0

Percona announces the *Percona Kubernetes Operator for Percona XtraDB Cluster* 1.2.0 release on September 20, 2019. This release is now the current GA release in the 1.2 series. Install the Kubernetes Operator for Percona XtraDB Cluster by following the instructions.

The Percona Kubernetes Operator for Percona XtraDB Cluster automates the lifecycle and provides a consistent Percona XtraDB Cluster instance. The Operator can be used to create a Percona XtraDB Cluster, or scale an existing Cluster and contains the necessary Kubernetes settings.

The Operator simplifies the deployment and management of the Percona XtraDB Cluster in Kubernetes-based environments. It extends the Kubernetes API with a new custom resource for deploying, configuring and managing the application through the whole life cycle.

The Operator source code is available in our Github repository. All of Percona's software is open-source and free.

New features and improvements:

- A Service Broker was implemented for the Operator, allowing a user to deploy Percona XtraDB Cluster on the OpenShift Platform, configuring it with a standard GUI, following the Open Service Broker API.
- Now the Operator supports Percona Monitoring and Management 2, which means being able to detect and register to PMM Server of both 1.x and 2.0 versions.
- A `NodeSelector` constraint is now supported for the backups, which allows using backup storage accessible to a limited set of nodes only (contributed by Chen Min).
- The resource constraint values were refined for all containers to eliminate the possibility of an out of memory error.
- Now it is possible to set the `schedulerName` option in the operator parameters. This allows using storage which depends on a custom scheduler, or a cloud provider which optimizes scheduling to run workloads in a cost-effective way (contributed by Smaine Kahlouch).
- A bug was fixed, which made cluster status oscillate between "initializing" and "ready" after an update.
- A 90 second startup delay which took place on freshly deployed Percona XtraDB Cluster was eliminated.

Percona XtraDB Cluster is an open source, cost-effective and robust clustering solution for businesses. It integrates Percona Server for MySQL with the Galera replication library to produce a highly-available and scalable MySQL® cluster complete with synchronous multi-primary replication, zero data loss and automatic node provisioning using Percona XtraBackup.

Help us improve our software quality by reporting any bugs you encounter using our bug tracking system.

33.9 *Percona Kubernetes Operator for Percona XtraDB Cluster* 1.1.0

Percona announces the general availability of *Percona Kubernetes Operator for Percona XtraDB Cluster* 1.1.0 on July 15, 2019. This release is now the current GA release in the 1.1 series. Install the Kubernetes Operator for Percona XtraDB Cluster by following the instructions.

The Percona Kubernetes Operator for Percona XtraDB Cluster automates the lifecycle and provides a consistent Percona XtraDB Cluster instance. The Operator can be used to create a Percona XtraDB Cluster, or scale an existing Cluster and contains the necessary Kubernetes settings.

The Operator simplifies the deployment and management of the Percona XtraDB Cluster in Kubernetes-based environments. It extends the Kubernetes API with a new custom resource for deploying, configuring and managing the application through the whole life cycle.

The Operator source code is available in our Github repository. All of Percona's software is open-source and free.

New features and improvements:

- Now the Percona Kubernetes Operator allows upgrading Percona XtraDB Cluster to newer versions, either in semi-automatic or in manual mode.
- Also, two modes are implemented for updating the Percona XtraDB Cluster `my.cnf` configuration file: in *automatic configuration update* mode Percona XtraDB Cluster Pods are immediately re-created to populate changed options from the Operator YAML file, while in *manual mode* changes are held until Percona XtraDB Cluster Pods are re-created manually.
- A separate service account is now used by the Operator's containers which need special privileges, and all other Pods run on default service account with limited permissions.
- User secrets are now generated automatically if don't exist: this feature especially helps reduce work in repeated development environment testing and reduces the chance of accidentally pushing predefined development passwords to production environments.
- The Operator is now able to generate TLS certificates itself which removes the need in manual certificate generation.
- The list of officially supported platforms now includes Minikube, which provides an easy way to test the Operator locally on your own machine before deploying it on a cloud.
- Also, Google Kubernetes Engine 1.14 and OpenShift Platform 4.1 are now supported.

Percona XtraDB Cluster is an open source, cost-effective and robust clustering solution for businesses. It integrates Percona Server for MySQL with the Galera replication library to produce a highly-available and scalable MySQL® cluster complete with synchronous multi-primary replication, zero data loss and automatic node provisioning using Percona XtraBackup.

Help us improve our software quality by reporting any bugs you encounter using our bug tracking system.

33.10 *Percona Kubernetes Operator for Percona XtraDB Cluster* 1.0.0

Percona announces the general availability of *Percona Kubernetes Operator for Percona XtraDB Cluster* 1.0.0 on May 29, 2019. This release is now the current GA release in the 1.0 series. Install the Kubernetes Operator for Percona XtraDB Cluster by following the instructions. Please see the GA release announcement. All of Percona's software is open-source and free.

The Percona Kubernetes Operator for Percona XtraDB Cluster automates the lifecycle and provides a consistent Percona XtraDB Cluster instance. The Operator can be used to create a Percona XtraDB Cluster, or scale an existing Cluster and contains the necessary Kubernetes settings.

The Percona Kubernetes Operators are based on best practices for configuration and setup of the Percona XtraDB Cluster. The Operator provides a consistent way to package, deploy, manage, and perform a backup and a restore for a Kubernetes application. Operators deliver automation advantages in cloud-native applications.

The advantages are the following:

- Deploy a Percona XtraDB Cluster environment with no single point of failure and environment can span multiple availability zones (AZs).
- Deployment takes about six minutes with the default configuration.
- Modify the Percona XtraDB Cluster size parameter to add or remove Percona XtraDB Cluster members
- Integrate with Percona Monitoring and Management (PMM) to seamlessly monitor your Percona XtraDB Cluster
- Automate backups or perform on-demand backups as needed with support for performing an automatic restore
- Supports using Cloud storage with S3-compatible APIs for backups
- Automate the recovery from failure of a single Percona XtraDB Cluster node
- TLS is enabled by default for replication and client traffic using Cert-Manager
- Access private registries to enhance security
- Supports advanced Kubernetes features such as pod disruption budgets, node selector, constraints, tolerations, priority classes, and affinity/anti-affinity
- You can use either PersistentVolumeClaims or local storage with hostPath to store your database
- Customize your MySQL configuration using ConfigMap.

33.10.1 Installation

Installation is performed by following the documentation installation instructions for Kubernetes and OpenShift.