Percona Kubernetes Operator for Percona Server for MongoDB Documentation

Release 1.8.0

May 06, 2021

CONTENTS

The Percona Kubernetes Operator for Percona Server for MongoDB automates the creation, modification, or deletion of items in your Percona Server for MongoDB environment. The Operator contains the necessary Kubernetes settings to maintain a consistent Percona Server for MongoDB instance.

The Percona Kubernetes Operators are based on best practices for the configuration of a Percona Server for MongoDB replica set. The Operator provides many benefits but saving time, a consistent environment are the most important.

Part I

Requirements

CHAPTER

ONE

SYSTEM REQUIREMENTS

The Operator was developed and tested with Percona Server for MongoDB 3.6, 4.0, 4.2, and 4.4. Other options may also work but have not been tested.

Note: The MMAPv1 storage engine is no longer supported for all MongoDB versions starting from the Operator version 1.6. MMAPv1 was already deprecated by MongoDB for a long time. WiredTiger is the default storage engine since MongoDB 3.2, and MMAPv1 was completely removed in MongoDB 4.2.

1.1 Officially supported platforms

The following platforms were tested and are officially supported by the Operator 1.8.0:

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

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 3 Nodes and the following resources (if *sharding* is turned off):

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

Consider using 4 CPU and 6 GB of RAM if *sharding* is turned on (the default behavior).

Also, the number of Replica Set Nodes should not be odd if *Arbiter* is not enabled.

Note: Use Storage Class with XFS as the default filesystem if possible to achieve better MongoDB performance.

1.3 Platform-specific limitations

The Operator is subsequent to specific platform limitations.

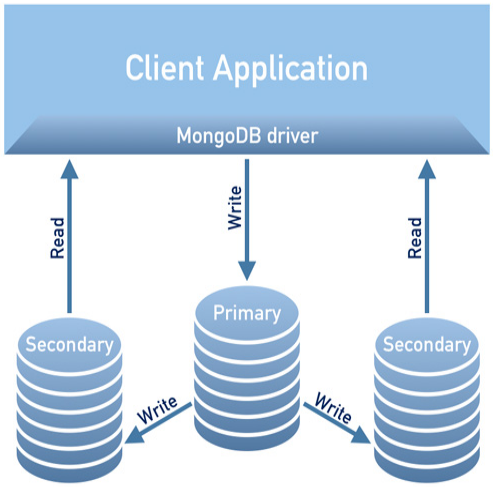
- Minikube doesn't support multi-node cluster configurations because of its local nature, which is in collision with the default affinity requirements of the Operator. To arrange this, the *Install Percona Server for MongoDB on Minikube* instruction includes an additional step which turns off the requirement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

CHAPTER

TWO

DESIGN OVERVIEW

The design of the operator is tighly bound to the Percona Server for MongoDB replica set, which is briefly described in the following diagram.

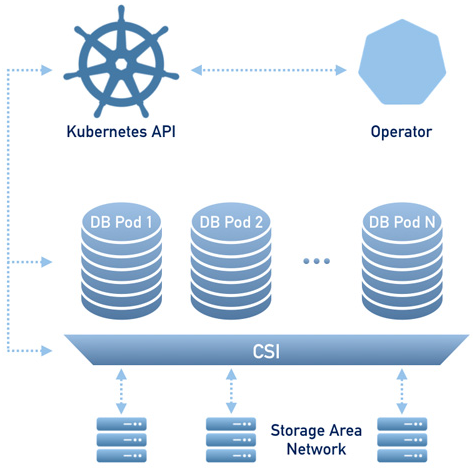


A replica set consists of one primary server and several secondary ones (two in the picture), and the client application accesses the servers via a driver.

To provide high availability the Operator uses node affinity to run MongoDB instances on separate worker nodes if possible, and the database cluster is deployed as a single Replica Set with at least three nodes. If a node fails, the pod with the mongod process is automatically re-created on another node. If the failed node was hosting the primary server, the replica set initiates elections to select a new primary. If the failed node was running the Operator, Kubernetes will restart the Operator on another node, so normal operation will not be interrupted.

Client applications should use a mongo+srv URI for the connection. This allows the drivers (3.6 and up) to retrieve the list of replica set members from DNS SRV entries without having to list hostnames for the dynamically assigned nodes.

Note: The Operator uses security settings which are more secure than the default Percona Server for MongoDB setup. The initial configuration contains default passwords for all needed user accounts, which should be changed in the production environment, as stated in the installation instructions.



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 *PerconaServerMongoDB* object, and it is implemented as a golang application. Each *PerconaServerMongoDB* object maps to one separate Percona Server for MongoDB setup. The Operator listens to all events on the created objects. When a new PerconaServerMongoDB 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 properly operating replica set.

Part II

Quickstart guides

CHAPTER

THREE

INSTALL PERCONA SERVER FOR MONGODB ON MINIKUBE

Installing the Percona Server for MongoDB 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 Kubernetes application locally prior to deploying it on a cloud.

The following steps are needed to run Percona Server for MongoDB Operator on minikube:

0. 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=5120 --cpus=4 --disk-size=30g` (parameters increase the virtual machine limits for the CPU cores, memory, and disk, 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.

1. Clone the percona-server-mongodb-operator repository:

   ```
   git clone -b v1.8.0 https://github.com/percona/percona-server-mongodb-operator
   cd percona-server-mongodb-operator
   ```

2. Deploy the operator with the following command:

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

3. Because minikube runs locally, the default `deploy/cr.yaml` file should be edited to adapt the Operator for the the local installation with limited resources. Change the following keys in the `replsets` section:

   1. comment **all occurrences** of the `resources.requests.memory` and `resources.requests.cpu` keys (this will fit the Operator in minikube default limitations)
   2. set **all occurrences** of the `affinity.antiAffinityTopologyKey` key to `"none"` (the Operator will be unable to spread the cluster on several nodes)

   Also, switch `allowUnsafeConfigurations` key to `true` (this option turns off the Operator's control over the cluster configuration, making it possible to deploy Percona Server for MongoDB as a one-node cluster).

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

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

The creation process may take some time. The process is over when all Pods have reached their Running status. You can check it with the following command:

```
kubectl get pods
```

The result should look as follows:

```
NAME                                               READY   STATUS    RESTARTS   ␣
↪AGE
my-cluster-name-cfg-0                              2/2     Running   0          ␣
↪11m
my-cluster-name-cfg-1                              2/2     Running   1          ␣
↪10m
my-cluster-name-cfg-2                              2/2     Running   1           9m
my-cluster-name-mongos-55659468f7-2kvc2            1/1     Running   0          ␣
↪11m
my-cluster-name-mongos-55659468f7-7jfqc            1/1     Running   0          ␣
↪11m
my-cluster-name-mongos-55659468f7-dfwcj            1/1     Running   0          ␣
↪11m
my-cluster-name-rs0-0                              2/2     Running   0          ␣
↪11m
my-cluster-name-rs0-1                              2/2     Running   0          ␣
↪10m
my-cluster-name-rs0-2                              2/2     Running   0           9m
percona-server-mongodb-operator-6fc78d686d-26hdz   1/1     Running   0          ␣
↪37m
```

5. During previous steps, the Operator has generated several secrets, including the password for the admin user, which you will 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 `my-cluster-name-secrets` name). Then `kubectl get secret my-cluster-name-secrets -o yaml` will return the YAML file with generated secrets, including the `MONGODB_USER_ADMIN` and `MONGODB_USER_ADMIN_PASSWORD` strings, which should look as follows:

```
...
data:
  ...
  MONGODB_USER_ADMIN_PASSWORD: aDAzQ0pCY3NSWEZ2ZUIzS1I=
  MONGODB_USER_ADMIN_USER: dXNlckFkbWlu
```

Here the actual login name and password are base64-encoded, and `echo 'aDAzQ0pCY3NSWEZ2ZUIzS1I=' | base64 --decode` will bring it back to a human-readable form.

6. Check connectivity to a newly created cluster.

First of all, run a container with a MongoDB client and connect its console output to your terminal. The following command will do this, naming the new Pod `percona-client`:

```
kubectl run -i --rm --tty percona-client --image=percona/percona-server-mongodb:4.
↪4.5-7 --restart=Never -- bash -il
```

Executing it may require some time to deploy the correspondent Pod. Now run `mongo` tool in the percona-client command shell using the login (which is `userAdmin`) and password obtained from the secret:

```
mongo "mongodb://userAdmin:userAdminPassword@my-cluster-name-mongos.default.svc.
↪cluster.local/admin?ssl=false"
```

CHAPTER

FOUR

INSTALL PERCONA SERVER FOR MONGODB ON GOOGLE KUBERNETES ENGINE (GKE)

This quickstart shows you how to configure a Percona server for MongoDB 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
```

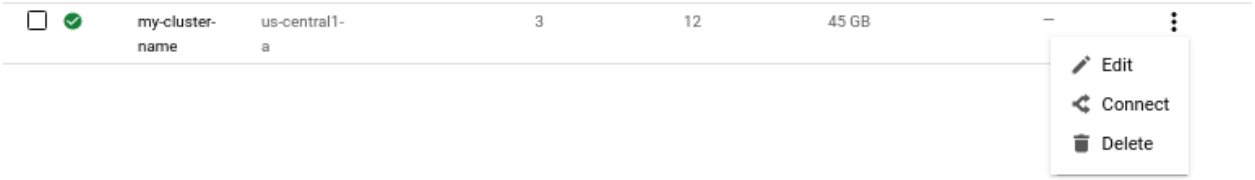
4.2 Configuring default settings for the cluster

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-name`:

```
$ gcloud container clusters create my-cluster-name --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-name --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-server-mongodb-operator repository:

   ```
   git clone -b v1.8.0 https://github.com/percona/percona-server-mongodb-operator
   ```

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

   ```
   cd percona-server-mongodb-operator
   ```

4. Deploy the Operator with the following command:

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

   The following confirmation is returned:

```
customresourcedefinition.apiextensions.k8s.io/perconaservermongodbs.psmdb.percona.
↪com created
customresourcedefinition.apiextensions.k8s.io/perconaservermongodbbackups.psmdb.
↪percona.com created
customresourcedefinition.apiextensions.k8s.io/perconaservermongodbrestores.psmdb.
↪percona.com created
role.rbac.authorization.k8s.io/percona-server-mongodb-operator created
serviceaccount/percona-server-mongodb-operator created
rolebinding.rbac.authorization.k8s.io/service-account-percona-server-mongodb-
↪operator created
deployment.apps/percona-server-mongodb-operator created
```

5. The operator has been started, and you can create the Percona Server for MongoDB:

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

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

```
perconaservermongodb.psmdb.percona.com/my-cluster-name 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 `MONGODB_USER_ADMIN` and `MONGODB_USER_ADMIN_PASSWORD` strings, which should look as follows:

```
...
data:
  ...
  MONGODB_USER_ADMIN_PASSWORD: aDAzQ0pCY3NSWEZ2ZUIzS1I=
  MONGODB_USER_ADMIN_USER: dXNlckFkbWlu
```

   Here the actual password is base64-encoded, and `echo 'aDAzQ0pCY3NSWEZ2ZUIzS1I=' | base64 --decode` will bring it back to a human-readable form.

4.4 Verifying the cluster operation

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

```
$ kubectl get pods
```

The result should look as follows:

```
NAME                                               READY   STATUS    RESTARTS   AGE
my-cluster-name-cfg-0                              2/2     Running   0          11m
my-cluster-name-cfg-1                              2/2     Running   1          10m
my-cluster-name-cfg-2                              2/2     Running   1          9m
my-cluster-name-mongos-55659468f7-2kvc2            1/1     Running   0          11m
my-cluster-name-mongos-55659468f7-7jfqc            1/1     Running   0          11m
my-cluster-name-mongos-55659468f7-dfwcj            1/1     Running   0          11m
my-cluster-name-rs0-0                              2/2     Running   0          11m
my-cluster-name-rs0-1                              2/2     Running   0          10m
```

(continues on next page)

(continued from previous page)

```
my-cluster-name-rs0-2                              2/2     Running   0          9m
percona-server-mongodb-operator-6fc78d686d-26hdz   1/1     Running   0          37m
```

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

Name	Status	Type	Namespace	Cluster
core		API Group		
Pod		Kind		
my-cluster-name-cfg-0	Running	Pod	default	my-cluster-name
my-cluster-name-cfg-1	Running	Pod	default	my-cluster-name
my-cluster-name-cfg-2	Running	Pod	default	my-cluster-name
my-cluster-name-mongos-55659468f7-2kvc2	Running	Pod	default	my-cluster-name
my-cluster-name-mongos-55659468f7-7jfqc	Running	Pod	default	my-cluster-name
my-cluster-name-mongos-55659468f7-dlwcj	Running	Pod	default	my-cluster-name
my-cluster-name-rs0-0	Running	Pod	default	my-cluster-name
my-cluster-name-rs0-1	Running	Pod	default	my-cluster-name
my-cluster-name-rs0-2	Running	Pod	default	my-cluster-name
percona-server-mongodb-operator-6fc78d686d-26hdz	Running	Pod	default	my-cluster-name

If all nodes are up and running, you can try to connect to the cluster.

First of all, run a container with a MongoDB client and connect its console output to your terminal. The following command will do this, naming the new Pod `percona-client`:

```
kubectl run -i --rm --tty percona-client --image=percona/percona-server-mongodb:4.4.5-
↪7 --restart=Never -- bash -il
```

Executing it may require some time to deploy the correspondent Pod. Now run `mongo` tool in the percona-client command shell using the login (which is `userAdmin`) and password obtained from the secret:

```
mongo "mongodb://userAdmin:userAdminPassword@my-cluster-name-mongos.<namespace name>.
↪svc.cluster.local/admin?ssl=false"
```

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 my-cluster-name-rs0-2
```

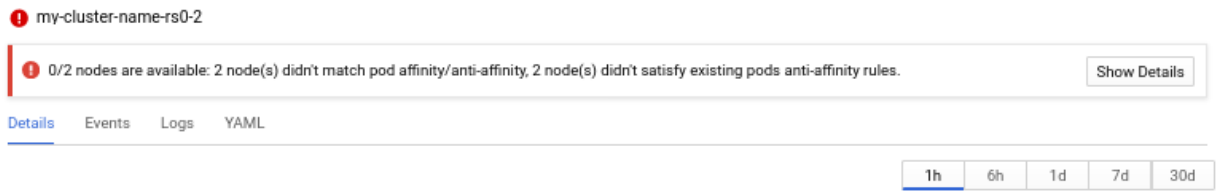
Review the detailed information for `Warning` statements and then correct the configuration. An example of a warning is as follows:

> *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	Namespace	Cluster
core		API Group		
Pod		Kind		
my-cluster-name-cfg-0	Running	Pod	default	my-cluster-name
my-cluster-name-cfg-1	Running	Pod	default	my-cluster-name
my-cluster-name-cfg-2	Running	Pod	default	my-cluster-name
my-cluster-name-mongos-55659468f7-2kvc2	Running	Pod	default	my-cluster-name
my-cluster-name-mongos-55659468f7-7jfqc	Running	Pod	default	my-cluster-name
my-cluster-name-mongos-55659468f7-dlwcj	Running	Pod	default	my-cluster-name
my-cluster-name-rs0-0	Running	Pod	default	my-cluster-name
my-cluster-name-rs0-1	Running	Pod	default	my-cluster-name
my-cluster-name-rs0-2	Unschedulable	Pod	default	my-cluster-name
percona-server-mongodb-operator-6fc78d686d-26hdz	Running	Pod	default	my-cluster-name

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



4.6 Removing the GKE cluster

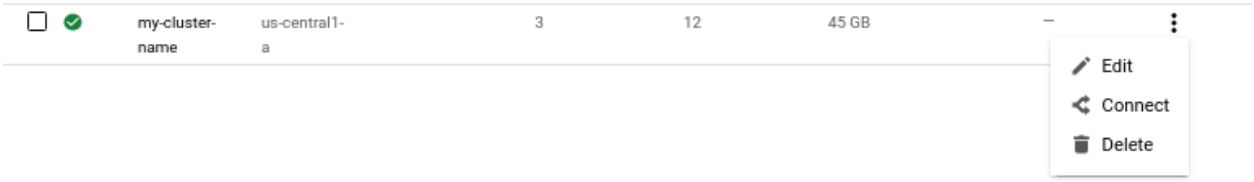
There are several ways that you can delete the cluster.

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 `Delete` popup menu item in the clusters list:



The cluster deletion may take time.

CHAPTER

FIVE

INSTALL PERCONA SERVER FOR MONGODB ON AMAZON ELASTIC KUBERNETES SERVICE (EKS)

This quickstart shows you how to deploy Percona server for MongoDB 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 desired ratio between on-demand and spot instances in the total number of nodes.

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

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-server-mongodb-operator repository:

   ```
   git clone -b v1.8.0 https://github.com/percona/percona-server-mongodb-operator
   ```

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

```
cd percona-server-mongodb-operator
```

3. Deploy the Operator with the following command:

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

The following confirmation is returned:

```
customresourcedefinition.apiextensions.k8s.io/perconaservermongodbs.psmdb.percona.
↪com created
customresourcedefinition.apiextensions.k8s.io/perconaservermongodbbackups.psmdb.
↪percona.com created
customresourcedefinition.apiextensions.k8s.io/perconaservermongodbrestores.psmdb.
↪percona.com created
role.rbac.authorization.k8s.io/percona-server-mongodb-operator created
serviceaccount/percona-server-mongodb-operator created
rolebinding.rbac.authorization.k8s.io/service-account-percona-server-mongodb-
↪operator created
deployment.apps/percona-server-mongodb-operator created
```

4. The operator has been started, and you can create the Percona Server for MongoDB:

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

The creation process may take some time. The process is over when all Pods have reached their Running status. You can check it with the following command:

```
kubectl get pods
```

The result should look as follows:

```
NAME                                               READY   STATUS    RESTARTS   ␣
↪AGE
my-cluster-name-cfg-0                              2/2     Running   0          ␣
↪11m
my-cluster-name-cfg-1                              2/2     Running   1          ␣
↪10m
my-cluster-name-cfg-2                              2/2     Running   1          9m
my-cluster-name-mongos-55659468f7-2kvc2            1/1     Running   0          ␣
↪11m
my-cluster-name-mongos-55659468f7-7jfqc            1/1     Running   0          ␣
↪11m
my-cluster-name-mongos-55659468f7-dfwcj            1/1     Running   0          ␣
↪11m
my-cluster-name-rs0-0                              2/2     Running   0          ␣
↪11m
my-cluster-name-rs0-1                              2/2     Running   0          ␣
↪10m
my-cluster-name-rs0-2                              2/2     Running   0          9m
percona-server-mongodb-operator-6fc78d686d-26hdz   1/1     Running   0          ␣
↪37m
```

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 `MONGODB_USER_ADMIN` and `MONGODB_USER_ADMIN_PASSWORD` strings, which should look as follows:

```
...
data:
  ...
  MONGODB_USER_ADMIN_PASSWORD: aDAzQ0pCY3NSWEZ2ZUIzS1I=
  MONGODB_USER_ADMIN_USER: dXNlckFkbWlu
```

Here the actual password is base64-encoded, and `echo 'aDAzQ0pCY3NSWEZ2ZUIzS1I=' | base64 --decode` will bring it back to a human-readable form.

6. Check connectivity to a newly created cluster.

   First of all, run a container with a MongoDB client and connect its console output to your terminal. The following command will do this, naming the new Pod `percona-client`:

```
kubectl run -i --rm --tty percona-client --image=percona/percona-server-mongodb:4.
↪4.5-7 --restart=Never -- bash -il
```

   Executing it may require some time to deploy the correspondent Pod. Now run `mongo` tool in the percona-client command shell using the login (which is `userAdmin`) and password obtained from the secret:

```
mongo "mongodb://userAdmin:userAdminPassword@my-cluster-name-mongos.<namespace␣
↪name>.svc.cluster.local/admin?ssl=false"
```

Part III

Advanced Installation Guides

CHAPTER

SIX

INSTALL PERCONA SERVER FOR MONGODB ON KUBERNETES

1. Clone the percona-server-mongodb-operator repository:

```
git clone -b v1.8.0 https://github.com/percona/percona-server-mongodb-operator
cd percona-server-mongodb-operator
```

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

2. The Custom Resource Definition for Percona Server for MongoDB should be created from the `deploy/crd.yaml` file. The Custom Resource Definition extends the standard set of resources which Kubernetes "knows" about with the new items, in our case these items are the core of the operator.

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

This step should be done only once; the step does not need to be repeated with any other Operator deployments.

3. 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.

4. The role-based access control (RBAC) for Percona Server for MongoDB is configured with the `deploy/rbac.yaml` file. Role-based access is based on defined roles and the available actions which correspond to each role. The role and actions are defined for Kubernetes resources in the yaml file. Further 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)
```

5. Start the operator within Kubernetes:

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

6. Add the MongoDB Users secrets to Kubernetes. These secrets should be placed as plain text in the stringData section of the `deploy/secrets.yaml` file as login name and passwords for the user accounts (see Kubernetes documentation for details).

   After editing the yaml file, MongoDB Users secrets should be created using the following command:

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

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

7. 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*.

8. After the operator is started, Percona Server for MongoDB cluster can be created with the following command:

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

   The creation process may take some time. The process is over when all Pods have reached their Running status. You can check it with the following command:

```
$ kubectl get pods
```

   The result should look as follows:

```
NAME                                               READY   STATUS    RESTARTS   AGE
my-cluster-name-cfg-0                              2/2     Running   0          11m
my-cluster-name-cfg-1                              2/2     Running   1          10m
my-cluster-name-cfg-2                              2/2     Running   1          9m
my-cluster-name-mongos-55659468f7-2kvc2            1/1     Running   0          11m
my-cluster-name-mongos-55659468f7-7jfqc            1/1     Running   0          11m
my-cluster-name-mongos-55659468f7-dfwcj            1/1     Running   0          11m
my-cluster-name-rs0-0                              2/2     Running   0          11m
my-cluster-name-rs0-1                              2/2     Running   0          10m
my-cluster-name-rs0-2                              2/2     Running   0          9m
percona-server-mongodb-operator-6fc78d686d-26hdz   1/1     Running   0          37m
```

9. Check connectivity to newly created cluster, using the login (which is `userAdmin`) and corresponding password from the secret:

```
$ kubectl run -i --rm --tty percona-client --image=percona/percona-server-
↪mongodb:4.4.5-7 --restart=Never -- bash -il
percona-client:/$ mongo "mongodb://userAdmin:userAdmin123456@my-cluster-name-
↪mongos.<namespace name>.svc.cluster.local/admin?ssl=false"
```

CHAPTER

SEVEN

INSTALL PERCONA SERVER FOR MONGODB ON OPENSHIFT

Installing Percona Server for MongoDB on OpenShift includes two steps:

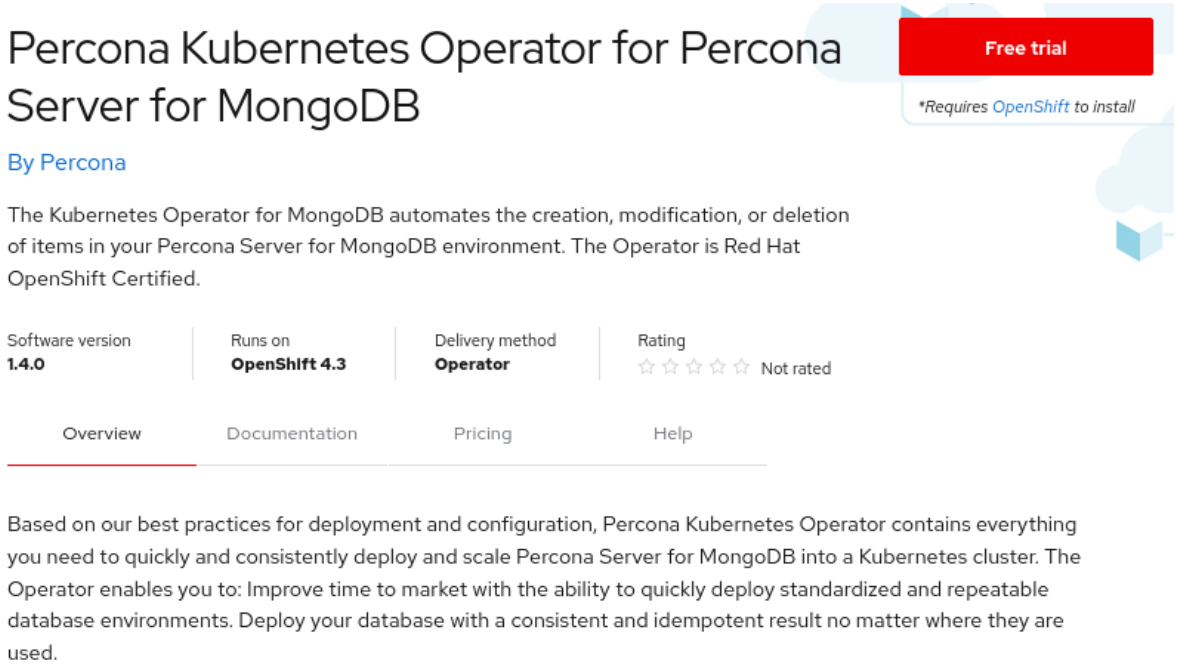
- Installing the Percona Operator for Percona Server for MongoDB,
- Install Percona Server for MongoDB using the Operator.

7.1 Install the Operator

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

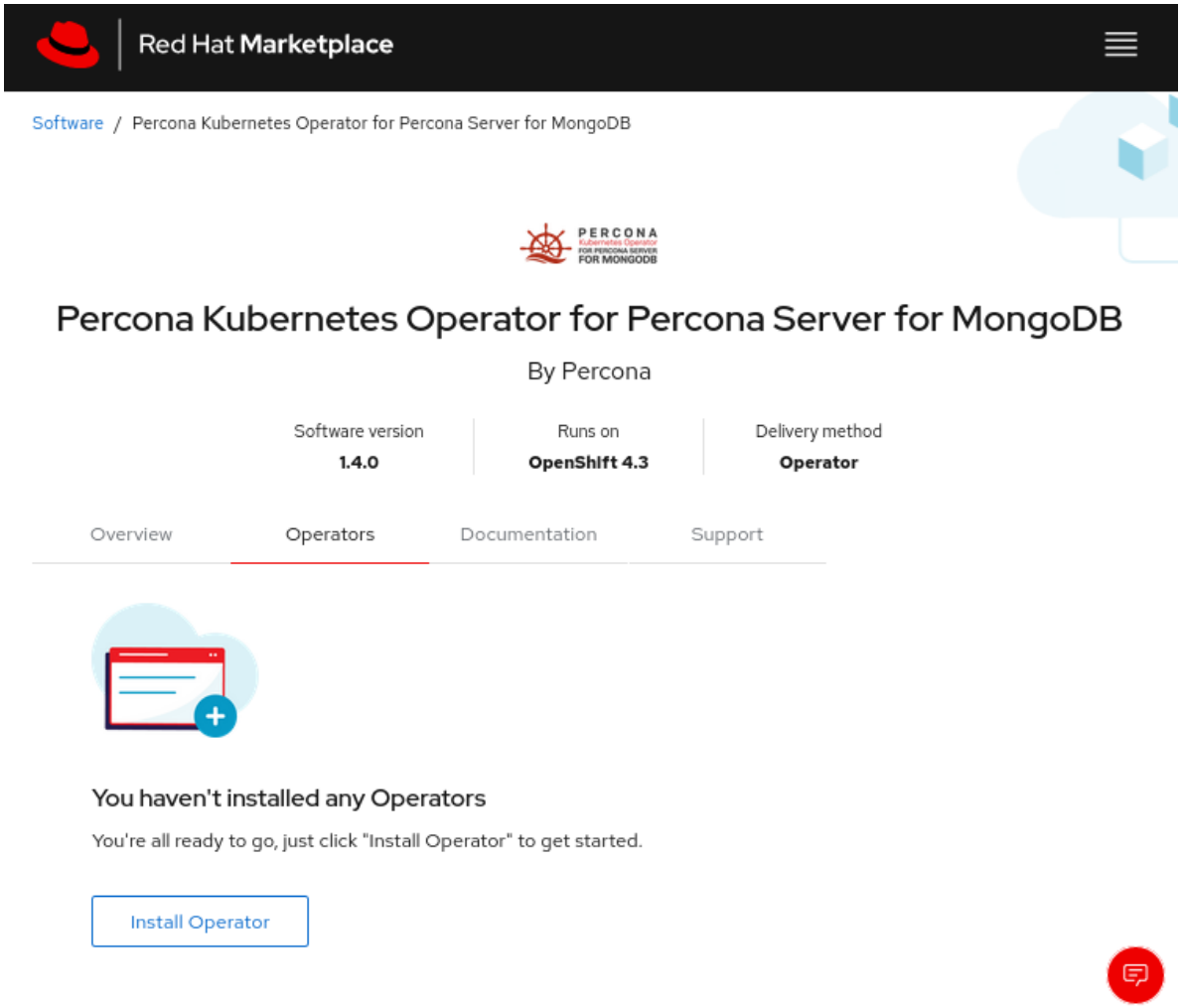
7.1.1 Install the Operator via the Red Hat Marketplace

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



Here you can "purchase" 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-server-mongodb-operator repository:

```
git clone -b v1.8.0 https://github.com/percona/percona-server-mongodb-operator
cd percona-server-mongodb-operator
```

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

2. The Custom Resource Definition for Percona Server for MongoDB should be created from the `deploy/crd.yaml` file. The Custom Resource Definition extends the standard set of resources which Kubernetes "knows" about with the new items, in our case these items are the core of the operator.

   This step should be done only once; it does not need to be repeated with other deployments.

```
$ 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 Percona Server for MongoDB cluster with a non-privileged user, the necessary permissions can be granted by applying the next clusterrole:

```
$ oc create clusterrole psmdb-admin --verb="*" --resource=perconaservermongodbs.
↪psmdb.percona.com,perconaservermongodbs.psmdb.percona.com/status,
↪perconaservermongodbbackups.psmdb.percona.com,perconaservermongodbbackups.psmdb.
↪percona.com/status,perconaservermongodbrestores.psmdb.percona.com,
↪perconaservermongodbrestores.psmdb.percona.com/status
$ oc adm policy add-cluster-role-to-user psmdb-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=iissuers.certmanager.k8s.
↪io,certificates.certmanager.k8s.io
$ oc adm policy add-cluster-role-to-user cert-admin <some-user>
```

3. Create a new `psmdb` project:

```
$ oc new-project psmdb
```

4. Add role-based access control (RBAC) for Percona Server for MongoDB is configured with the `deploy/rbac.yaml` file. RBAC is based on clearly defined roles and corresponding allowed actions. These actions are allowed on specific Kubernetes resources. The details about users and roles can be found in OpenShift documentation.

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

5. Start the Operator within OpenShift:

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

7.2 Install Percona Server for MongoDB

1. Add the MongoDB Users secrets to OpenShift. These secrets should be placed as plain text in the stringData section of the `deploy/secrets.yaml` file as login name and passwords for the user accounts (see Kubernetes documentation for details).

   After editing the yaml file, the secrets should be created with 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. Percona Server for MongoDB cluster can be created at any time with the following two steps:

a. Uncomment the `deploy/cr.yaml` field `#platform:` and edit the field to `platform: openshift`. The result should be like this:

```
apiVersion: psmdb.percona.com/v1alpha1
kind: PerconaServerMongoDB
metadata:
  name: my-cluster-name
spec:
  platform: openshift
...
```

b. (optional) In you're using minishift, please adjust antiaffinity policy to `none`

```
   affinity:
     antiAffinityTopologyKey: "none"
...
```

c. Create/apply the CR file:

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

The creation process will take time. The process is complete when all Pods have reached their Running status. You can check it with the following command:

```
$ oc get pods
```

The result should look as follows:

```
NAME                                               READY   STATUS    RESTARTS   AGE
my-cluster-name-cfg-0                              2/2     Running   0          11m
my-cluster-name-cfg-1                              2/2     Running   1          10m
my-cluster-name-cfg-2                              2/2     Running   1          9m
my-cluster-name-mongos-55659468f7-2kvc2            1/1     Running   0          11m
my-cluster-name-mongos-55659468f7-7jfqc            1/1     Running   0          11m
my-cluster-name-mongos-55659468f7-dfwcj            1/1     Running   0          11m
my-cluster-name-rs0-0                              2/2     Running   0          11m
my-cluster-name-rs0-1                              2/2     Running   0          10m
my-cluster-name-rs0-2                              2/2     Running   0          9m
percona-server-mongodb-operator-6fc78d686d-26hdz   1/1     Running   0          37m
```

4. Check connectivity to newly created cluster. Please note that mongo client command shall be executed inside the container manually.

```
$ oc run -i --rm --tty percona-client --image=percona/percona-server-mongodb:4.4.5-7 --restart=Never -- bash -il
percona-client:/$ mongo "mongodb://userAdmin:userAdmin123456@my-cluster-name-mongos.psmdb.svc.cluster.local/admin?ssl=false"
```

CHAPTER

EIGHT

USE DOCKER IMAGES FROM A CUSTOM REGISTRY

Using images from a private Docker registry may required for privacy, security or other reasons. In these cases, Percona Server for MongoDB Operator allows the use of a custom registry This following example of the Operator deployed in the OpenShift environment demonstrates the process:

1. Log into the OpenShift and create a project.

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

2. You need obtain the following objects to configure your custom registry access:

   - A user token
   - the registry IP address

   You can view the token with the following command:

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

   The following command returns 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. Use the user token and the registry IP address to login to the registry:

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

4. Use the Docker commands to pull the needed image by its SHA digest:

```
$ docker pull docker.io/perconalab/percona-server-
↪mongodb@sha256:991d6049059e5eb1a74981290d829a5fb4ab0554993748fde1e67b2f46f26bf0
Trying to pull repository docker.io/perconalab/percona-server-mongodb ...
sha256:991d6049059e5eb1a74981290d829a5fb4ab0554993748fde1e67b2f46f26bf0: Pulling␣
↪from docker.io/perconalab/percona-server-mongodb
```

(continues on next page)

(continued from previous page)

```
Digest: sha256:991d6049059e5eb1a74981290d829a5fb4ab0554993748fde1e67b2f46f26bf0
Status: Image is up to date for docker.io/perconalab/percona-server-
↪mongodb@sha256:991d6049059e5eb1a74981290d829a5fb4ab0554993748fde1e67b2f46f26bf0
```

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

5. The following method can push an image to the custom registry for the example OpenShift `psmdb` project:

```
$ docker tag \
    docker.io/perconalab/percona-server-
↪mongodb@sha256:991d6049059e5eb1a74981290d829a5fb4ab0554993748fde1e67b2f46f26bf0␣
↪\
    172.30.162.173:5000/psmdb/percona-server-mongodb:4.4.5-7
$ docker push 172.30.162.173:5000/psmdb/percona-server-mongodb:4.4.5-7
```

6. Verify the image is available in the OpenShift registry with the following command:

```
$ oc get is
NAME                              DOCKER REPO                                    ␣
↪                          TAGS              UPDATED
percona-server-mongodb            docker-registry.default.svc:5000/psmdb/percona-
↪server-mongodb  4.4.5-7  2 hours ago
```

7. When the custom registry image is available, edit the the `image:` option in `deploy/operator.yaml` configuration file with a Docker Repo + Tag string (it should look like``docker-registry.default.svc:5000/psmdb/percona-server-mongodb:4.4.5-7``)

Note: If the registry requires authentication, you can specify the `imagePullSecrets` option for all images.

8. Repeat steps 3-5 for other images, and update corresponding options in the `deploy/cr.yaml` file.
9. Now follow the standard Percona Server for MongoDB Operator installation instruction

CHAPTER

NINE

DEPLOY PERCONA SERVER FOR MONGODB 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 Server for MongoDB 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 as follows:

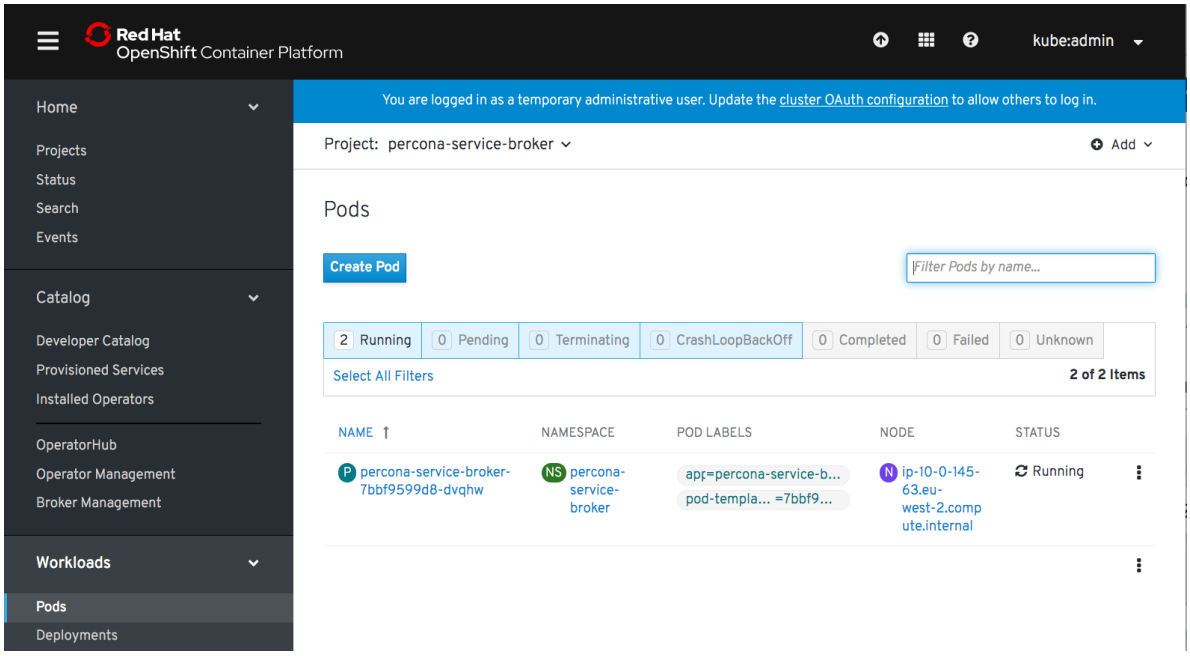
   ```
   $ oc patch servicecatalogapiservers cluster --patch '{"spec":{"managementState":
   ↪"Managed"}}' --type=merge
   $ oc patch servicecatalogcontrollermanagers cluster --patch '{"spec":{
   ↪"managementState":"Managed"}}' --type=merge
   ```

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

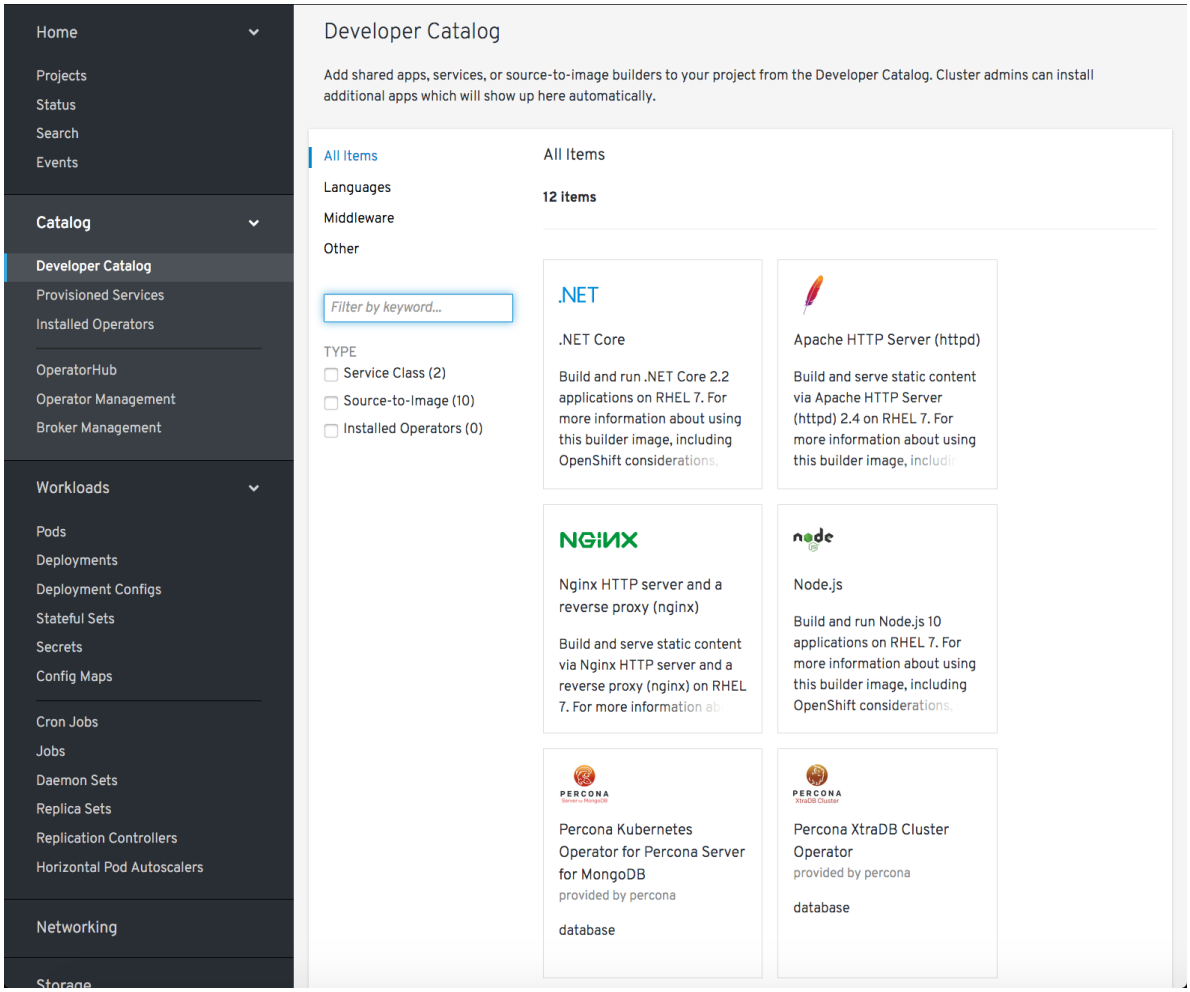
   ```
   $ oc apply -f https://raw.githubusercontent.com/Percona-Lab/percona-dbaas-cli/
   ↪broker/deploy/percona-broker.yaml
   ```

   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:

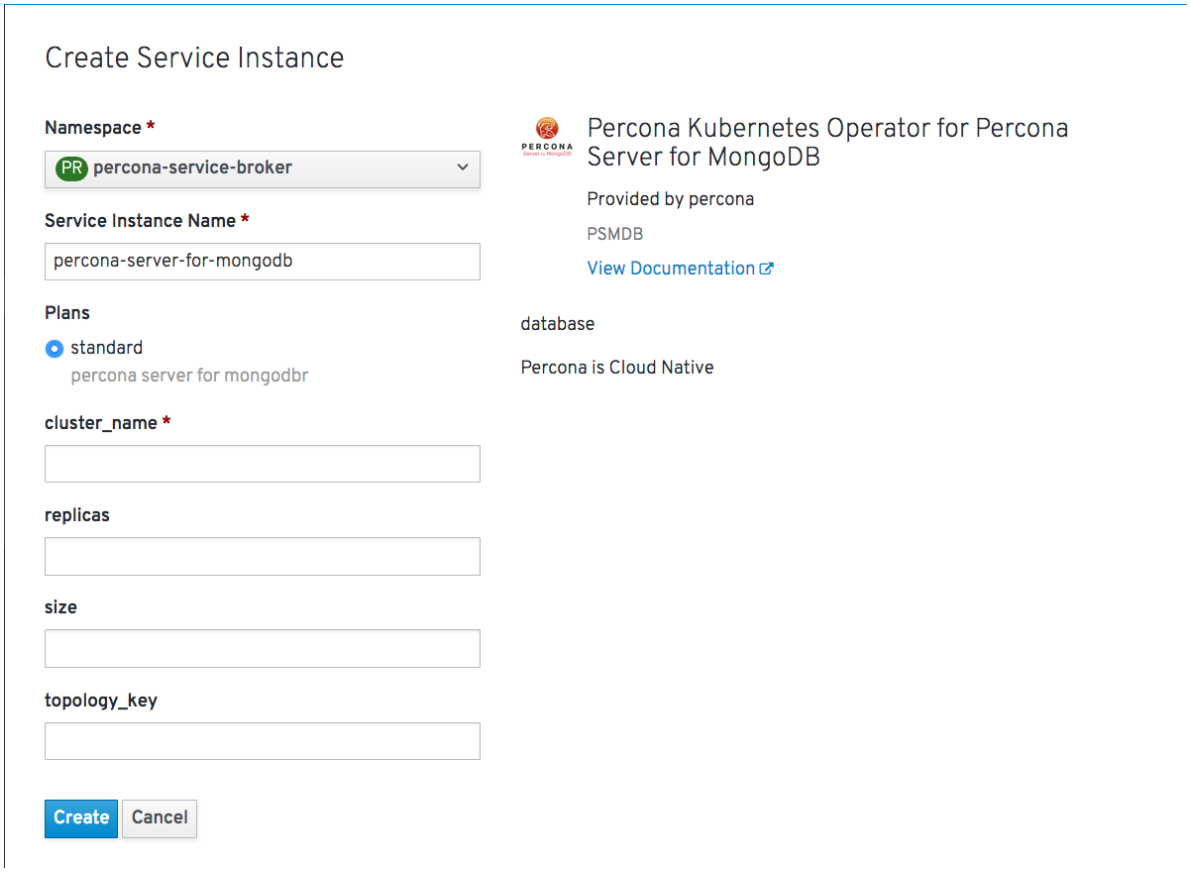


Now switch to the Developer Catalog and select Percona Kubernetes Operator for MongoDB:



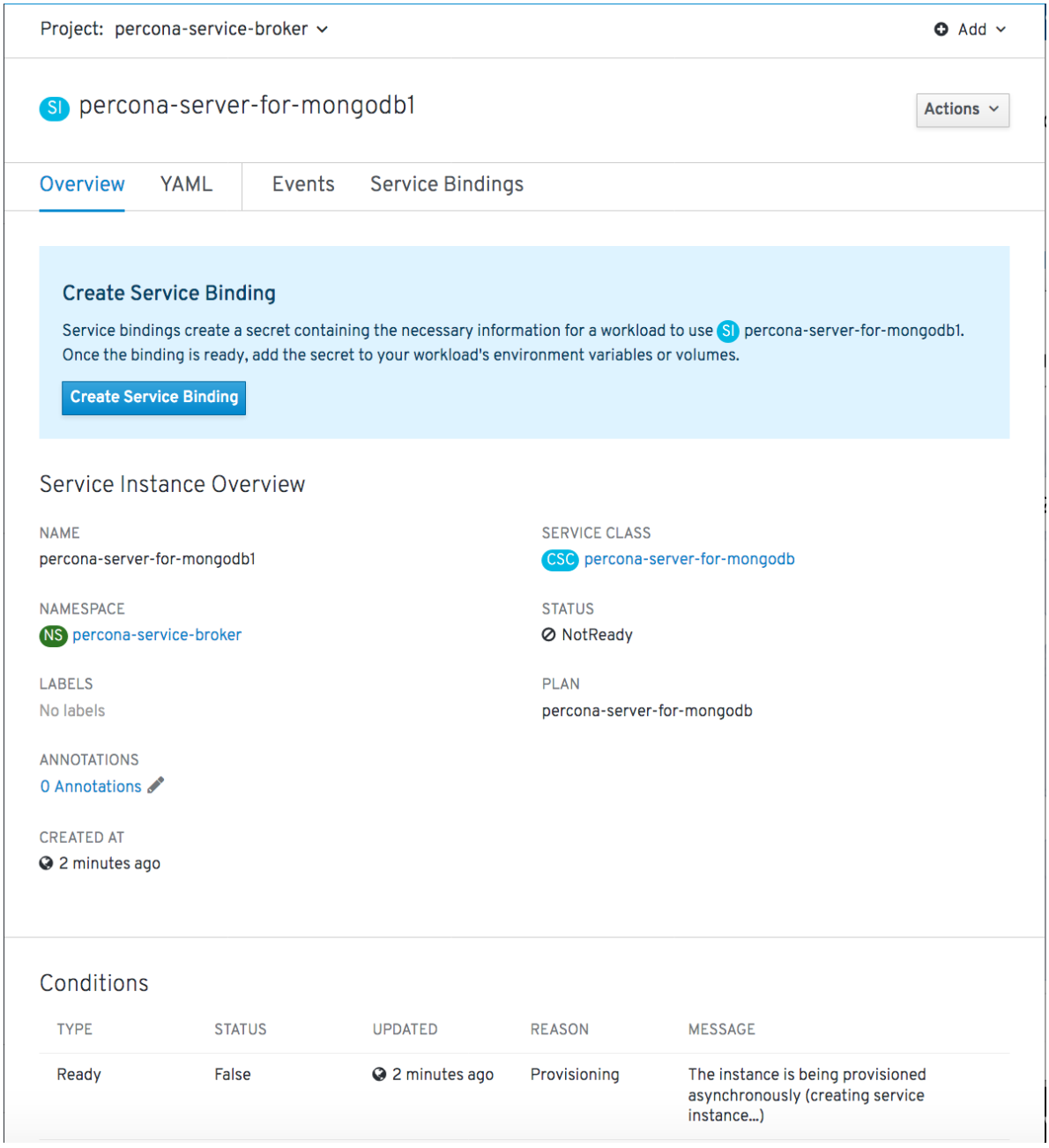
Choose `Percona Kubernetes Operator for Percona Server for MongoDB` item. This will lead you to the Operator page with the *Create Service Instance* button.

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 SERVER FOR MONGODB 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 Percona Operator for Percona Server for MongoDB:

```
helm install my-op percona/psmdb-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 Server for MongoDB:

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

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

10.3 Installing Percona Server for MongoDB with customized parameters

The command above installs Percona Server for MongoDB 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 Server for MongoDB Cluster in the `psmdb` namespace, with disabled backups and 20 Gi storage:

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

Part IV

Configuration

CHAPTER

ELEVEN

USERS

MongoDB 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 MongoDB Health checks.

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

11.1 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 -i --rm --tty percona-client --image=percona/percona-server-mongodb:4.4.
↪5-7 --restart=Never -- bash -il
mongodb@percona-client:/$ mongo "mongodb+srv://userAdmin:userAdmin123456@my-cluster-
↪name-rs0.psmdb.svc.cluster.local/admin?replicaSet=rs0&ssl=false"
rs0:PRIMARY> db.createUser({
    user: "myApp",
    pwd: "myAppPassword",
    roles: [
      { db: "myApp", role: "readWrite" }
    ],
    mechanisms: [
       "SCRAM-SHA-1"
    ]

})
```

Now check the newly created user:

```
$ kubectl run -i --rm --tty percona-client --image=percona/percona-server-mongodb:4.4.
↪5-7 --restart=Never -- bash -il
mongodb@percona-client:/$ mongo "mongodb+srv://myApp:myAppPassword@my-cluster-name-
↪rs0.psmdb.svc.cluster.local/admin?replicaSet=rs0&ssl=false"
rs0:PRIMARY> use myApp
rs0:PRIMARY> db.test.insert({ x: 1 })
rs0:PRIMARY> db.test.findOne()
```

11.2 System Users

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

During installation, the Operator requires Kubernetes Secrets to be deployed before the Operator is started. The name of the required secrets can be set in `deploy/cr.yaml` under the `spec.secrets` section.

Default Secret name: `my-cluster-name-secrets`

Secret name field: `spec.secrets.users`

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

User Purpose	Username Secret Key	Password Secret Key
Backup/Restore	MONGODB_BACKUP_USER	MONGODB_BACKUP_PASSWORD
Cluster Admin	MONGODB_CLUSTER_ADMIN_USER	MONGODB_CLUSTER_ADMIN_PASSWORD
Cluster Monitor	MONGODB_CLUSTER_MONITOR_USER	MONGODB_CLUSTER_MONITOR_PASSWORD
User Admin	MONGODB_USER_ADMIN_USER	MONGODB_USER_ADMIN_PASSWORD
PMM Server	PMM_SERVER_USER	PMM_SERVER_PASSWORD

Backup/Restore - MongoDB Role: backup, clusterMonitor, restore

Cluster Admin - MongoDB Role: clusterAdmin

Cluster Monitor - MongoDB Role: clusterMonitor

User Admin - MongoDB Role: userAdmin

Note: If you change credentials for the `MONGODB_CLUSTER_MONITOR` user, the cluster Pods will go into restart cycle, and the cluster can be not accessible through the `mongos` service until this cycle finishes.

11.2.1 YAML Object Format

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

```
apiVersion: v1
kind: Secret
metadata:
  name: my-cluster-name-secrets
type: Opaque
stringData:
  MONGODB_BACKUP_USER: backup
  MONGODB_BACKUP_PASSWORD: backup123456
  MONGODB_CLUSTER_ADMIN_USER: clusterAdmin
  MONGODB_CLUSTER_ADMIN_PASSWORD: clusterAdmin123456
  MONGODB_CLUSTER_MONITOR_USER: clusterMonitor
  MONGODB_CLUSTER_MONITOR_PASSWORD: clusterMonitor123456
  MONGODB_USER_ADMIN_USER: userAdmin
  MONGODB_USER_ADMIN_PASSWORD: userAdmin123456
  PMM_SERVER_USER: pmm
  PMM_SERVER_PASSWORD: supa|^|pazz
```

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":{"PMM_SERVER_PASSWORD": "'
↪$(echo -n new_password | base64)'"}}'
```

Note: The operator creates and updates an additional Secrets object named based on the cluster name, like `internal-my-cluster-name-users`. It is used only by the Operator and should undergo no manual changes by the user. This object contains secrets with the same passwords as the one specified in `spec.secrets.users` (e.g. `my-cluster-name-secrets`). When the user updates `my-cluster-name-secrets`, the Operator propagates these changes to the internal `internal-my-cluster-name-users` Secrets object.

11.2.2 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 `secrets.users` option in CR, make changes inside the secrets object itself.

11.3 Development Mode

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

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

Secret Key	Secret Value
MONGODB_BACKUP_USER	backup
MONGODB_BACKUP_PASSWORD	backup123456
MONGODB_CLUSTER_ADMIN_USER	clusterAdmin
MONGODB_CLUSTER_ADMIN_PASSWORD	clusterAdmin123456
MONGODB_CLUSTER_MONITOR_USER	clusterMonitor
MONGODB_CLUSTER_MONITOR_PASSWORD	clusterMonitor123456
MONGODB_USER_ADMIN_USER	userAdmin
MONGODB_USER_ADMIN_PASSWORD	userAdmin123456
PMM_SERVER_USER	pmm
PMM_SERVER_PASSWORD	supa|^|pazz

Warning: Do not use the default MongoDB Users in production!

11.4 MongoDB Internal Authentication Key (optional)

Default Secret name: `my-cluster-name-mongodb-key`

Secret name field: `spec.secrets.key`

By default, the operator will create a random, 1024-byte key for MongoDB Internal Authentication if it does not already exist. If you would like to deploy a different key, create the secret manually before starting the operator.

CHAPTER

TWELVE

LOCAL STORAGE SUPPORT FOR THE PERCONA SERVER FOR MONGODB OPERATOR

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

12.1 emptyDir

A Pod emptyDir volume is created when the Pod is assigned to a Node. The volume is initially empty and is erased when the Pod is removed from the Node. The containers in the Pod can read and write the files in the emptyDir volume.

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

The `emptyDir` is useful when you use Percona Memory Engine.

12.2 hostPath

A hostPath volume mounts an existing file or directory from the host node's filesystem into the Pod. If the pod is removed, the data persists in the host node's filesystem.

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, you must created the hostPath manually and should have following attributes:

- access permissions
- ownership
- SELinux security context

The `hostPath` volume is useful when you perform manual actions during the first run and require improved disk performance. Consider using the tolerations settings to avoid a 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

THIRTEEN

BINDING PERCONA SERVER FOR MONGODB COMPONENTS TO SPECIFIC KUBERNETES/OPENSHIFT NODES

The operator does a good job of automatically assigning new pods to nodes to achieve balanced distribution across the cluster. There are situations when you must ensure that pods land on specific nodes: for example, for the advantage of speed on an SSD-equipped machine, or reduce costs by choosing nodes in the same availability zone.

The appropriate (sub)sections (`replsets`, `replsets.arbiter`, `backup`, etc.) of the deploy/cr.yaml file contain the keys which can be used to do assign pods to nodes.

13.1 Node selector

The `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
```

13.2 Affinity and anti-affinity

Affinity defines eligible pods that can be scheduled on the node which already has pods with specific labels. Anti-affinity defines pods that are not eligible. This approach is reduces costs by ensuring several pods with intensive data exchange occupy the same availability zone or even the same node or, on the contrary, to spread the pods on different nodes or even different availability zones for high availability and balancing purposes.

Percona Server for MongoDB 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.

13.2.1 Simple approach - use antiAffinityTopologyKey of the Percona Server for MongoDB Operator

Percona Server for MongoDB Operator provides an `antiAffinityTopologyKey` 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 Server for MongoDB Pods to avoid occupying the same node:

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

13.2.2 Advanced approach - use standard Kubernetes constraints

The previous method can be used without 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, the `advanced` option placed in the deploy/cr.yaml file turns off the effect of the `antiAffinityTopologyKey` and allows the use of the 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:
             - e2e-az1
             - e2e-az2
```

(continues on next page)

(continued from previous page)

```
      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.

13.3 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 equal variant requires a corresponding `value` for comparison).

Toleration should have a specified `effect`, such as the following:

- `NoSchedule` - less strict
- `PreferNoSchedule`
- `NoExecute`

When a *taint* with the `NoExecute` effect is assigned to a node, any pod configured to not tolerating this *taint* is removed from the node. This removal can be immediate or after the `tolerationSeconds` interval. The following example defines this effect and the removal interval:

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

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

13.4 Priority Classes

Pods may belong to some *priority classes*. This flexibility allows the scheduler to distinguish more and less important Pods when needed, such as the situation when a higher priority Pod cannot be scheduled without evicting a lower priority one. This ability can be accomplished by 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.

13.5 Pod Disruption Budgets

Creating the Pod Disruption Budget is the Kubernetes method to limit the number of Pods of an application that can go down simultaneously due to *voluntary disruptions* such as the cluster administrator's actions during a deployment update. Distribution Budgets allow large applications to retain their high availability during maintenance and other administrative activities. The `maxUnavailable` and `minAvailable` options in the deploy/cr.yaml file can be used to set these limits. The recommended variant is the following:

```
podDisruptionBudget:
   maxUnavailable: 1
```

CHAPTER

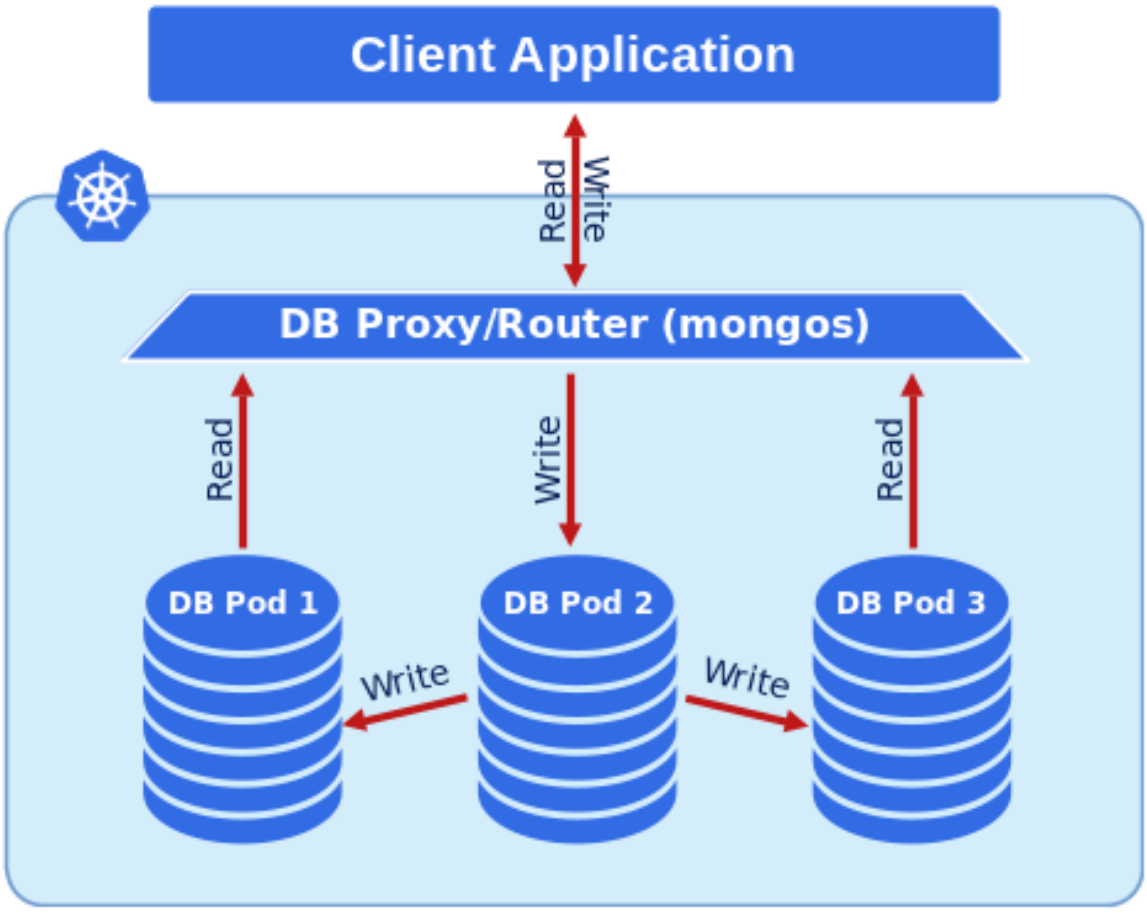
FOURTEEN

EXPOSING CLUSTER NODES WITH DEDICATED IP ADDRESSES

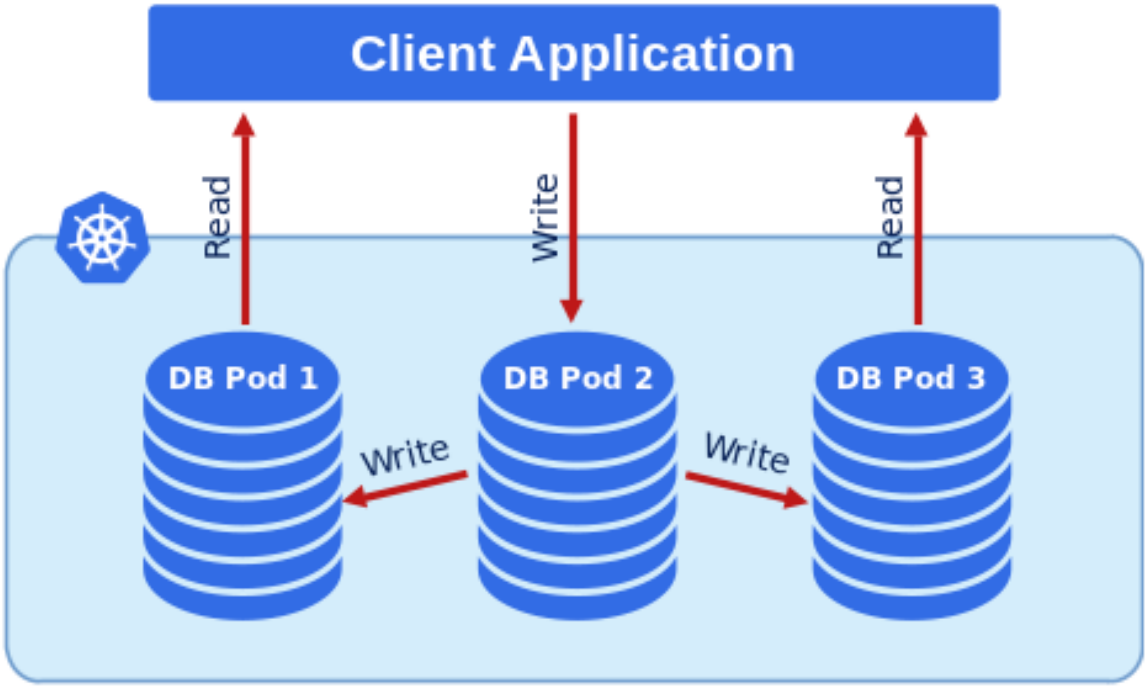
14.1 Using single entry point vs. accessing MongoDB Instances

Percona Operator for Percona Server for MongoDB provides two scenarios for accessing the database.

1. If *Percona Server for MongoDB Sharding* mode is turned **on** (default bahaviour), then database cluster runs special `mongos` Pods - query routers, which acts as an entry point for client applications,



2. If *Percona Server for MongoDB Sharding* mode is turned **off**, the application needs access to all MongoDB Pods of the replica set:



You can find more on sharding in the official MongoDB documentation.

14.2 Accessing the Pod

When Kubernetes creates Pods, each Pod has an IP address in the internal virtual network of the cluster. Creating and destroying Pods is a dynamic process, therefore binding communication between Pods to specific IP addresses would cause problems as things change over time as a result of the cluster scaling, maintenance, etc. Due to this changing environment, you should connect to Percona Server for MongoDB via Kubernetes internal DNS names in URI (e.g. using `mongodb+srv://userAdmin:userAdmin123456@<cluster-name>-rs0.<namespace>.svc.cluster.local/admin?replicaSet=rs0&ssl=false` to access one of the Replica Set Pods). URI-based access is strictly recommended.

Sometimes you cannot communicate with the Pods using the Kubernetes internal DNS names. To make Pods of the Replica Set accessible, Percona Server for MongoDB Operator can assign a Kubernetes Service to each Pod.

This feature can be configured in the `replsets` (for MondgoDB instances Pod) and `sharding` (for mongos Pod) sections of the deploy/cr.yaml file:

- set 'expose.enabled' option to 'true' to allow exposing Pods via services,
- set 'expose.exposeType' option specifying the IP address type to be used:
  - `ClusterIP` - expose the Pod's service with an internal static IP address. This variant makes MongoDB Pod only reachable from within the Kubernetes cluster.
  - `NodePort` - expose the Pod's service on each Kubernetes node's IP address at a static port. ClusterIP service, to which the node port will be routed, is automatically created in this variant. As an advantage, the service will be reachable from outside the cluster by node address and port number, but the address will be bound to a specific Kubernetes node.
  - `LoadBalancer` - expose the Pod's service externally using a cloud provider's load balancer. Both ClusterIP and NodePort services are automatically created in this variant.

If this feature is enabled, URI looks like `mongodb://userAdmin:userAdmin123456@<ip1>:<port1>,<ip2>:<port2>,<ip3>:<port3>/admin?replicaSet=rs0&ssl=false` All IP adresses should be *directly* reachable by application.

CHAPTER

FIFTEEN

ENABLING REPLICA SET ARBITER NODES

Percona Server for MongoDB replication model is based on elections, when nodes of the Replica Set choose which node becomes the primary node. Elections are the reason to avoid an even number of nodes in the cluster. The cluster should have at least three nodes. Normally, each node stores a complete copy of the data, but there is also a possibility, to reduce disk IO and space used by the database, to add an arbiter node. An arbiter cannot become a primary and does not have a complete copy of the data. The arbiter does have one election vote and can be the odd number for elections. The arbiter does not demand a persistent volume.

Percona Server for MongoDB Operator has the ability to create Replica Set Arbiter nodes if needed. This feature can be configured in the Replica Set section of the deploy/cr.yaml file:

- set `arbiter.enabled` option to `true` to allow Arbiter nodes,
- use `arbiter.size` option to set the desired amount of the Replica Set nodes which should be Arbiter ones instead of containing data.

For example, the following keys in `deploy/cr.yaml` will create a cluster with 2 data instances and 1 Arbiter:

```
....
replsets:
  ....
  size: 3
  ....
  arbiter:
    enabled: true
    size: 1
    ....
```

CHAPTER

SIXTEEN

PERCONA SERVER FOR MONGODB SHARDING

16.1 About sharding

Sharding provides horizontal database scaling, distributing data across multiple MongoDB Pods. It is useful for large data sets when a single machine's overall processing speed or storage capacity turns out to be not enough. Sharding allows splitting data across several machines with a special routing of each request to the necessary subset of data (so-called *shard*).

A MongoDB Sharding involves the following components:

- `shard` - a replica set which contains a subset of data stored in the database (similar to a traditional MongoDB replica set),
- `mongos` - a query router, which acts as an entry point for client applications,
- `config servers` - a replica set to store metadata and configuration settings for the sharded database cluster.

Note: Percona Server for MongoDB Operator 1.6.0 supported only one shard of a MongoDB cluster; still, this limited sharding support allowed using `mongos` as an entry point instead of provisioning a load-balancer per replica set node. Multiple shards are supported starting from the Operator 1.7.0.

16.2 Turning sharding on and off

Sharding is controlled by the `sharding` section of the `deploy/cr.yaml` configuration file and is turned on by default.

To enable sharding, set the `sharding.enabled` key to `true` (this will turn existing MongoDB replica set nodes into sharded ones). To disable sharding, set the `sharding.enabled` key to `false`.

When sharding is turned on, the Operator runs replica sets with config servers and mongos instances. Their number is controlled by `configsvrReplSet.size` and `mongos.size` keys, respectively.

Note: Config servers for now can properly work only with WiredTiger engine, and sharded MongoDB nodes can use either WiredTiger or InMemory one.

By default *replsets section* of the `deploy/cr.yaml` configuration file contains only one replica set, `rs0`. You can add more replica sets with different names to the `replsets` section in a similar way. Please take into account that having more than one replica set is possible only with the sharding turned on.

16.3 Checking connectivity to sharded and non-sharded cluster

With sharding turned on, you have `mongos` service as an entry point to access your database. If you do not use sharding, you have to access `mongod` processes of your replica set.

1. Run percona-client and connect its console output to your terminal (running it may require some time to deploy the corresponding Pod):

```
kubectl run -i --rm --tty percona-client --image=percona/percona-server-mongodb:4.
↪4.5-7 --restart=Never -- bash -il
```

2. Find the password for the admin user, which you will 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 `my-cluster-name-secrets` name). Then `kubectl get secret my-cluster-name-secrets -o yaml` will return the YAML file with generated secrets, including the `MONGODB_USER_ADMIN` and `MONGODB_USER_ADMIN_PASSWORD` strings:

```
...
data:
  ...
  MONGODB_USER_ADMIN_PASSWORD: aDAzQ0pCY3NSWEZ2ZUIzS1I=
  MONGODB_USER_ADMIN_USER: dXNlckFkbWlu
```

   Here the actual login name and password are base64-encoded, and `echo 'aDAzQ0pCY3NSWEZ2ZUIzS1I=' | base64 --decode` will bring it back to a human-readable form.

3. Now run `mongo` tool in the percona-client command shell using the login (which is normally `userAdmin`) and password obtained from the secret.

   - If sharding is turned on, the command will look as follows (with your database cluster namespace instead of the `<namespace name>` placeholder).

```
mongo "mongodb://userAdmin:userAdminPassword@my-cluster-name-mongos.
↪<namespace name>.svc.cluster.local/admin?ssl=false"
```

   - If sharding is turned off, the command will look as follows (with your database cluster namespace instead of the `<namespace name>` placeholder).

```
mongo "mongodb+srv://userAdmin:userAdminPassword@my-cluster-name-rs0.
↪<namespace name>.svc.cluster.local/admin?replicaSet=rs0&ssl=false"
```

CHAPTER

SEVENTEEN

TRANSPORT LAYER SECURITY (TLS)

The Percona Kubernetes Operator for Percona Server for MongoDB uses Transport Layer Security (TLS) cryptographic protocol for the following types of communication:

- Internal - communication between Percona Server for MongoDB instances in the cluster
- External - communication between the client application and the cluster

The internal certificate is also used as an authorization method.

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

- The Operator can use a specifically installed *cert-manager* for the automatic certificates generation,
- 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.

17.1 Install and use the *cert-manager*

17.1.1 About the *cert-manager*

A *cert-manager* is a Kubernetes certificate management controller which 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 Server for MongoDB requires all certificates are issued by the same CA.

The creation of the self-signed issuer allows you to deploy and use the Percona Operator without creating a clusterissuer separately.

17.1.2 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 apply -f https://github.com/jetstack/cert-manager/releases/download/v0.15.1/
↪cert-manager.yaml --validate=false
```

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.

17.2 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`.

Supposing that your cluster name is `my-cluster-name-rs0`, the instructions to generate certificates manually are as follows:

```
CLUSTER_NAME=my-cluster-name
NAMESPACE=default
cat <<EOF | cfssl gencert -initca - | cfssljson -bare ca
  {
    "CN": "Root CA",
    "names": [
      {
        "O": "PSMDB"
      }
    ],
```

(continues on next page)

(continued from previous page)

```
    "key": {
      "algo": "rsa",
      "size": 2048
    }
  }
EOF

cat <<EOF > ca-config.json
  {
    "signing": {
      "default": {
        "expiry": "87600h",
        "usages": ["signing", "key encipherment", "server auth", "client auth"]
      }
    }
  }
EOF

cat <<EOF | cfssl gencert -ca=ca.pem  -ca-key=ca-key.pem -config=./ca-config.json - |␣
↪cfssljson -bare server
  {
    "hosts": [
      "localhost",
      "${CLUSTER_NAME}-rs0",
      "${CLUSTER_NAME}-rs0.${NAMESPACE}",
      "${CLUSTER_NAME}-rs0.${NAMESPACE}.svc.cluster.local",
      "*.${CLUSTER_NAME}-rs0",
      "*.${CLUSTER_NAME}-rs0.${NAMESPACE}",
      "*.${CLUSTER_NAME}-rs0.${NAMESPACE}.svc.cluster.local"
    ],
    "names": [
      {
        "O": "PSMDB"
      }
    ],
    "CN": "${CLUSTER_NAME/-rs0}",
    "key": {
      "algo": "rsa",
      "size": 2048
    }
  }
EOF
cfssl bundle -ca-bundle=ca.pem -cert=server.pem | cfssljson -bare server

kubectl create secret generic my-cluster-name-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

cat <<EOF | cfssl gencert -ca=ca.pem  -ca-key=ca-key.pem -config=./ca-config.json - |␣
↪cfssljson -bare client
  {
    "hosts": [
      "${CLUSTER_NAME}-rs0",
      "${CLUSTER_NAME}-rs0.${NAMESPACE}",
      "${CLUSTER_NAME}-rs0.${NAMESPACE}.svc.cluster.local",
      "*.${CLUSTER_NAME}-rs0",
      "*.${CLUSTER_NAME}-rs0.${NAMESPACE}",
```

(continues on next page)

(continued from previous page)

```
      "*.${CLUSTER_NAME}-rs0.${NAMESPACE}.svc.cluster.local"
    ],
    "names": [
      {
        "O": "PSMDB"
      }
    ],
    "CN": "${CLUSTER_NAME/-rs0}",
    "key": {
      "algo": "rsa",
      "size": 2048
    }
  }
EOF

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

17.3 Run Percona Server for MongoDB 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

EIGHTEEN

DATA AT REST ENCRYPTION

Data at rest encryption in Percona Server for MongoDB is supported by the Operator since version 1.1.0.

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

Following options the `mongod` section of the `deploy/cr.yaml` file should be edited to turn this feature on:

1. The `security.enableEncryption` key should be set to `true` (the default value).
2. The `security.encryptionCipherMode` key should specify proper cipher mode for decryption. The value can be one of the following two variants:
   - `AES256-CBC` (the default one for the Operator and Percona Server for MongoDB)
   - `AES256-GCM`
3. `security.encryptionKeySecret` should specify a secret object with the encryption key:

```
mongod:
  ...
  security:
    ...
    encryptionKeySecret: my-cluster-name-mongodb-encryption-key
```

   Encryption key secret will be created automatically if it doesn't exist. If you would like to create it yourself, take into account that the key must be a 32 character string encoded in base64.

Part V

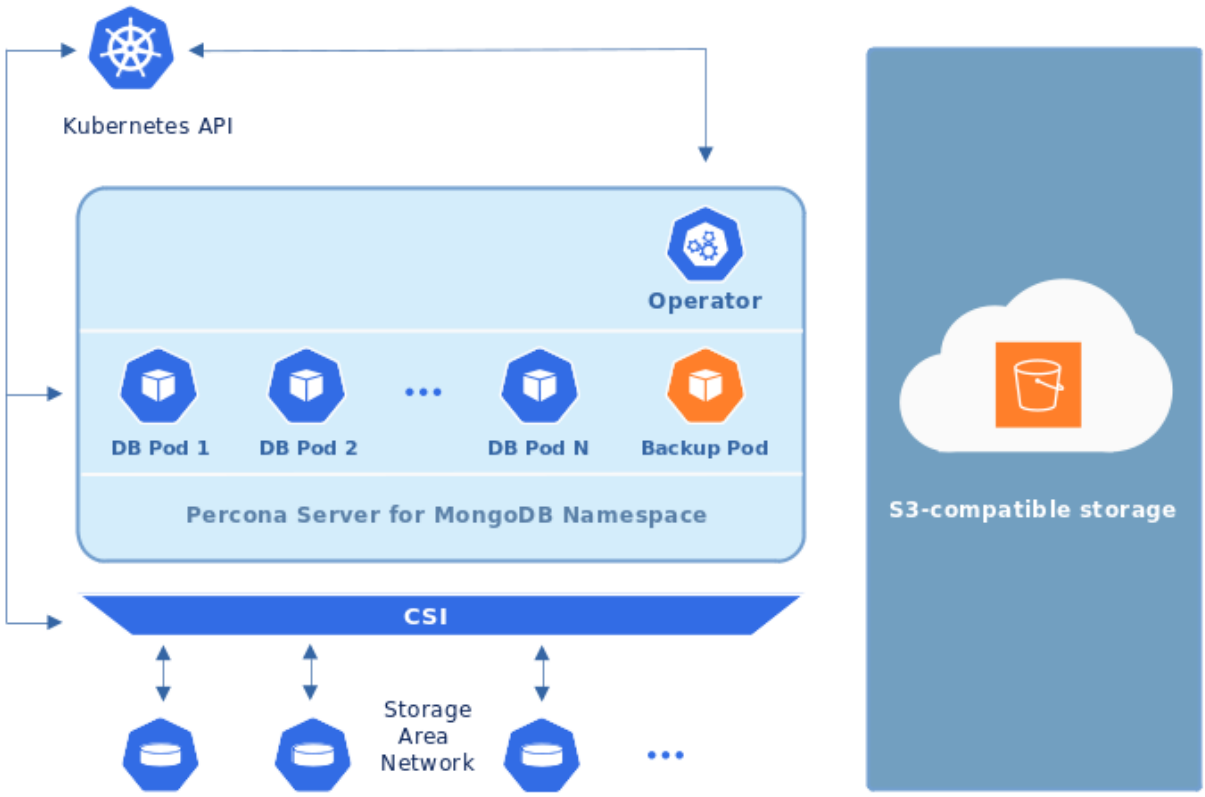
Management

CHAPTER

NINETEEN

PROVIDING BACKUPS

The Operator usually stores Server for MongoDB backups on Amazon S3 or S3-compatible storage outside the Kubernetes cluster:



The Operator allows doing cluster backup 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. Both ways use the Percona Backup for MongoDB tool.

- *Making scheduled backups*
- *Making on-demand backup*
- *Storing operations logs for point-in-time recovery*
- *Restore the cluster from a previously saved backup*

19.1 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.

```
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 three subsections:

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

Here is an example which uses Amazon S3 storage for backups:

```
...
backup:
  enabled: true
  version: 0.3.0
  ...
  storages:
    s3-us-west:
      type: s3
      s3:
        bucket: S3-BACKUP-BUCKET-NAME-HERE
        region: us-west-2
        credentialsSecret: my-cluster-name-backup-s3
  ...
  tasks:
   - name: "sat-night-backup"
     schedule: "0 0 * * 6"
     keep: 3
```

(continues on next page)

(continued from previous page)

```
    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 *Operator 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 *Operator Custom Resource options*.

19.2 Making on-demand backup

To make on-demand backup, user should use YAML file with correct names for the backup and the Percona Server for MongoDB Cluster, and correct PVC settings. The example of such file is deploy/backup/backup.yaml.

When the backup config file is ready, actual backup command is executed:

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

The example of such file is 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: psmdb.percona.com/v1
kind: PerconaServerMongoDBBackup
metadata:
  name: backup1
spec:
  psmdbCluster: my-cluster-name
  storageName: s3-us-west
EOF
```

19.3 Storing operations logs for point-in-time recovery

Point-in-time recovery functionality allows users to roll back the cluster to a specific date and time. Technically, this feature involves saving operations log updates to the S3-compatible backup storage.

To be used, it requires setting the *backup.pitr.enabled* key in the `deploy/cr.yaml` configuration file:

```
backup:
  ...
```

(continues on next page)

(continued from previous page)

```
  pitr:
    enabled: true
```

Note: It is necessary to have at least one full backup to use point-in-time recovery. Percona Backup for MongoDB will not upload operations logs if there is no full backup. This is true for new clusters and also true for clusters which have been just recovered from backup.

Percona Backup for MongoDB uploads operations logs to the same bucket where full backup is stored. This makes point-in-time recovery functionality available only if there is a single bucket in *spec.backup.storages*. Otherwise point-in-time recovery will not be enabled and there will be an error message in the operator logs.

Note: Adding a new bucket when point-in-time recovery is enabled will not break it, but put error message about the additional bucket in the operator logs as well.

19.4 Restore the cluster from a previously saved backup

The Operator supports the ability to perform a full restore on a MongoDB cluster as well as a point-in-time-recovery.

19.4.1 Restoring without point-in-time recovery

Following steps are needed to restore a previously saved backup:

1. First of all make sure that the cluster is running.
2. Now find out correct names for the **backup** and the **cluster**. Available backups can be listed with the following command:

   ```
   kubectl get psmdb-backup
   ```

   And the following command will list available clusters:

   ```
   kubectl get psmdb
   ```

3. When both correct names are known, run the actual restoration process:

   ```
   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: psmdb.percona.com/v1
   kind: PerconaServerMongoDBRestore
   metadata:
     name: restore1
   spec:
     clusterName: my-cluster-name
     backupName: backup1
   EOF
   ```

19.4.2 Restoring backup with point-in-time recovery

Following steps are needed to roll back the cluster to a specific date and time:

1. First of all make sure that the cluster is running.
2. Now find out correct names for the **backup** and the **cluster**. Available backups can be listed with the following command:

```
kubectl get psmdb-backup
```

   And the following command will list available clusters:

```
kubectl get psmdb
```

3. Edit the `deploy/backup/restore.yaml` file, adding the right cluster name and additional restoration parameters to the `pitr` section

```
...
spec:
  clusterName: my-cluster-name
  backupName: backup1
  pitr:
    type: date
    date: YYYY-MM-DD hh:mm:ss
```

3. Run the actual restoration process:

```
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: psmdb.percona.com/v1
kind: PerconaServerMongoDBRestore
metadata:
  name: restore1
spec:
  clusterName: my-cluster-name
  backupName: backup1
  pitr:
    type: date
    date: YYYY-MM-DD hh:mm:ss
EOF
```

19.5 Delete the unneeded backup

The maximum amount of stored backups is controlled by the *backup.tasks.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 psmdb-backup
```

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

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

CHAPTER

TWENTY

PAUSE/RESUME PERCONA SERVER FOR MONGODB

There may be external situations when it is needed to shutdown the 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
```

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

CHAPTER

TWENTYONE

CREATING A PRIVATE S3-COMPATIBLE CLOUD FOR BACKUPS

As it is mentioned in backups any cloud storage which implements the S3 API can be used for backups. The one way to setup and implement the S3 API storage on Kubernetes or OpenShift is Minio - the S3-compatible object storage server deployed via Docker on your own infrastructure.

Setting up Minio to be used with Percona Server for MongoDB Operator backups involves following steps:

1. Install Minio in your Kubernetes or OpenShift environment and create the correspondent Kubernetes Service as follows:

```
helm install \
  --name minio-service \
  --set accessKey=some-access-key \
  --set secretKey=some-secret-key \
  --set service.type=ClusterIP \
  --set configPath=/tmp/.minio/ \
  --set persistence.size=2G \
  --set environment.MINIO_REGION=us-east-1 \
  stable/minio
```

   Don't forget to substitute default `some-access-key` and `some-secret-key` strings in this command with actual unique key values. The values can be used later for access control. The `storageClass` option is needed if you are using the special Kubernetes Storage Class for backups. Otherwise, this setting may be omitted. You may also notice the `MINIO_REGION` value which is may not be used within a private cloud. Use the same region value here and on later steps (`us-east-1` is a good default choice).

2. Create an S3 bucket for backups:

```
kubectl run -i --rm aws-cli --image=perconalab/awscli --restart=Never -- \
 bash -c 'AWS_ACCESS_KEY_ID=some-access-key \
 AWS_SECRET_ACCESS_KEY=some-secret-key \
 AWS_DEFAULT_REGION=us-east-1 \
  /usr/bin/aws \
  --endpoint-url http://minio-service:9000 \
  s3 mb s3://operator-testing'
```

   This command creates the bucket named `operator-testing` with the selected access and secret keys (substitute `some-access-key` and `some-secret-key` with the values used on the previous step).

3. Now edit the backup section of the deploy/cr.yaml file to set proper values for the `bucket` (the S3 bucket for backups created on the previous step), `region`, `credentialsSecret` and the `endpointUrl` (which should point to the previously created Minio Service).

```
...
backup:
```

(continues on next page)

(continued from previous page)

```
  enabled: true
  version: 0.3.0
  ...
  storages:
    minio:
      type: s3
      s3:
        bucket: operator-testing
        region: us-east-1
        credentialsSecret: my-cluster-name-backup-minio
        endpointUrl: http://minio-service:9000
  ...
```

The option which should be specially mentioned is `credentialsSecret` which is a Kubernetes secret for backups. Sample backup-s3.yaml can be used to create this secret object. Check that the object contains the proper `name` value and is equal to the one specified for `credentialsSecret`, i.e. `my-cluster-name-backup-minio` in the backup to Minio example, and also contains the proper `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY` keys. After you have finished editing the file, the secrets object are created or updated when you run the following command:

```
$ kubectl apply -f deploy/backup-s3.yaml
```

4. When the setup process is completed, making the backup is based on a script. Following example illustrates how to make an on-demand backup:

```
kubectl run -it --rm pbmctl --image=percona/percona-server-mongodb-operator:0.3.0-
↪backup-pbmctl --restart=Never -- \
  run backup \
  --server-address=<cluster-name>-backup-coordinator:10001 \
  --storage <storage> \
  --compression-algorithm=gzip \
  --description=my-backup
```

Don't forget to specify the name of your cluster instead of the `<cluster-name>` part of the Backup Coordinator URL (the cluster name is specified in the deploy/cr.yaml file). Also substitute `<storage>` with the actual storage name located in a subsection inside of the `backups` in the deploy/cr.yaml file. In the earlier example this value is `minio`.

5. To restore a previously saved backup you must specify the backup name. With the proper Backup Coordinator URL and storage name, you can obtain a list of the available backups:

```
kubectl run -it --rm pbmctl --image=percona/percona-server-mongodb-operator:0.3.0-
↪backup-pbmctl --restart=Never -- list backups --server-address=<cluster-name>-
↪backup-coordinator:10001
```

Now, restore the backup, using backup name instead of the `backup-name` parameter:

```
kubectl run -it --rm pbmctl --image=percona/percona-server-mongodb-operator:0.3.0-
↪backup-pbmctl --restart=Never -- \
  run restore \
  --server-address=<cluster-name>-backup-coordinator:10001 \
  --storage <storage> \
  backup-name
```

CHAPTER

TWENTYTWO

UPDATE PERCONA SERVER FOR MONGODB OPERATOR

Starting from the version 1.1.0 the Percona Kubernetes Operator for MongoDB allows upgrades to newer versions. This includes upgrades of the Operator itself, and upgrades of the Percona Server for MongoDB.

22.1 Upgrading the Operator

This upgrade can be done either in semi-automatic or in manual mode.

Note: Manual update mode is the recommended way for a production cluster.

22.1.1 Semi-automatic upgrade

Note: Only the incremental update to a nearest minor version is supported (for example, update from 1.5.0 to 1.6.0). To update to a newer version, which differs from the current version by more than one, make several incremental updates sequentially.

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-server-mongodb-
↪operator/v1.8.0/deploy/crd.yaml
kubectl apply -f https://raw.githubusercontent.com/percona/percona-server-mongodb-
↪operator/v1.8.0/deploy/rbac.yaml
```

2. Edit the `deploy/cr.yaml` file, setting `updateStrategy` key to `RollingUpdate`.

3. Now you should apply a patch to your deployment, supplying necessary image names with a newer version tag. This is done with the `kubectl patch deployment` command. For example, updating to the `1.8.0` version should look as follows:

```
kubectl patch deployment percona-server-mongodb-operator \
   -p'{"spec":{"template":{"spec":{"containers":[{"name":"percona-server-mongodb-
↪operator","image":"percona/percona-server-mongodb-operator:1.8.0"}]}}}}'

kubectl patch psmdb my-cluster-name --type=merge --patch '{
   "spec": {
       "crVersion":"1.8.0",
       "image": "percona/percona-server-mongodb:4.4.5-7",
```

(continues on next page)

(continued from previous page)

```
      "backup": { "image": "percona/percona-server-mongodb-operator:1.8.0-backup
↪" },
      "pmm": { "image": "percona/pmm-client:2.12.0" }
   }}'
```

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

```
kubectl rollout status sts my-cluster-name-rs0
```

22.1.2 Manual upgrade

Note: Only the incremental update to a nearest minor version of the Operator is supported (for example, update from 1.5.0 to 1.6.0). To update to a newer version, which differs from the current version by more than one, make several incremental updates sequentially.

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-server-mongodb-
↪operator/v1.8.0/deploy/crd.yaml
kubectl apply -f https://raw.githubusercontent.com/percona/percona-server-mongodb-
↪operator/v1.8.0/deploy/rbac.yaml
```

2. Edit the `deploy/cr.yaml` file, setting `updateStrategy` key to `OnDelete`.

3. Now you should apply a patch to your deployment, supplying necessary image names with a newer version tag. This is done with the `kubectl patch deployment` command. For example, updating to the `1.8.0` version should look as follows:

```
kubectl patch deployment percona-server-mongodb-operator \
   -p'{"spec":{"template":{"spec":{"containers":[{"name":"percona-server-mongodb-
↪operator","image":"percona/percona-server-mongodb-operator:1.8.0"}]}}}}'

kubectl patch psmdb my-cluster-name --type=merge --patch '{
   "spec": {
      "crVersion":"1.8.0",
      "image": "percona/percona-server-mongodb:4.4.5-7",
      "backup": { "image": "percona/percona-server-mongodb-operator:1.8.0-backup
↪" },
      "pmm": { "image": "percona/pmm-client:2.12.0" }
   }}'
```

4. Pod with the newer Percona Server for MongoDB image will start after you delete it. Delete targeted Pods manually one by one to make them restart in the desired order:

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

   ```
   kubectl delete pod my-cluster-name-rs0-2
   ```

   2. Wait until Pod becomes ready:

   ```
   kubectl get pod my-cluster-name-rs0-2
   ```

The output should be like this:

```
NAME                       READY   STATUS    RESTARTS   AGE
my-cluster-name-rs0-2      1/1     Running   0          3m33s
```

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

22.2 Upgrading Percona Server for MongoDB

Starting from version 1.5.0, the Operator can do fully automatic upgrades to the newer versions of Percona Server for MongoDB 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.

Note: Being enabled, Smart Update will force the Operator to take MongoDB version from Version Service and not from the `mongod.image` option during the very first start of the cluster.

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 upgrade will choose the most recent version of software flagged as Recommended (for clusters created from scratch, the Percona Server for MongoDB 4.4 version will be selected instead of the Percona Server for MongoDB 4.2, 4.0, or 3.6 version regardless of the image path; for already existing clusters, the 4.4 vs. 4.2, 4.0, or 3.6 branch choice will be preserved),
   - `4.4-recommended`, `4.2-recommended`, `4.0-recommended`, `3.6-recommended` - same as above, but preserves specific major MongoDB version for newly provisioned clusters (ex. 4.4 will not be automatically used instead of 4.2),
   - `Latest` - automatic upgrade will choose the most recent version of the software available (for clusters created from scratch, the Percona Server for MongoDB 4.4 version will be selected instead of the Percona Server for MongoDB 4.2, 4.0, or 3.6 version regardless of the image path; for already existing clusters, the 4.4 vs. 4.2, 4.0, or 3.6 branch choice will be preserved),
   - `4.4-latest`, `4.2-latest`, `4.0-latest`, `3.6-latest` - same as above, but preserves specific major MongoDB version for newly provisioned clusters (ex. 4.4 will not be automatically used instead of 4.2),
   - *version number* - specify the desired version explicitly (version numbers are specified as 4.4.5-7, 4.2.13-14, 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 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 * * *"
...
```

22.2.1 Percona Server for MongoDB major version upgrades

Normally automatic upgrade takes place within minor versions (for example, from `4.2.11-12` to `4.2.12-13`) of MongoDB. Major versions upgrade (for example moving from `4.2-recommended` to `4.4-recommended`) is more complicated task which might potentially affect how data is stored and how applications interacts with the database (in case of some API changes).

Such upgrade is supported by the Operator within one major version at a time: for example, to change Percona Server for MongoDB major version from 4.0 to 4.4, you should first upgrade it to 4.2, and later make a separate upgrade from 4.2 to 4.4. The same is true for major version downgrades.

Note: It is recommended to take a backup before upgrade, as well as to perform upgrade on staging environment.

Major version upgrade can be initiated using the *upgradeOptions.apply* key in the `deploy/cr.yaml` configuration file:

```
spec:
  upgradeOptions:
    apply: 4.4-recommended
```

Note: When making downgrades (e.g. changing version from 4.4 to 4.2), make sure to remove incompatible features that are persisted and/or update incompatible configuration settings. Compatibility issues between major MongoDB versions can be found in upstream documentation.

By default the Operator doesn't set FeatureCompatibilityVersion (FCV) to match the new version, thus making sure that backwards-incompatible features are not automatically enabled with the major version upgrade (which is recommended and safe behavior). You can turn this backward compatibility off at any moment (after the upgrade or even before it) by setting the *upgradeOptions.setFCV* flag in the `deploy/cr.yaml` configuration file to `true`.

Note: With setFeatureCompatibilityVersion set major version rollback is not currently supported by the Operator. Therefore it is recommended to stay without enabling this flag for some time after the major upgrade to ensure the likelihood of downgrade is minimal. Setting `setFCV` flag to `true` simultaneously with the `apply` flag should be done only if the whole procedure is tested on staging and you are 100% sure about it.

CHAPTER

TWENTYTHREE

SCALE PERCONA SERVER FOR MONGODB ON KUBERNETES AND OPENSHIFT

One of the great advantages brought by Kubernetes and the OpenShift platform is the ease of an application scaling. Scaling a Deployment up or down ensures new Pods are created and set to available Kubernetes nodes.

The size of the cluster is controlled by the `size` key in the *Custom Resource options* configuration.

Note: The Operator will not allow to scale Percona Server for MongoDB with the `kubectl scale statefulset <StatefulSet name>` command as it puts `size` configuration options out of sync.

You can change size separately for different components of your cluster by setting this option in the appropriate subsections:

- *replsets.size* allows to set the size of the MongoDB Replica Set,
- *replsets.arbiter.size* allows to set the number of *Replica Set Arbiter instances*,
- *sharding.configsvrReplSet.size* allows to set the number of Config Server instances,
- *sharding.mongos.size* allows to set the number of mongos instances.

For example, the following update in `deploy/cr.yaml` will set the size of the MongoDB Replica Set to `5` nodes:

```
....
replsets:
  ....
  size: 5
  ....
```

Don't forget to apply changes as usual, running the `kubectl apply -f deploy/cr.yaml` command.

CHAPTER

TWENTYFOUR

MONITORING

Percona Monitoring and Management (PMM) provides an excellent solution of monitoring Percona Server for MongoDB.

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.

24.1 Installing 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.

24.2 Installing 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``PMM_SERVER_USER`` key in the deploy/secrets.yaml secrets file contains your PMM Server user name (`admin` by default).
   - make sure the `PMM_SERVER_PASSWORD` 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":{"PMM_SERVER_PASSWORD
↪": '$(echo -n new_password | base64)'}}'
```

- you can also use `pmm.mongodParams` and `pmm.mongosParams` keys to specify additional parameters for the pmm-admin add mongodb command for `mongod` and `mongos` Pods respectively, if needed.

Note: Please take into account that Operator automatically manages common MongoDB Service Monitoring parameters mentioned in the officiall `pmm-admin add mongodb` 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 my-cluster-name-rs0-0 -c pmm-client
```

3. Run the following command:

   ```
   kubectl get service/monitoring-service -o wide
   ```

   In the results, locate the the `EXTERNAL-IP` field. The external-ip address can be used to access PMM via *https* in a web browser, with the login/password authentication, and the browser is configured to show Percona Server for MongoDB metrics.

CHAPTER

TWENTYFIVE

DEBUG

For the cases when Pods are failing for some reason or just show abnormal behavior, the Operator can be used with a special *debug image* of the Percona Server for MongoDB, which has the following specifics:

- it avoids restarting on fail,
- it contains additional tools useful for debugging (sudo, telnet, gdb, mongodb-debuginfo package, etc.),
- extra verbosity is added to the mongodb daemon.

Particularly, using this image is useful if the container entry point fails (`mongod` 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.

To use the debug image instead of the normal one, set the following image name for the `image` key in the `deploy/cr.yaml` configuration file:

```
percona/percona-server-mongodb:4.4.5-7-debug
```

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

TWENTYSIX

CUSTOM RESOURCE OPTIONS

The operator is 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 Server for MongoDB Cluster,
- `finalizers.delete-psmdb-pvc`, if present, activates the Finalizer which deletes appropriate Persistent Volume Claims 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
platform	string	`kubernetes`	Override/set the Kubernetes platform: *kubernetes* or *openshift*. Set openshift on OpenShift 3.11+
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.
crVersion	string	`1.8.0`	Version of the Operator the Custom Resource belongs to
image	string	`percona/percona-server-mongodb:4.4.5-7`	The Docker image of Percona Server for MongoDB to deploy (actual image names can be found *in the list of certified images*)
imagePullPolicy	string	`Always`	The policy used to update images
imagePullSecrets.name	string	`private-registry-credentials`	The Kubernetes ImagePullSecret to access the *custom registry*
ClusterServiceDNSSuffix	string	`svc.cluster.local`	The (non-standard) cluster domain to be used as a suffix of the Service name
runUid	int	1001	The (non-standard) user ID
allowUnsafeConfigurations	boolean	`false`	Prevents users from configuring a cluster with unsafe parameters: starting it with less than 3 replica set nodes, with an odd number of replica set nodes and no arbiter, or without TLS/SSL certificates, or running a sharded cluster with less than 3 config server Pods or less than 2 mongos Pods (if `false`, the Operator will automatically change unsafe parameters to safe defaults)
updateStrategy	string	`SmartUpdate`	A strategy the Operator uses for *upgrades*. Possible values are *SmartUpdate*, RollingUpdate and OnDelete.
upgradeOptions	*subdoc*		Upgrade configuration section
secrets	*subdoc*		Operator secrets section
replsets	*subdoc*		Operator MongoDB Replica Set section
pmm	*subdoc*		Percona Monitoring and Management section
sharding	*subdoc*		MongoDB sharding configuration section
mongod	*subdoc*		Operator MongoDB Mongod configuration section
backup	*subdoc*		Percona Server for MongoDB backups section

26.1 Upgrade Options Section

The `upgradeOptions` section in the deploy/cr.yaml file contains various configuration options to control Percona Server for MongoDB 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	`4.4-recommended`
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 Server for MongoDB (e.g. `4.4.2-4`) 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
Key	upgradeOptions.setFCV
Value	boolean
Example	`false`
Description	If enabled, FeatureCompatibilityVersion (FCV) will be set to match the version during major version upgrade

26.2 Secrets section

Each spec in its turn may contain some key-value pairs. The secrets one has only two of them:

Key	secrets.key
Value Type	string
Example	`my-cluster-name-mongodb-key`
Description	The secret name for the MongoDB Internal Auth Key. This secret is auto-created by the operator if it doesn't exist.
Key	secrets.users
Value Type	string
Example	`my-cluster-name-mongodb-users`
Description	The secret name for the MongoDB users required to run the operator. **This secret is required to run the operator.**

26.3 Replsets Section

The replsets section controls the MongoDB Replica Set.

Key	replsets.name
Value Type	string
Example	`rs 0`
Description	The name of the MongoDB Replica Set
Key	replsets.size
Value Type	int
Example	3
Description	The size of the MongoDB Replica Set, must be >= 3 for High-Availability
Key	replsets.affinity.antiAffinityTopologyKey
Value Type	string
Example	`kubernetes.io/hostname`
Description	The Kubernetes topologyKey node affinity constraint for the Replica Set nodes
Key	replsets.affinity.advanced
Value Type	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	replsets.tolerations.key
Value Type	string
Example	`node.alpha.kubernetes.io/unreachable`
Description	The Kubernetes Pod tolerations key for the Replica Set nodes
Key	replsets.tolerations.operator
Value Type	string
Example	`Exists`
Description	The Kubernetes Pod tolerations operator for the Replica Set nodes
Key	replsets.tolerations.effect
Value Type	string
Example	`NoExecute`
Description	The Kubernetes Pod tolerations effect for the Replica Set nodes
Key	replsets.tolerations.tolerationSeconds
Value Type	int
Example	`6000`
Description	The Kubernetes Pod tolerations time limit for the Replica Set nodes
Key	replsets.priorityClassName
Value Type	string
Example	`high priority`
Description	The Kuberentes Pod priority class for the Replica Set nodes

continues on next page

Table 1 – continued from previous page

Key	replsets.annotations.iam.amazonaws.com/role
Value Type	string
Example	`role-arn`
Description	The AWS IAM role for the Replica Set nodes
Key	replsets.labels
Value Type	label
Example	`rack: rack-22`
Description	The Kubernetes affinity labels for the Replica Set nodes
Key	replsets.nodeSelector
Value Type	label
Example	`disktype: ssd`
Description	The Kubernetes nodeSelector affinity constraint for the Replica Set nodes
Key	replsets.livenessProbe.failureThreshold
Value Type	int
Example	`4`
Description	Number of consecutive unsuccessful tries of the liveness probe to be undertaken before giving up.
Key	replsets.livenessProbe.initialDelaySeconds
Value Type	int
Example	`60`
Description	Number of seconds to wait after the container start before initiating the liveness probe.
Key	replsets.livenessProbe.periodSeconds
Value Type	int
Example	`30`
Description	How often to perform a liveness probe (in seconds).
Key	replsets.livenessProbe.successThreshold
Value Type	int
Example	`1`
Description	Minimum consecutive successes for the liveness probe to be considered successful after having failed.
Key	replsets.livenessProbe.timeoutSeconds
Value Type	int
Example	`5`
Description	Number of seconds after which the liveness probe times out.
Key	replsets.livenessProbe.startupDelaySeconds
Value Type	int
Example	`7200`
Description	Time after which the liveness probe is failed if the MongoDB instance didn't finish its full startup yet
Key	replsets.runtimeClassName
Value Type	string

continues on next page

Table 1 – continued from previous page

Example	`image-rc`
Description	Name of the Kubernetes Runtime Class for Replica Set Pods
Key	replsets.sidecars.image
Value Type	string
Example	`busybox`
Description	Image for the *custom sidecar container* for Replica Set Pods
Key	replsets.sidecars.command
Value Type	array
Example	`["/bin/sh"]`
Description	Command for the *custom sidecar container* for Replica Set Pods
Key	replsets.sidecars.args
Value Type	array
Example	`["-c", "while true; do echo echo $(date -u) 'test' >> /dev/null; sleep 5;done"]`
Description	Command arguments for the *custom sidecar container* for Replica Set Pods
Key	replsets.sidecars.name
Value Type	string
Example	`rs-sidecar-1`
Description	Name of the *custom sidecar container* for Replica Set Pods
Key	replsets.podDisruptionBudget.maxUnavailable
Value Type	int
Example	`1`
Description	The Kubernetes Pod distribution budget limit specifying the maximum value for unavailable Pods
Key	replsets.podDisruptionBudget.minAvailable
Value Type	int
Example	`1`
Description	The Kubernetes Pod distribution budget limit specifying the minimum value for available Pods
Key	replsets.expose.enabled
Value Type	boolean
Example	`false`
Description	Enable or disable exposing MongoDB Replica Set nodes with dedicated IP addresses
Key	replsets.expose.exposeType
Value Type	string
Example	`ClusterIP`
Description	The IP address type to be exposed
Key	replsets.arbiter.enabled
Value Type	boolean
Example	`false`
Description	Enable or disable creation of Replica Set Arbiter nodes within the cluster
Key	replsets.arbiter.size

continues on next page

Table 1 – continued from previous page

Value Type	int
Example	`1`
Description	The number of Replica Set Arbiter instances within the cluster
Key	replsets.arbiter.afinity.antiAffinityTopologyKey
Value Type	string
Example	`kubernetes.io/hostname`
Description	The Kubernetes topologyKey node affinity constraint for the Arbiter
Key	replsets.arbiter.affinity.advanced
Value Type	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	replsets.arbiter.tolerations.key
Value Type	string
Example	`node.alpha.kubernetes.io/unreachable`
Description	The Kubernetes Pod tolerations key for the Arbiter nodes
Key	replsets.arbiter.tolerations.operator
Value Type	string
Example	`Exists`
Description	The Kubernetes Pod tolerations operator for the Arbiter nodes
Key	replsets.arbiter.tolerations.effect
Value Type	string
Example	`NoExecute`
Description	The Kubernetes Pod tolerations effect for the Arbiter nodes
Key	replsets.arbiter.tolerations.tolerationSeconds
Value Type	int
Example	`6000`
Description	The Kubernetes Pod tolerations time limit for the Arbiter nodes
Key	replsets.arbiter.priorityClassName
Value Type	string
Example	`high priority`
Description	The Kuberentes Pod priority class for the Arbiter nodes
Key	replsets.arbiter.annotations.iam.amazonaws.com/role
Value Type	string
Example	`role-arn`
Description	The AWS IAM role for the Arbiter nodes
Key	replsets.arbiter.labels
Value Type	label
Example	`rack: rack-22`
Description	The Kubernetes affinity labels for the Arbiter nodes

continues on next page

Table 1 – continued from previous page

Key	replsets.arbiter.nodeSelector
Value Type	label
Example	`disktype: ssd`
Description	The Kubernetes nodeSelector affinity constraint for the Arbiter nodes
Key	replsets.schedulerName
Value Type	string
Example	`default`
Description	The Kubernetes Scheduler
Key	replsets.resources.limits.cpu
Value Type	string
Example	`300m`
Description	Kubernetes CPU limit for MongoDB container
Key	replsets.resources.limits.memory
Value Type	string
Example	`0.5G`
Description	Kubernetes Memory limit for MongoDB container
Key	replsets.resources.requests.cpu
Value Type	string
Example	
Description	The Kubernetes CPU requests for MongoDB container
Key	replsets.resources.requests.memory
Value Type	string
Example	
Description	The Kubernetes Memory requests for MongoDB container
Key	replsets.volumeSpec.emptyDir
Value Type	string
Example	`{}`
Description	The Kubernetes emptyDir volume, i.e. the directory which will be created on a node, and will be accessible to the MongoDB Pod containers
Key	replsets.volumeSpec.hostPath.path
Value Type	string
Example	`/data`
Description	Kubernetes hostPath volume, i.e. the file or directory of a node that will be accessible to the MongoDB Pod containers
Key	replsets.volumeSpec.hostPath.type
Value Type	string
Example	`Directory`
Description	The Kubernetes hostPath volume type
Key	replsets.volumeSpec.persistentVolumeClaim.storageClassName
Value Type	string

continues on next page

Table 1 – continued from previous page

Example	`standard`
Description	The Kubernetes Storage Class to use with the MongoDB container Persistent Volume Claim. Use Storage Class with XFS as the default filesystem if possible, for better MongoDB performance
Key	replsets.volumeSpec.persistentVolumeClaim.accessModes
Value Type	array
Example	`[ "ReadWriteOnce" ]`
Description	The Kubernetes Persistent Volume access modes for the MongoDB container
Key	replsets.volumeSpec.persistentVolumeClaim.resources.requests.storage
Value Type	string
Example	`3Gi`
Description	The Kubernetes Persistent Volume size for the MongoDB container

26.4 PMM Section

The `pmm` section in the deploy/cr.yaml file contains configuration options for Percona Monitoring and Management.

Key	pmm.enabled
Value Type	boolean
Example	`false`
Description	Enables or disables monitoring Percona Server for MongoDB with PMM
Key	pmm.image
Value Type	string
Example	`percona/pmm-client:2.12.0`
Description	PMM Client docker image to use
Key	pmm.serverHost
Value Type	string
Example	`monitoring-service`
Description	Address of the PMM Server to collect data from the Cluster
Key	pmm.mongodParams
Value Type	string
Example	`--environment=DEV-ENV --custom-labels=DEV-ENV`
Description	Additional parameters which will be passed to the pmm-admin add mongodb command for `mongod` Pods
Key	pmm.mongosParams
Value Type	string
Example	`--environment=DEV-ENV --custom-labels=DEV-ENV`
Description	Additional parameters which will be passed to the pmm-admin add mongodb command for `mongos` Pods

26.5 Sharding Section

The `sharding` section in the deploy/cr.yaml file contains configuration options for Percona Server for MondoDB *sharding*.

Key	sharding.enabled
Value Type	boolean
Example	`true`
Description	Enables or disables Percona Server for MondoDB sharding
Key	sharding.configsvrReplSet.size
Value Type	int
Example	`3`
Description	The number of Config Server instances within the cluster
Key	sharding.configsvrReplSet.runtimeClassName
Value Type	string
Example	`image-rc`
Description	Name of the Kubernetes Runtime Class for Config Server Pods
Key	sharding.configsvrReplSet.sidecars.image
Value Type	string
Example	`busybox`
Description	Image for the *custom sidecar container* for Config Server Pods
Key	sharding.configsvrReplSet.sidecars.command
Value Type	array
Example	`["/bin/sh"]`
Description	Command for the *custom sidecar container* for Config Server Pods
Key	sharding.configsvrReplSet.sidecars.args
Value Type	array
Example	`["-c", "while true; do echo echo $(date -u) 'test' >> /dev/null; sleep 5;done"]`
Description	Command arguments for the *custom sidecar container* for Config Server Pods
Key	sharding.configsvrReplSet.sidecars.name
Value Type	string
Example	`rs-sidecar-1`
Description	Name of the *custom sidecar container* for Config Server Pods
Key	sharding.configsvrReplSet.volumeSpec.emptyDir
Value Type	string
Example	`{}`
Description	The Kubernetes emptyDir volume, i.e. the directory which will be created on a node, and will be accessible to the Config Server Pod containers
Key	sharding.configsvrReplSet.volumeSpec.hostPath.path
Value Type	string
Example	`/data`

continues on next page

Table 2 – continued from previous page

Description	Kubernetes hostPath volume, i.e. the file or directory of a node that will be accessible to the Config Server Pod containers
Key	sharding.configsvrReplSet.volumeSpec.hostPath.type
Value Type	string
Example	`Directory`
Description	The Kubernetes hostPath volume type
Key	sharding.configsvrReplSet.volumeSpec.persistentVolumeClaim.storageClassName
Value Type	string
Example	`standard`
Description	The Kubernetes Storage Class to use with the Config Server container Persistent Volume Claim. Use Storage Class with XFS as the default filesystem if possible, for better MongoDB performance
Key	sharding.configsvrReplSet.volumeSpec.persistentVolumeClaim.accessModes
Value Type	array
Example	`[ "ReadWriteOnce" ]`
Description	The Kubernetes Persistent Volume access modes for the Config Server container
Key	sharding.configsvrReplSet.volumeSpec.persistentVolumeClaim.resources.requests.storage
Value Type	string
Example	`3Gi`
Description	The Kubernetes Persistent Volume size for the Config Server container
Key	sharding.mongos.size
Value Type	int
Example	`3`
Description	The number of mongos instances within the cluster
Key	sharding.mongos.afinity.antiAffinityTopologyKey
Value Type	string
Example	`kubernetes.io/hostname`
Description	The Kubernetes topologyKey node affinity constraint for mongos
Key	sharding.mongos.affinity.advanced
Value Type	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	sharding.mongos.tolerations.key
Value Type	string
Example	`node.alpha.kubernetes.io/unreachable`
Description	The Kubernetes Pod tolerations key for mongos instances
Key	sharding.mongos.tolerations.operator
Value Type	string
Example	`Exists`

continues on next page

Table 2 – continued from previous page

Description	The Kubernetes Pod tolerations operator for mongos instances
Key	sharding.mongos.tolerations.effect
Value Type	string
Example	`NoExecute`
Description	The Kubernetes Pod tolerations effect for mongos instances
Key	sharding.mongos.tolerations.tolerationSeconds
Value Type	int
Example	`6000`
Description	The Kubernetes Pod tolerations time limit for mongos instances
Key	sharding.mongos.priorityClassName
Value Type	string
Example	`high priority`
Description	The Kuberentes Pod priority class for mongos instances
Key	sharding.mongos.annotations.iam.amazonaws.com/role
Value Type	string
Example	`role-arn`
Description	The AWS IAM role for mongos instances
Key	sharding.mongos.labels
Value Type	label
Example	`rack: rack-22`
Description	The Kubernetes affinity labels for mongos instances
Key	sharding.mongos.nodeSelector
Value Type	label
Example	`disktype: ssd`
Description	The Kubernetes nodeSelector affinity constraint for mongos instances
Key	sharding.mongos.runtimeClassName
Value Type	string
Example	`image-rc`
Description	Name of the Kubernetes Runtime Class for mongos Pods
Key	sharding.mongos.sidecars.image
Value Type	string
Example	`busybox`
Description	Image for the *custom sidecar container* for mongos Pods
Key	sharding.mongos.sidecars.command
Value Type	array
Example	`["/bin/sh"]`
Description	Command for the *custom sidecar container* for mongos Pods
Key	sharding.mongos.sidecars.args
Value Type	array

continues on next page

Table 2 – continued from previous page

Example	`["-c", "while true; do echo echo $(date -u) 'test' >> /dev/null; sleep 5;done"]`
Description	Command arguments for the *custom sidecar container* for mongos Pods
Key	sharding.mongos.sidecars.name
Value Type	string
Example	`rs-sidecar-1`
Description	Name of the *custom sidecar container* for mongos Pods
Key	sharding.mongos.limits.cpu
Value Type	string
Example	`300m`
Description	Kubernetes CPU limit for mongos container
Key	sharding.mongos.limits.memory
Value Type	string
Example	`0.5G`
Description	Kubernetes Memory limit for mongos container
Key	sharding.mongos.resources.requests.cpu
Value Type	string
Example	`300m`
Description	The Kubernetes CPU requests for mongos container
Key	sharding.mongos.requests.memory
Value Type	string
Example	`0.5G`
Description	The Kubernetes Memory requests for mongos container
Key	sharding.mongos.expose.enabled
Value Type	boolean
Example	`false`
Description	Enable or disable exposing MongoDB mongos daemons with dedicated IP addresses
Key	sharding.mongos.expose.exposeType
Value Type	string
Example	`ClusterIP`
Description	The IP address type to be exposed
Key	sharding.mongos.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	sharding.mongos.serviceAnnotations
Value	string
Example	`service.beta.kubernetes.io/aws-load-balancer-backend-protocol: http`
Description	The Kubernetes annotations metadata for the MongoDB mongos daemon

continues on next page

Table 2 – continued from previous page

Key	sharding.mongos.auditLog.destination
Value Type	string
Example	
Description	Sets the auditLog.destination option for the MongoDB mongos daemon
Key	sharding.mongos.auditLog.format
Value Type	string
Example	`BSON`
Description	Sets the auditLog.format option for the MongoDB mongos daemon
Key	sharding.mongos.auditLog.filter
Value Type	string
Example	`{}`
Description	Sets the auditLog.filter option for the MongoDB mongos daemon

26.6 Mongod Section

This section contains the Mongod configuration options.

Key	mongod.net.port
Value Type	int
Example	`27017`
Description	Sets the MongoDB net.port option
Key	mongod.net.hostport
Value Type	int
Example	`0`
Description	Sets the Kubernetes hostPort option
Key	mongod.security.redactClientLogData
Value Type	bool
Example	`false`
Description	Enables/disables Percona Server for MongoDB Log Redaction
Key	mongod.security.enableEncryption
Value Type	bool
Example	`true`
Description	Enables/disables Percona Server for MongoDB data at rest encryption
Key	mongod.security.encryptionKeySecret
Value Type	string
Example	`my-cluster-name-mongodb-encryption-key`
Description	Specifies a secret object with the encryption key
Key	mongod.security.encryptionCipherMode
Value Type	string
Example	`AES256-CBC`

continues on next page

Table 3 – continued from previous page

Description	Sets Percona Server for MongoDB encryption cipher mode
Key	mongod.setParameter.ttlMonitorSleepSecs
Value Type	int
Example	`60`
Description	Sets the Percona Server for MongoDB `ttlMonitorSleepSecs` option
Key	mongod.setParameter.wiredTigerConcurrentReadTransactions
Value Type	int
Example	`128`
Description	Sets the wiredTigerConcurrentReadTransactions option
Key	mongod.setParameter.wiredTigerConcurrentWriteTransactions
Value Type	int
Example	`128`
Description	Sets the wiredTigerConcurrentWriteTransactions option
Key	mongod.storage.engine
Value Type	string
Example	`wiredTiger`
Description	Sets the storage.engine option
Key	mongod.storage.inMemory.engineConfig.inMemorySizeRatio
Value Type	float
Example	`0.9`
Description	The ratio used to compute the storage.engine.inMemory.inMemorySizeGb option
Key	mongod.storage.mmapv1.nsSize
Value Type	int
Example	`16`
Description	Sets the storage.mmapv1.nsSize option
Key	mongod.storage.mmapv1.smallfiles
Value Type	bool
Example	`false`
Description	Sets the storage.mmapv1.smallfiles option
Key	mongod.storage.wiredTiger.engineConfig.cacheSizeRatio
Value Type	float
Example	`0.5`
Description	The ratio used to compute the storage.wiredTiger.engineConfig.cacheSizeGB option
Key	mongod.storage.wiredTiger.engineConfig.directoryForIndexes
Value Type	bool
Example	`false`
Description	Sets the storage.wiredTiger.engineConfig.directoryForIndexes option
Key	mongod.storage.wiredTiger.engineConfig.journalCompressor
Value Type	string
Example	`snappy`

continues on next page

Table 3 – continued from previous page

Description	Sets the storage.wiredTiger.engineConfig.journalCompressor option
Key	mongod.storage.wiredTiger.collectionConfig.blockCompressor
Value Type	string
Example	`snappy`
Description	Sets the storage.wiredTiger.collectionConfig.blockCompressor option
Key	mongod.storage.wiredTiger.indexConfig.prefixCompression
Value Type	bool
Example	`true`
Description	Sets the storage.wiredTiger.indexConfig.prefixCompression option
Key	mongod.operationProfiling.mode
Value Type	string
Example	`slowOp`
Description	Sets the operationProfiling.mode option
Key	mongod.operationProfiling.slowOpThresholdMs
Value Type	int
Example	`100`
Description	Sets the operationProfiling.slowOpThresholdMs option
Key	mongod.operationProfiling.rateLimit
Value Type	int
Example	`1`
Description	Sets the operationProfiling.rateLimit option
Key	mongod.auditLog.destination
Value Type	string
Example	
Description	Sets the auditLog.destination option
Key	mongod.auditLog.format
Value Type	string
Example	`BSON`
Description	Sets the auditLog.format option
Key	mongod.auditLog.filter
Value Type	string
Example	`{}`
Description	Sets the auditLog.filter option

26.7 Backup Section

The `backup` section in the deploy/cr.yaml file contains the following configuration options for the regular Percona Server for MongoDB backups.

Key	backup.enabled
Value Type	boolean
Example	`true`
Description	Enables or disables making backups
Key	backup.debug
Value Type	boolean
Example	`true`
Description	Enables or disables debug mode for backups
Key	backup.restartOnFailure
Value Type	boolean
Example	`true`
Description	Enables or disables restarting the previously failed backup process
Key	backup.image
Value Type	string
Example	`percona/percona-server-mongodb-operator:1.8.0-backup`
Description	The Percona Server for MongoDB Docker image to use for the backup
Key	backup.serviceAccountName
Value Type	string
Example	`percona-server-mongodb-operator`
Description	Nname of the separate privileged service account used by the Operator
Key	backup.resources.limits.cpu
Value Type	string
Example	`100m`
Description	Kubernetes CPU limit for backups
Key	backup.resources.limits.memory
Value Type	string
Example	`0.2G`
Description	Kubernetes Memory limit for backups
Key	backup.resources.requests.cpu
Value Type	string
Example	`100m`
Description	The Kubernetes CPU requests for backups
Key	backup.resources.requests.memory
Value Type	string
Example	`0.1G`
Description	The Kubernetes Memory requests for backups

continues on next page

Table 4 – continued from previous page

Key	backup.storages.<storage-name>.type
Value	string
Example	`s3`
Description	The cloud storage type used for backups. Only `s3` type is currently supported
Key	backup.storages.<storage-name>.s3.credentialsSecret
Value	string
Example	`my-cluster-name-backup-s3`
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.pitr.enabled
Value	boolean
Example	`false`
Description	Enables or disables *point-in-time-recovery functionality*
Key	backup.tasks.name
Value Type	string
Example	
Description	The name of the backup
Key	backup.tasks.enabled
Value Type	boolean
Example	`true`
Description	Enables or disables this exact backup
Key	backup.tasks.schedule
Value Type	string
Example	`0 0 * * 6`
Description	The scheduled time to make a backup, specified in the crontab format
Key	backup.tasks.keep
Value Type	int

continues on next page

Table 4 – continued from previous page

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.tasks.storageName
Value Type	string
Example	`st-us-west`
Description	The name of the S3-compatible storage for backups, configured in the *storages* subsection
Key	backup.tasks.compressionType
Value Type	string
Example	`gzip`
Description	The backup compression format

CHAPTER

TWENTYSEVEN

PERCONA CERTIFIED IMAGES

Following table presents Percona's certified docker images to be used with the Percona Operator for Percona Server for MongoDB:

Image	Digest
percona/percona-server-mongodb-operator:1.8.0	c9d8253acb97d2bfe3d1cf1120216d20565da4eb5dfd0f3f6385bab7eeea2110
percona/pmm-client:2.12.0	e29616e36dcd5a6fd7de67b444e5a80680d56f52f1398c0f49ee92427be797e6
percona/percona-server-mongodb-operator:1.8.0-backup	e5d1fb76407bb8ed3626ee5db5119477bf369a5ab2d76b811bc7cf4c38dfe274
percona/percona-server-mongodb:4.4.5-7	14b419b686c0ca5cf856e2439bc213b85070ee2fe30832de0f46d1b602cc65ff
percona/percona-server-mongodb:4.4.3-5	28c137c87dc0ed3530398ee281b6998a902ca5bc39bf766f42de03b1dfc1fc57
percona/percona-server-mongodb:4.4.2-4	991d6049059e5eb1a74981290d829a5fb4ab0554993748fde1e67b2f46f26bf0
percona/percona-server-mongodb:4.2.13-14	5419687291e7fa1e3bf2ae1fa0340cbf72f38f8ad70c0800fa7b8c145f5e75ae
percona/percona-server-mongodb:4.2.12-13	dda89e647ea5aa1266055ef465d66a139722d9e3f78a839a90a9f081b09ce26d
percona/percona-server-mongodb:4.2.11-12	1909cb7a6ecea9bf0535b54aa86b9ae74ba2fa303c55cf4a1a54262fb0edbd3c
percona/percona-server-mongodb:4.2.8-8	a66e889d3e986413e41083a9c887f33173da05a41c8bd107cf50eede4588a505
percona/percona-server-mongodb:4.2.7-7	1d8a0859b48a3e9cadf9ad7308ec5aa4b278a64ca32ff5d887156b1b46146b13
percona/percona-server-mongodb:4.0.23-18	5f76442ea702633670982aa1059c495cb85d3be2fd4d7c713a832f1ab5a6f1ce
percona/percona-server-mongodb:4.0.22-17	51d8cd31489c5eefed8ed94a477f25176279c7e30b97024f69a0a76a3d55e64e
percona/percona-server-mongodb:4.0.21-15	663f6eb98ae625792d59c6072b5d3a3095112380de67097a411f597d785c423a
percona/percona-server-mongodb:4.0.20-13	badef1eb2807b0b27a2298f697388f1dffa5398d5caa306a65fc41b98f7a72e3
percona/percona-server-mongodb:4.0.19-12	24a8214d84c3a9a4147c12c4c159d4a1aa3dae831859f77b3db1a563a268e2bf
percona/percona-server-mongodb:4.0.18	bf9e69712868f7e93daef22c14c083bbb2a74d3028d78d8597b2aeacda340c69
percona/percona-server-mongodb:3.6.23-13.0	364ac99e61ec8d15a7ca89dfbf809156d6fbab51e45cb4f2540daeaaf7b8e7e7
percona/percona-server-mongodb:3.6.21-10.0	3868831e0b7e9a210d3fda5d794aea5438c3a92159a0a35688b090f8ee3ce1be
percona/percona-server-mongodb:3.6.19-7.0	fbc2a312446b393a0221797c93acb8fc4df84a1f725eb78e04f5111c63dbec62
percona/percona-server-mongodb:3.6.18-6.0	d559d75611d7bc0254a6d049dd95eacbb9b32cd7c4f7eee854d02e81e26d03f7
percona/percona-server-mongodb:3.6.18-5.0	0dc8bf7f135c5c7fdf15e1b9a02b0a6f08bc3de4c96f79b4f532ff682d2aff4b

CHAPTER

TWENTYEIGHT

PERCONA SERVER FOR MONGODB OPERATOR API DOCUMENTATION

Percona Operator Operator for Percona Server for MongoDB 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.

28.1 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-server-
↪mongodb-operator -o jsonpath='{.secrets[0].name}' -n default) -o jsonpath='{.
↪data.token}' -n default | base64 --decode )
```

28.2 Create new Percona Server for MongoDB cluster

Description:

```
The command to create a new Percona Server for MongoDB cluster
```

Kubectl Command:

```
kubectl apply -f percona-server-mongodb-operator/deploy/cr.yaml
```

URL:

```
https://$API_SERVER/apis/psmdb.percona.com/v1-8-0/namespaces/default/
↪perconaservermongodbs
```

Authentication:

```
Authorization: Bearer $KUBE_TOKEN
```

cURL Request:

```
curl -k -v -XPOST "https://$API_SERVER/apis/psmdb.percona.com/v1-8-0/namespaces/
↪default/perconaservermongodbs" \
            -H "Content-Type: application/json" \
            -H "Accept: application/json" \
            -H "Authorization: Bearer $KUBE_TOKEN" \
            -d "@cluster.json"
```

Request Body (cluster.json):

JSON:

```
{
   "apiVersion": "psmdb.percona.com/v1-5-0",
   "kind": "PerconaServerMongoDB",
   "metadata": {
      "name": "my-cluster-name"
   },
   "spec": {
      "image": "percona/percona-server-mongodb:4.2.8-8",
      "imagePullPolicy": "Always",
      "allowUnsafeConfigurations": false,
      "updateStrategy": "SmartUpdate",
      "secrets": {
         "users": "my-cluster-name-secrets"
      },
      "pmm": {
         "enabled": false,
```

(continues on next page)

(continued from previous page)

```
        "image": "percona/percona-server-mongodb-operator:1.5.0-pmm",
        "serverHost": "monitoring-service"
    },
    "replsets": [
        {
            "name": "rs0",
            "size": 3,
            "affinity": {
                "antiAffinityTopologyKey": "none"
            },
            "podDisruptionBudget": {
                "maxUnavailable": 1
            },
            "expose": {
                "enabled": false,
                "exposeType": "LoadBalancer"
            },
            "arbiter": {
                "enabled": false,
                "size": 1,
                "affinity": {
                    "antiAffinityTopologyKey": "none"
                }
            },
            "resources": {
                "limits": null
            },
            "volumeSpec": {
                "persistentVolumeClaim": {
                    "storageClassName": "standard",
                    "accessModes": [
                        "ReadWriteOnce"
                    ],
                    "resources": {
                        "requests": {
                            "storage": "3Gi"
                        }
                    }
                }
            }
        }
    ],
    "mongod": {
        "net": {
            "port": 27017,
            "hostPort": 0
        },
        "security": {
            "redactClientLogData": false,
            "enableEncryption": true,
            "encryptionKeySecret": "my-cluster-name-mongodb-encryption-key",
            "encryptionCipherMode": "AES256-CBC"
        },
        "setParameter": {
            "ttlMonitorSleepSecs": 60,
            "wiredTigerConcurrentReadTransactions": 128,
            "wiredTigerConcurrentWriteTransactions": 128
```

(continues on next page)

(continued from previous page)


```
        },
        "storage": {
           "engine": "wiredTiger",
           "inMemory": {
              "engineConfig": {
                 "inMemorySizeRatio": 0.9
              }
           },
           "mmapv1": {
              "nsSize": 16,
              "smallfiles": false
           },
           "wiredTiger": {
              "engineConfig": {
                 "cacheSizeRatio": 0.5,
                 "directoryForIndexes": false,
                 "journalCompressor": "snappy"
              },
              "collectionConfig": {
                 "blockCompressor": "snappy"
              },
              "indexConfig": {
                 "prefixCompression": true
              }
           }
        },
        "operationProfiling": {
           "mode": "slowOp",
           "slowOpThresholdMs": 100,
           "rateLimit": 100
        }
     },
     "backup": {
        "enabled": true,
        "restartOnFailure": true,
        "image": "percona/percona-server-mongodb-operator:1.5.0-backup",
        "serviceAccountName": "percona-server-mongodb-operator",
        "storages": null,
        "tasks": null
     }
  }
}
```


Inputs:

Metadata:

1. Name (String, min-length: 1) : `contains name of cluster`

Spec:

1. secrets[users] (String, min-length: 1) : `contains name of secret for the users`
2. allowUnsafeConfigurations (Boolean, Default: false) : `allow unsafe configurations to run`
3. image (String, min-length: 1) : `name of the Percona Server for MongoDB cluster image`

replsets:

1. name (String, min-length: 1) : `name of monogo replicaset`
2. size (Integer, min-value: 1) : `contains size of MongoDB replicaset`
3. expose[exposeType] (Integer, min-value: 1) : `type of service to expose replicaset`
4. arbiter (Object) : `configuration for mongo arbiter`

mongod:

1. net:
   1. port (Integer, min-value: 0) : `contains mongod container port`
   2. hostPort (Integer, min-value: 0) : `host port to expose mongod on`
2. security:
   1. enableEncryption (Boolean, Default: true) : `enable encrypting mongod storage`
   2. encryptionKeySecret (String, min-length: 1) : `name of encryption key secret`
   3. encryptionCipherMode (String, min-length: 1) : `type of encryption cipher to use`
3. setParameter (Object): `configure mongod enginer paramters`
4. storage:
   1. engine (String, min-length: 1, default "wiredTiger"): `name of mongod storage engine`
   2. inMemory (Object) : `wiredTiger engine configuration`
   3. wiredTiger (Object) : `wiredTiger engine configuration`

pmm:

1. serverHost (String, min-length: 1) : `serivce name for monitoring`
2. image (String, min-length: 1) : `name of pmm image`

backup:

1. image (String, min-length: 1) : `name of MngoDB backup docker image`
2. serviceAccountName (String, min-length: 1) `name of service account to use for backup`
3. storages (Object) : `storage configuration object for backup`

Response:

JSON


```
{
   "apiVersion":"psmdb.percona.com/v1-5-0",
   "kind":"PerconaServerMongoDB",
   "metadata":{
      "annotations":{
         "kubectl.kubernetes.io/last-applied-configuration":"{\"apiVersion\":\"psmdb.
↪percona.com/v1-5-0\",\"kind\":\"PerconaServerMongoDB\",\"metadata\":{\"annotations\
↪":{},\"name\":\"my-cluster-name\",\"namespace\":\"default\"},\"spec\":{\
↪"allowUnsafeConfigurations\":false,\"backup\":{\"enabled\":true,\"image\":\"percona/
↪percona-server-mongodb-operator:1.5.0-backup\",\"restartOnFailure\":true,\
↪"serviceAccountName\":\"percona-server-mongodb-operator\",\"storages\":null,\"tasks\
↪":null},\"image\":\"percona/percona-server-mongodb:4.2.8-8\",\"imagePullPolicy\":\
↪"Always\",\"mongod\":{\"net\":{\"hostPort\":0,\"port\":27017},\"operationProfiling\
↪":{\"mode\":\"slowOp\",\"rateLimit\":100,\"slowOpThresholdMs\":100},\"security\":{\
↪"enableEncryption\":true,\"encryptionCipherMode\":\"AES256-CBC\",\
↪"encryptionKeySecret\":\"my-cluster-name-mongodb-encryption-key\",\
↪"redactClientLogData\":false},\"setParameter\":{\"ttlMonitorSleepSecs\":60,\
↪"wiredTigerConcurrentReadTransactions\":128,\"wiredTigerConcurrentWriteTransactions\
↪":128},\"storage\":{\"engine\":\"wiredTiger\",\"inMemory\":{\"engineConfig\":{\
↪"inMemorySizeRatio\":0.9},\"mmapv1\":{\"nsSize\":16,\"smallfiles\":false},\
```


(continues on next page)

(continued from previous page)


```
    },
    "creationTimestamp":"2020-07-24T14:27:58Z",
    "generation":1,
    "managedFields":[
       {
          "apiVersion":"psmdb.percona.com/v1-5-0",
          "fieldsType":"FieldsV1",
          "fieldsV1":{
             "f:metadata":{
                "f:annotations":{
                   ".":{

                   },
                   "f:kubectl.kubernetes.io/last-applied-configuration":{

                   }
                }
             },
             "f:spec":{
                ".":{

                },
                "f:allowUnsafeConfigurations":{

                },
                "f:backup":{
                   ".":{

                   },
                   "f:enabled":{

                   },
                   "f:image":{

                   },
                   "f:restartOnFailure":{

                   },
                   "f:serviceAccountName":{

                   },
                   "f:storages":{

                   },
                   "f:tasks":{

                   }
                },
                "f:image":{

                },
                "f:imagePullPolicy":{

                },
                "f:mongod":{
                   ".":{
```


(continues on next page)

(continued from previous page)


```
                    },
                    "f:net":{
                        ".":{

                        },
                        "f:hostPort":{

                        },
                        "f:port":{

                        }
                    },
                    "f:operationProfiling":{
                        ".":{

                        },
                        "f:mode":{

                        },
                        "f:rateLimit":{

                        },
                        "f:slowOpThresholdMs":{

                        }
                    },
                    "f:security":{
                        ".":{

                        },
                        "f:enableEncryption":{

                        },
                        "f:encryptionCipherMode":{

                        },
                        "f:encryptionKeySecret":{

                        },
                        "f:redactClientLogData":{

                        }
                    },
                    "f:setParameter":{
                        ".":{

                        },
                        "f:ttlMonitorSleepSecs":{

                        },
                        "f:wiredTigerConcurrentReadTransactions":{

                        },
                        "f:wiredTigerConcurrentWriteTransactions":{

                        }
                    },
```


(continues on next page)

(continued from previous page)

```
                    "f:storage":{
                        ".":{

                        },
                        "f:engine":{

                        },
                        "f:inMemory":{
                            ".":{

                            },
                            "f:engineConfig":{
                                ".":{

                                },
                                "f:inMemorySizeRatio":{

                                }
                            }
                        },
                        "f:mmapv1":{
                            ".":{

                            },
                            "f:nsSize":{

                            },
                            "f:smallfiles":{

                            }
                        },
                        "f:wiredTiger":{
                            ".":{

                            },
                            "f:collectionConfig":{
                                ".":{

                                },
                                "f:blockCompressor":{

                                }
                            },
                            "f:engineConfig":{
                                ".":{

                                },
                                "f:cacheSizeRatio":{

                                },
                                "f:directoryForIndexes":{

                                },
                                "f:journalCompressor":{

                                }
                            },
```

(continues on next page)

(continued from previous page)


```
                        "f:indexConfig":{
                           ".":{

                           },
                           "f:prefixCompression":{

                           }
                        }
                     }
                  }
               },
               "f:pmm":{
                  ".":{

                  },
                  "f:enabled":{

                  },
                  "f:image":{

                  },
                  "f:serverHost":{

                  }
               },
               "f:replsets":{

               },
               "f:secrets":{
                  ".":{

                  },
                  "f:users":{

                  }
               },
               "f:updateStrategy":{

               }
            }
         },
         "manager":"kubectl",
         "operation":"Update",
         "time":"2020-07-24T14:27:58Z"
      }
   ],
   "name":"my-cluster-name",
   "namespace":"default",
   "resourceVersion":"1268922",
   "selfLink":"/apis/psmdb.percona.com/v1-5-0/namespaces/default/
↪perconaservermongodbs/my-cluster-name",
   "uid":"5207e71a-c83f-4707-b892-63aa93fb615c"
},
"spec":{
   "allowUnsafeConfigurations":false,
   "backup":{
      "enabled":true,
```


(continues on next page)

(continued from previous page)


```
            "image":"percona/percona-server-mongodb-operator:1.5.0-backup",
            "restartOnFailure":true,
            "serviceAccountName":"percona-server-mongodb-operator",
            "storages":null,
            "tasks":null
         },
         "image":"percona/percona-server-mongodb:4.2.8-8",
         "imagePullPolicy":"Always",
         "mongod":{
            "net":{
               "hostPort":0,
               "port":27017
            },
            "operationProfiling":{
               "mode":"slowOp",
               "rateLimit":100,
               "slowOpThresholdMs":100
            },
            "security":{
               "enableEncryption":true,
               "encryptionCipherMode":"AES256-CBC",
               "encryptionKeySecret":"my-cluster-name-mongodb-encryption-key",
               "redactClientLogData":false
            },
            "setParameter":{
               "ttlMonitorSleepSecs":60,
               "wiredTigerConcurrentReadTransactions":128,
               "wiredTigerConcurrentWriteTransactions":128
            },
            "storage":{
               "engine":"wiredTiger",
               "inMemory":{
                  "engineConfig":{
                     "inMemorySizeRatio":0.9
                  }
               },
               "mmapv1":{
                  "nsSize":16,
                  "smallfiles":false
               },
               "wiredTiger":{
                  "collectionConfig":{
                     "blockCompressor":"snappy"
                  },
                  "engineConfig":{
                     "cacheSizeRatio":0.5,
                     "directoryForIndexes":false,
                     "journalCompressor":"snappy"
                  },
                  "indexConfig":{
                     "prefixCompression":true
                  }
               }
            }
         },
         "pmm":{
            "enabled":false,
```


(continues on next page)

(continued from previous page)

```
        "image":"percona/percona-server-mongodb-operator:1.5.0-pmm",
        "serverHost":"monitoring-service"
      },
      "replsets":[
        {
          "affinity":{
            "antiAffinityTopologyKey":"none"
          },
          "arbiter":{
            "affinity":{
              "antiAffinityTopologyKey":"none"
            },
            "enabled":false,
            "size":1
          },
          "expose":{
            "enabled":false,
            "exposeType":"LoadBalancer"
          },
          "name":"rs0",
          "podDisruptionBudget":{
            "maxUnavailable":1
          },
          "resources":{
            "limits":null
          },
          "size":3,
          "volumeSpec":{
            "persistentVolumeClaim":{
              "accessModes":[
                "ReadWriteOnce"
              ],
              "resources":{
                "requests":{
                  "storage":"3Gi"
                }
              },
              "storageClassName":"standard"
            }
          }
        }
      ],
      "secrets":{
        "users":"my-cluster-name-secrets"
      },
      "updateStrategy":"SmartUpdate"
    }
}
```

28.3 List Percona Server for MongoDB clusters

Description:

```
Lists all Percona Server for MongoDB clusters that exist in your kubernetes cluster
```

Kubectl Command:

```
kubectl get psmdb
```

URL:

```
https://$API_SERVER/apis/psmdb.percona.com/v1/namespaces/default/
↪perconaservermongodbs?limit=500
```

Authentication:

```
Authorization: Bearer $KUBE_TOKEN
```

cURL Request:

```
curl -k -v -XGET "https://$API_SERVER/apis/psmdb.percona.com/v1/namespaces/default/
↪perconaservermongodbs?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/psmdb.percona.com/v1/namespaces/default/perconaservermongodbs
↪",
      "resourceVersion":"1273793"
   },
   "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",
         "priority":0
      },
      {
         "name":"Status",
```

(continues on next page)

(continued from previous page)


```
            "type":"string",
            "format":"",
            "description":"Custom resource definition column (in JSONPath format): .
↪status.state",
            "priority":0
         },
         {
            "name":"Age",
            "type":"date",
            "format":"",
            "description":"Custom resource definition column (in JSONPath format): .
↪metadata.creationTimestamp",
            "priority":0
         }
      ],
      "rows":[
         {
            "cells":[
               "my-cluster-name",
               "ready",
               "37m"
            ],
            "object":{
               "kind":"PartialObjectMetadata",
               "apiVersion":"meta.k8s.io/v1",
               "metadata":{
                  "name":"my-cluster-name",
                  "namespace":"default",
                  "selfLink":"/apis/psmdb.percona.com/v1/namespaces/default/
↪perconaservermongodbs/my-cluster-name",
                  "uid":"5207e71a-c83f-4707-b892-63aa93fb615c",
                  "resourceVersion":"1273788",
                  "generation":1,
                  "creationTimestamp":"2020-07-24T14:27:58Z",
                  "annotations":{
                     "kubectl.kubernetes.io/last-applied-configuration":"{\"apiVersion\
↪":\"psmdb.percona.com/v1-5-0\",\"kind\":\"PerconaServerMongoDB\",\"metadata\":{\
↪"annotations\":{},\"name\":\"my-cluster-name\",\"namespace\":\"default\"},\"spec\":
↪{\"allowUnsafeConfigurations\":false,\"backup\":{\"enabled\":true,\"image\":\
↪"percona/percona-server-mongodb-operator:1.5.0-backup\",\"restartOnFailure\":true,\
↪"serviceAccountName\":\"percona-server-mongodb-operator\",\"storages\":null,\"tasks\
↪":null},\"image\":\"percona/percona-server-mongodb:4.2.8-8\",\"imagePullPolicy\":\
↪"Always\",\"mongod\":{\"net\":{\"hostPort\":0,\"port\":27017},\"operationProfiling\
↪":{\"mode\":\"slowOp\",\"rateLimit\":100,\"slowOpThresholdMs\":100},\"security\":{\
↪"enableEncryption\":true,\"encryptionCipherMode\":\"AES256-CBC\",\
↪"encryptionKeySecret\":\"my-cluster-name-mongodb-encryption-key\",\
↪"redactClientLogData\":false},\"setParameter\":{\"ttlMonitorSleepSecs\":60,\
↪"wiredTigerConcurrentReadTransactions\":128,\"wiredTigerConcurrentWriteTransactions\
↪":128},\"storage\":{\"engine\":\"wiredTiger\",\"inMemory\":{\"engineConfig\":{\
↪"inMemorySizeRatio\":0.9}},\"mmapv1\":{\"nsSize\":16,\"smallfiles\":false},\
↪"wiredTiger\":{\"collectionConfig\":{\"blockCompressor\":\"snappy\"},\"engineConfig\
↪":{\"cacheSizeRatio\":0.5,\"directoryForIndexes\":false,\"journalCompressor\":\
↪"snappy\"},\"indexConfig\":{\"prefixCompression\":true}}}},\"pmm\":{\"enabled\
↪":false,\"image\":\"percona/percona-server-mongodb-operator:1.5.0-pmm\",\
↪"serverHost\":\"monitoring-service\"},\"replsets\":[{\"affinity\":{\
↪"antiAffinityTopologyKey\":\"none\"},\"arbiter\":{\"affinity\":{\
↪"antiAffinityTopologyKey\":\"none\"},\"enabled\":false,\"size\":1},\"expose\":{\
↪"enabled\":false,\"exposeType\":\"LoadBalancer\"},\"name\":\"rs0\",\
↪"podDisruptionBudget\":{\"maxUnavailable\":1},\"resources\":{\"limits\":null},\
↪"size\":3,\"volumeSpec\":{\"persistentVolumeClaim\":{\"accessModes\":[\
↪"ReadWriteOnce\"],\"resources\":{\"requests\":{\"storage\":\"3Gi\"}},\
↪"storageClassName\":\"standard\"}}}],\"secrets\":{\"users\":\"my-cluster-name-
↪secrets\"},\"updateStrategy\":\"SmartUpdate\"}}\n"
```


(continues on next page)

(continued from previous page)


```
        },
        "managedFields": [
            {
                "manager": "kubectl",
                "operation": "Update",
                "apiVersion": "psmdb.percona.com/v1-5-0",
                "time": "2020-07-24T14:27:58Z",
                "fieldsType": "FieldsV1",
                "fieldsV1": {
                    "f:metadata": {
                        "f:annotations": {
                            ".": {

                            },
                            "f:kubectl.kubernetes.io/last-applied-configuration": {

                            }
                        }
                    },
                    "f:spec": {
                        ".": {

                        },
                        "f:allowUnsafeConfigurations": {

                        },
                        "f:backup": {
                            ".": {

                            },
                            "f:enabled": {

                            },
                            "f:image": {

                            },
                            "f:serviceAccountName": {

                            }
                        },
                        "f:image": {

                        },
                        "f:imagePullPolicy": {

                        },
                        "f:mongod": {
                            ".": {

                            },
                            "f:net": {
                                ".": {

                                },
                                "f:port": {

                                }
```


(continues on next page)

(continued from previous page)


```
                    },
                    "f:operationProfiling":{
                        ".":{

                        },
                        "f:mode":{

                        },
                        "f:rateLimit":{

                        },
                        "f:slowOpThresholdMs":{

                        }
                    },
                    "f:security":{
                        ".":{

                        },
                        "f:enableEncryption":{

                        },
                        "f:encryptionCipherMode":{

                        },
                        "f:encryptionKeySecret":{

                        }
                    },
                    "f:setParameter":{
                        ".":{

                        },
                        "f:ttlMonitorSleepSecs":{

                        },
                        "f:wiredTigerConcurrentReadTransactions":{

                        },
                        "f:wiredTigerConcurrentWriteTransactions":{

                        }
                    },
                    "f:storage":{
                        ".":{

                        },
                        "f:engine":{

                        },
                        "f:inMemory":{
                            ".":{

                            },
                            "f:engineConfig":{
                                ".":{
```


(continues on next page)

(continued from previous page)


```
                    },
                    "f:inMemorySizeRatio":{

                    }
                }
            },
            "f:mmapv1":{
                ".":{

                },
                "f:nsSize":{

                }
            },
            "f:wiredTiger":{
                ".":{

                },
                "f:collectionConfig":{
                    ".":{

                    },
                    "f:blockCompressor":{

                    }
                },
                "f:engineConfig":{
                    ".":{

                    },
                    "f:cacheSizeRatio":{

                    },
                    "f:journalCompressor":{

                    }
                },
                "f:indexConfig":{
                    ".":{

                    },
                    "f:prefixCompression":{

                    }
                }
            }
        }
    },
    "f:pmm":{
        ".":{

        },
        "f:image":{

        },
        "f:serverHost":{
```


(continues on next page)

(continued from previous page)


```
                        }
                    },
                    "f:secrets":{
                        ".":{

                        },
                        "f:users":{

                        }
                    },
                    "f:updateStrategy":{

                    }
                }
            }
        },
        {
            "manager":"percona-server-mongodb-operator",
            "operation":"Update",
            "apiVersion":"psmdb.percona.com/v1",
            "time":"2020-07-24T15:04:55Z",
            "fieldsType":"FieldsV1",
            "fieldsV1":{
                "f:spec":{
                    "f:backup":{
                        "f:containerSecurityContext":{
                            ".":{

                            },
                            "f:runAsNonRoot":{

                            },
                            "f:runAsUser":{

                            }
                        },
                        "f:podSecurityContext":{
                            ".":{

                            },
                            "f:fsGroup":{

                            }
                        }
                    },
                    "f:clusterServiceDNSSuffix":{

                    },
                    "f:replsets":{

                    },
                    "f:runUid":{

                    },
                    "f:secrets":{
                        "f:ssl":{
```


(continues on next page)

(continued from previous page)


```
                                    },
                                    "f:sslInternal":{

                                    }
                                }
                            },
                            "f:status":{
                                ".":{

                                },
                                "f:conditions":{

                                },
                                "f:observedGeneration":{

                                },
                                "f:replsets":{
                                    ".":{

                                    },
                                    "f:rs0":{
                                        ".":{

                                        },
                                        "f:ready":{

                                        },
                                        "f:size":{

                                        },
                                        "f:status":{

                                        }
                                    }
                                },
                                "f:state":{

                                }
                            }
                        }
                    ]
                }
            }
        }
    ]
}
```

28.4 Get status of Percona Server for MongoDB cluster

Description:

```
Gets all information about specified Percona Server for MongoDB cluster
```

Kubectl Command:

```
kubectl get psmdb/my-cluster-name -o json
```

URL:

```
https://$API_SERVER/apis/psmdb.percona.com/v1/namespaces/default/
↪perconaservermongodbs/my-cluster-name
```

Authentication:

```
Authorization: Bearer $KUBE_TOKEN
```

cURL Request:

```
curl -k -v -XGET "https://$API_SERVER/apis/psmdb.percona.com/v1/namespaces/default/
↪perconaservermongodbs/my-cluster-name" \
            -H "Accept: application/json" \
            -H "Authorization: Bearer $KUBE_TOKEN"
```

Request Body:

```
None
```

Response:

JSON:


```
{
   "apiVersion":"psmdb.percona.com/v1",
   "kind":"PerconaServerMongoDB",
   "metadata":{
      "annotations":{
         "kubectl.kubernetes.io/last-applied-configuration":"{\"apiVersion\":\"psmdb.
↪percona.com/v1-5-0\",\"kind\":\"PerconaServerMongoDB\",\"metadata\":{\"annotations\
↪":{},\"name\":\"my-cluster-name\",\"namespace\":\"default\"},\"spec\":{\
↪"allowUnsafeConfigurations\":false,\"backup\":{\"enabled\":true,\"image\":\"percona/
↪percona-server-mongodb-operator:1.5.0-backup\",\"restartOnFailure\":true,\
↪"serviceAccountName\":\"percona-server-mongodb-operator\",\"storages\":null,\"tasks\
↪":null},\"image\":\"percona/percona-server-mongodb:4.2.8-8\",\"imagePullPolicy\":\
↪"Always\",\"mongod\":{\"net\":{\"hostPort\":0,\"port\":27017},\"operationProfiling\
↪":{\"mode\":\"slowOp\",\"rateLimit\":100,\"slowOpThresholdMs\":100},\"security\":{\
↪"enableEncryption\":true,\"encryptionCipherMode\":\"AES256-CBC\",\
↪"encryptionKeySecret\":\"my-cluster-name-mongodb-encryption-key\",\
↪"redactClientLogData\":false},\"setParameter\":{\"ttlMonitorSleepSecs\":60,\
↪"wiredTigerConcurrentReadTransactions\":128,\"wiredTigerConcurrentWriteTransactions\
↪":128},\"storage\":{\"engine\":\"wiredTiger\",\"inMemory\":{\"engineConfig\":{\
↪"inMemorySizeRatio\":0.9}},\"mmapv1\":{\"nsSize\":16,\"smallfiles\":false},\
↪"wiredTiger\":{\"collectionConfig\":{\"blockCompressor\":\"snappy\"},\"engineConfig\
↪":{\"cacheSizeRatio\":0.5,\"directoryForIndexes\":false,\"journalCompressor\":\
↪"snappy\"},\"indexConfig\":{\"prefixCompression\":true}}}},\"pmm\":{\"enabled\
↪":false,\"image\":\"percona/percona-server-mongodb-operator:1.5.0-pmm\",\
↪"serverHost\":\"monitoring-service\"},\"replsets\":[{\"affinity\":{\
↪"antiAffinityTopologyKey\":\"none\"},\"arbiter\":{\"affinity\":{\
↪"antiAffinityTopologyKey\":\"none\"},\"enabled\":false,\"size\":1},\"expose\":{\
↪"enabled\":false,\"exposeType\":\"LoadBalancer\"},\"name\":\"rs0\",\
↪"podDisruptionBudget\":{\"maxUnavailable\":1},\"resources\":{\"limits\":null},\
↪"size\":3,\"volumeSpec\":{\"persistentVolumeClaim\":{\"accessModes\":[\
↪"ReadWriteOnce\"],\"resources\":{\"requests\":{\"storage\":\"3Gi\"}},\
```


(continues on next page)

(continued from previous page)


```
    },
    "creationTimestamp":"2020-07-24T14:27:58Z",
    "generation":1,
    "managedFields":[
       {
          "apiVersion":"psmdb.percona.com/v1-5-0",
          "fieldsType":"FieldsV1",
          "fieldsV1":{
             "f:metadata":{
                "f:annotations":{
                   ".":{

                   },
                   "f:kubectl.kubernetes.io/last-applied-configuration":{

                   }
                }
             },
             "f:spec":{
                ".":{

                },
                "f:allowUnsafeConfigurations":{

                },
                "f:backup":{
                   ".":{

                   },
                   "f:enabled":{

                   },
                   "f:image":{

                   },
                   "f:serviceAccountName":{

                   }
                },
                "f:image":{

                },
                "f:imagePullPolicy":{

                },
                "f:mongod":{
                   ".":{

                   },
                   "f:net":{
                      ".":{

                      },
                      "f:port":{

                      }
                   },
```


(continues on next page)

(continued from previous page)


```
                "f:operationProfiling":{
                    ".":{

                    },
                    "f:mode":{

                    },
                    "f:rateLimit":{

                    },
                    "f:slowOpThresholdMs":{

                    }
                },
                "f:security":{
                    ".":{

                    },
                    "f:enableEncryption":{

                    },
                    "f:encryptionCipherMode":{

                    },
                    "f:encryptionKeySecret":{

                    }
                },
                "f:setParameter":{
                    ".":{

                    },
                    "f:ttlMonitorSleepSecs":{

                    },
                    "f:wiredTigerConcurrentReadTransactions":{

                    },
                    "f:wiredTigerConcurrentWriteTransactions":{

                    }
                },
                "f:storage":{
                    ".":{

                    },
                    "f:engine":{

                    },
                    "f:inMemory":{
                        ".":{

                        },
                        "f:engineConfig":{
                            ".":{

                            },
```


(continues on next page)

(continued from previous page)


```
                        "f:inMemorySizeRatio":{

                        }
                    }
                },
                "f:mmapv1":{
                    ".":{

                    },
                    "f:nsSize":{

                    }
                },
                "f:wiredTiger":{
                    ".":{

                    },
                    "f:collectionConfig":{
                        ".":{

                        },
                        "f:blockCompressor":{

                        }
                    },
                    "f:engineConfig":{
                        ".":{

                        },
                        "f:cacheSizeRatio":{

                        },
                        "f:journalCompressor":{

                        }
                    },
                    "f:indexConfig":{
                        ".":{

                        },
                        "f:prefixCompression":{

                        }
                    }
                }
            },
            "f:pmm":{
                ".":{

                },
                "f:image":{

                },
                "f:serverHost":{

                }
```


(continues on next page)

(continued from previous page)


```
                    },
                    "f:secrets":{
                        ".":{

                        },
                        "f:users":{

                        }
                    },
                    "f:updateStrategy":{

                    }
                }
            },
            "manager":"kubectl",
            "operation":"Update",
            "time":"2020-07-24T14:27:58Z"
        },
        {
            "apiVersion":"psmdb.percona.com/v1",
            "fieldsType":"FieldsV1",
            "fieldsV1":{
                "f:spec":{
                    "f:backup":{
                        "f:containerSecurityContext":{
                            ".":{

                            },
                            "f:runAsNonRoot":{

                            },
                            "f:runAsUser":{

                            }
                        },
                        "f:podSecurityContext":{
                            ".":{

                            },
                            "f:fsGroup":{

                            }
                        }
                    },
                    "f:clusterServiceDNSSuffix":{

                    },
                    "f:replsets":{

                    },
                    "f:runUid":{

                    },
                    "f:secrets":{
                        "f:ssl":{

                        },
```


(continues on next page)

(continued from previous page)


```
                    "f:sslInternal":{

                    }
                }
            },
            "f:status":{
                ".":{

                },
                "f:conditions":{

                },
                "f:observedGeneration":{

                },
                "f:replsets":{
                    ".":{

                    },
                    "f:rs0":{
                        ".":{

                        },
                        "f:ready":{

                        },
                        "f:size":{

                        },
                        "f:status":{

                        }
                    }
                },
                "f:state":{

                }
            }
        },
        "manager":"percona-server-mongodb-operator",
        "operation":"Update",
        "time":"2020-07-24T15:09:40Z"
      }
    ],
    "name":"my-cluster-name",
    "namespace":"default",
    "resourceVersion":"1274523",
    "selfLink":"/apis/psmdb.percona.com/v1/namespaces/default/perconaservermongodbs/
↪my-cluster-name",
    "uid":"5207e71a-c83f-4707-b892-63aa93fb615c"
  },
  "spec":{
    "allowUnsafeConfigurations":false,
    "backup":{
      "enabled":true,
      "image":"percona/percona-server-mongodb-operator:1.5.0-backup",
      "restartOnFailure":true,
```


(continues on next page)

(continued from previous page)


```
        "serviceAccountName":"percona-server-mongodb-operator",
        "storages":null,
        "tasks":null
    },
    "image":"percona/percona-server-mongodb:4.2.8-8",
    "imagePullPolicy":"Always",
    "mongod":{
        "net":{
            "hostPort":0,
            "port":27017
        },
        "operationProfiling":{
            "mode":"slowOp",
            "rateLimit":100,
            "slowOpThresholdMs":100
        },
        "security":{
            "enableEncryption":true,
            "encryptionCipherMode":"AES256-CBC",
            "encryptionKeySecret":"my-cluster-name-mongodb-encryption-key",
            "redactClientLogData":false
        },
        "setParameter":{
            "ttlMonitorSleepSecs":60,
            "wiredTigerConcurrentReadTransactions":128,
            "wiredTigerConcurrentWriteTransactions":128
        },
        "storage":{
            "engine":"wiredTiger",
            "inMemory":{
                "engineConfig":{
                    "inMemorySizeRatio":0.9
                }
            },
            "mmapv1":{
                "nsSize":16,
                "smallfiles":false
            },
            "wiredTiger":{
                "collectionConfig":{
                    "blockCompressor":"snappy"
                },
                "engineConfig":{
                    "cacheSizeRatio":0.5,
                    "directoryForIndexes":false,
                    "journalCompressor":"snappy"
                },
                "indexConfig":{
                    "prefixCompression":true
                }
            }
        }
    },
    "pmm":{
        "enabled":false,
        "image":"percona/percona-server-mongodb-operator:1.5.0-pmm",
        "serverHost":"monitoring-service"
```


(continues on next page)

(continued from previous page)


```
    },
    "replsets":[
       {
          "affinity":{
             "antiAffinityTopologyKey":"none"
          },
          "arbiter":{
             "affinity":{
                "antiAffinityTopologyKey":"none"
             },
             "enabled":false,
             "size":1
          },
          "expose":{
             "enabled":false,
             "exposeType":"LoadBalancer"
          },
          "name":"rs0",
          "podDisruptionBudget":{
             "maxUnavailable":1
          },
          "resources":{
             "limits":null
          },
          "size":3,
          "volumeSpec":{
             "persistentVolumeClaim":{
                "accessModes":[
                   "ReadWriteOnce"
                ],
                "resources":{
                   "requests":{
                      "storage":"3Gi"
                   }
                },
                "storageClassName":"standard"
             }
          }
       }
    ],
    "secrets":{
       "users":"my-cluster-name-secrets"
    },
    "updateStrategy":"SmartUpdate"
 },
 "status":{
    "conditions":[
       {
          "lastTransitionTime":"2020-07-24T14:28:03Z",
          "status":"True",
          "type":"ClusterInitializing"
       },
       {
          "lastTransitionTime":"2020-07-24T14:28:39Z",
          "status":"True",
          "type":"Error"
       },
```


(continues on next page)

(continued from previous page)


```
        {
            "lastTransitionTime":"2020-07-24T14:28:41Z",
            "status":"True",
            "type":"ClusterInitializing"
        },
        {
            "lastTransitionTime":"2020-07-24T14:28:41Z",
            "status":"True",
            "type":"Error"
        },
        {
            "lastTransitionTime":"2020-07-24T14:29:10Z",
            "status":"True",
            "type":"ClusterReady"
        },
        {
            "lastTransitionTime":"2020-07-24T14:49:46Z",
            "status":"True",
            "type":"ClusterInitializing"
        },
        {
            "lastTransitionTime":"2020-07-24T14:50:00Z",
            "status":"True",
            "type":"ClusterInitializing"
        },
        {
            "lastTransitionTime":"2020-07-24T14:52:31Z",
            "status":"True",
            "type":"ClusterInitializing"
        },
        {
            "lastTransitionTime":"2020-07-24T14:52:43Z",
            "status":"True",
            "type":"Error"
        },
        {
            "lastTransitionTime":"2020-07-24T14:53:01Z",
            "status":"True",
            "type":"ClusterInitializing"
        },
        {
            "lastTransitionTime":"2020-07-24T14:53:05Z",
            "status":"True",
            "type":"ClusterInitializing"
        },
        {
            "lastTransitionTime":"2020-07-24T14:53:05Z",
            "status":"True",
            "type":"ClusterReady"
        }
    ],
    "observedGeneration":1,
    "replsets":{
        "rs0":{
            "ready":3,
            "size":3,
            "status":"ready"
```


(continues on next page)

(continued from previous page)

```
        }
      },
      "state":"ready"
   }
}
```

28.5 Scale up/down Percona Server for MongoDB cluster

Description:

```
Increase or decrease the size of the Percona Server for MongoDB cluster nodes to fit␣
↪the current high availability needs
```

Kubectl Command:

```
kubectl patch psmdb my-cluster-name --type=merge --patch '{
"spec": {"replsets":{ "size": "5" }
}}'
```

URL:

```
https://$API_SERVER/apis/psmdb.percona.com/v1/namespaces/default/
↪perconaservermongodbs/my-cluster-name
```

Authentication:

```
Authorization: Bearer $KUBE_TOKEN
```

cURL Request:

```
curl -k -v -XPATCH "https://$API_SERVER/apis/psmdb.percona.com/v1/namespaces/default/
↪perconaservermongodbs/my-cluster-name" \
            -H "Authorization: Bearer $KUBE_TOKEN" \
            -H "Content-Type: application/merge-patch+json"
            -H "Accept: application/json" \
            -d '{
                  "spec": {"replsets":{ "size": "5" }
                  }}'
```

Request Body:

JSON:

```
{
"spec": {"replsets":{ "size": "5" }
}}
```

Input:

spec:

replsets

1. size (Int or String, Defaults: 3): `Specifiy the sie of the replsets cluster to scale up or down to`

Response:

JSON:


```
{
   "apiVersion":"psmdb.percona.com/v1",
   "kind":"PerconaServerMongoDB",
   "metadata":{
      "annotations":{
         "kubectl.kubernetes.io/last-applied-configuration":"{\"apiVersion\":\"psmdb.
↪percona.com/v1-5-0\",\"kind\":\"PerconaServerMongoDB\",\"metadata\":{\"annotations\
↪":{},\"name\":\"my-cluster-name\",\"namespace\":\"default\"},\"spec\":{\
↪"allowUnsafeConfigurations\":false,\"backup\":{\"enabled\":true,\"image\":\"percona/
↪percona-server-mongodb-operator:1.5.0-backup\",\"restartOnFailure\":true,\
↪"serviceAccountName\":\"percona-server-mongodb-operator\",\"storages\":null,\"tasks\
↪":null},\"image\":\"percona/percona-server-mongodb:4.2.8-8\",\"imagePullPolicy\":\
↪"Always\",\"mongod\":{\"net\":{\"hostPort\":0,\"port\":27017},\"operationProfiling\
↪":{\"mode\":\"slowOp\",\"rateLimit\":100,\"slowOpThresholdMs\":100},\"security\":{\
↪"enableEncryption\":true,\"encryptionCipherMode\":\"AES256-CBC\",\
↪"encryptionKeySecret\":\"my-cluster-name-mongodb-encryption-key\",\
↪"redactClientLogData\":false},\"setParameter\":{\"ttlMonitorSleepSecs\":60,\
↪"wiredTigerConcurrentReadTransactions\":128,\"wiredTigerConcurrentWriteTransactions\
↪":128},\"storage\":{\"engine\":\"wiredTiger\",\"inMemory\":{\"engineConfig\":{\
↪"inMemorySizeRatio\":0.9}},\"mmapv1\":{\"nsSize\":16,\"smallfiles\":false},\
↪"wiredTiger\":{\"collectionConfig\":{\"blockCompressor\":\"snappy\"},\"engineConfig\
↪":{\"cacheSizeRatio\":0.5,\"directoryForIndexes\":false,\"journalCompressor\":\
↪"snappy\"},\"indexConfig\":{\"prefixCompression\":true}}}},\"pmm\":{\"enabled\
↪":false,\"image\":\"percona/percona-server-mongodb-operator:1.5.0-pmm\",\
↪"serverHost\":\"monitoring-service\"},\"replsets\":[{\"affinity\":{\
↪"antiAffinityTopologyKey\":\"none\"},\"arbiter\":{\"affinity\":{\
↪"antiAffinityTopologyKey\":\"none\"},\"enabled\":false,\"size\":1},\"expose\":{\
↪"enabled\":false,\"exposeType\":\"LoadBalancer\"},\"name\":\"rs0\",\
↪"podDisruptionBudget\":{\"maxUnavailable\":1},\"resources\":{\"limits\":null},\
↪"size\":3,\"volumeSpec\":{\"persistentVolumeClaim\":{\"accessModes\":[\
↪"ReadWriteOnce\"],\"resources\":{\"requests\":{\"storage\":\"3Gi\"}},\
↪"storageClassName\":\"standard\"}}}],\"secrets\":{\"users\":\"my-cluster-name-
↪secrets\"},\"updateStrategy\":\"SmartUpdate\"}}\n"
      },
      "creationTimestamp":"2020-07-24T14:27:58Z",
      "generation":4,
      "managedFields":[
         {
            "apiVersion":"psmdb.percona.com/v1-5-0",
            "fieldsType":"FieldsV1",
            "fieldsV1":{
               "f:metadata":{
                  "f:annotations":{
                     ".":{

                     },
                     "f:kubectl.kubernetes.io/last-applied-configuration":{

                     }
                  }
               },
               "f:spec":{
                  ".":{
```


(continues on next page)

(continued from previous page)


```
                    },
                    "f:allowUnsafeConfigurations":{

                    },
                    "f:backup":{
                        ".":{

                        },
                        "f:enabled":{

                        },
                        "f:image":{

                        },
                        "f:serviceAccountName":{

                        }
                    },
                    "f:image":{

                    },
                    "f:imagePullPolicy":{

                    },
                    "f:mongod":{
                        ".":{

                        },
                        "f:net":{
                            ".":{

                            },
                            "f:port":{

                            }
                        },
                        "f:operationProfiling":{
                            ".":{

                            },
                            "f:mode":{

                            },
                            "f:rateLimit":{

                            },
                            "f:slowOpThresholdMs":{

                            }
                        },
                        "f:security":{
                            ".":{

                            },
                            "f:enableEncryption":{

                            },
```


(continues on next page)

(continued from previous page)


```
                    "f:encryptionCipherMode":{

                    },
                    "f:encryptionKeySecret":{

                    }
                },
                "f:setParameter":{
                    ".":{

                    },
                    "f:ttlMonitorSleepSecs":{

                    },
                    "f:wiredTigerConcurrentReadTransactions":{

                    },
                    "f:wiredTigerConcurrentWriteTransactions":{

                    }
                },
                "f:storage":{
                    ".":{

                    },
                    "f:engine":{

                    },
                    "f:inMemory":{
                        ".":{

                        },
                        "f:engineConfig":{
                            ".":{

                            },
                            "f:inMemorySizeRatio":{

                            }
                        }
                    },
                    "f:mmapv1":{
                        ".":{

                        },
                        "f:nsSize":{

                        }
                    },
                    "f:wiredTiger":{
                        ".":{

                        },
                        "f:collectionConfig":{
                            ".":{

                            },
```


(continues on next page)

(continued from previous page)


```
                            "f:blockCompressor":{

                            }
                        },
                        "f:engineConfig":{
                            ".":{

                            },
                            "f:cacheSizeRatio":{

                            },
                            "f:journalCompressor":{

                            }
                        },
                        "f:indexConfig":{
                            ".":{

                            },
                            "f:prefixCompression":{

                            }
                        }
                    }
                },
                "f:pmm":{
                    ".":{

                    },
                    "f:image":{

                    },
                    "f:serverHost":{

                    }
                },
                "f:secrets":{
                    ".":{

                    },
                    "f:users":{

                    }
                },
                "f:updateStrategy":{

                }
            }
        },
        "manager":"kubectl",
        "operation":"Update",
        "time":"2020-07-24T14:27:58Z"
    },
    {
        "apiVersion":"psmdb.percona.com/v1",
        "fieldsType":"FieldsV1",
```


(continues on next page)

(continued from previous page)

```
            "fieldsV1":{
               "f:spec":{
                  "f:backup":{
                     "f:containerSecurityContext":{
                        ".":{

                        },
                        "f:runAsNonRoot":{

                        },
                        "f:runAsUser":{

                        }
                     },
                     "f:podSecurityContext":{
                        ".":{

                        },
                        "f:fsGroup":{

                        }
                     }
                  },
                  "f:clusterServiceDNSSuffix":{

                  },
                  "f:runUid":{

                  },
                  "f:secrets":{
                     "f:ssl":{

                     },
                     "f:sslInternal":{

                     }
                  }
               },
               "f:status":{
                  ".":{

                  },
                  "f:conditions":{

                  },
                  "f:observedGeneration":{

                  },
                  "f:replsets":{
                     ".":{

                     },
                     "f:rs0":{
                        ".":{

                        },
                        "f:ready":{
```

(continues on next page)

(continued from previous page)


```
                        },
                        "f:size":{

                        },
                        "f:status":{

                        }
                     }
                  },
                  "f:state":{

                  }
               }
            },
            "manager":"percona-server-mongodb-operator",
            "operation":"Update",
            "time":"2020-07-24T15:35:14Z"
         },
         {
            "apiVersion":"psmdb.percona.com/v1",
            "fieldsType":"FieldsV1",
            "fieldsV1":{
               "f:spec":{
                  "f:replsets":{
                     ".":{

                     },
                     "f:size":{

                     }
                  }
               }
            },
            "manager":"kubectl",
            "operation":"Update",
            "time":"2020-07-24T15:43:19Z"
         }
      ],
      "name":"my-cluster-name",
      "namespace":"default",
      "resourceVersion":"1279009",
      "selfLink":"/apis/psmdb.percona.com/v1/namespaces/default/perconaservermongodbs/
↪my-cluster-name",
      "uid":"5207e71a-c83f-4707-b892-63aa93fb615c"
   },
   "spec":{
      "allowUnsafeConfigurations":false,
      "backup":{
         "enabled":true,
         "image":"percona/percona-server-mongodb-operator:1.5.0-backup",
         "restartOnFailure":true,
         "serviceAccountName":"percona-server-mongodb-operator",
         "storages":null,
         "tasks":null
      },
      "image":"percona/percona-server-mongodb:4.2.8-8",
```


(continues on next page)

(continued from previous page)

```
    "imagePullPolicy":"Always",
    "mongod":{
       "net":{
          "hostPort":0,
          "port":27017
       },
       "operationProfiling":{
          "mode":"slowOp",
          "rateLimit":100,
          "slowOpThresholdMs":100
       },
       "security":{
          "enableEncryption":true,
          "encryptionCipherMode":"AES256-CBC",
          "encryptionKeySecret":"my-cluster-name-mongodb-encryption-key",
          "redactClientLogData":false
       },
       "setParameter":{
          "ttlMonitorSleepSecs":60,
          "wiredTigerConcurrentReadTransactions":128,
          "wiredTigerConcurrentWriteTransactions":128
       },
       "storage":{
          "engine":"wiredTiger",
          "inMemory":{
             "engineConfig":{
                "inMemorySizeRatio":0.9
             }
          },
          "mmapv1":{
             "nsSize":16,
             "smallfiles":false
          },
          "wiredTiger":{
             "collectionConfig":{
                "blockCompressor":"snappy"
             },
             "engineConfig":{
                "cacheSizeRatio":0.5,
                "directoryForIndexes":false,
                "journalCompressor":"snappy"
             },
             "indexConfig":{
                "prefixCompression":true
             }
          }
       }
    },
    "pmm":{
       "enabled":false,
       "image":"percona/percona-server-mongodb-operator:1.5.0-pmm",
       "serverHost":"monitoring-service"
    },
    "replsets":{
       "size":"5"
    },
    "secrets":{
```

(continues on next page)

(continued from previous page)


```
        "users":"my-cluster-name-secrets"
      },
      "updateStrategy":"SmartUpdate"
   },
   "status":{
      "conditions":[
         {
            "lastTransitionTime":"2020-07-24T14:28:03Z",
            "status":"True",
            "type":"ClusterInitializing"
         },
         {
            "lastTransitionTime":"2020-07-24T14:28:39Z",
            "status":"True",
            "type":"Error"
         },
         {
            "lastTransitionTime":"2020-07-24T14:28:41Z",
            "status":"True",
            "type":"ClusterInitializing"
         },
         {
            "lastTransitionTime":"2020-07-24T14:28:41Z",
            "status":"True",
            "type":"Error"
         },
         {
            "lastTransitionTime":"2020-07-24T14:29:10Z",
            "status":"True",
            "type":"ClusterReady"
         },
         {
            "lastTransitionTime":"2020-07-24T14:49:46Z",
            "status":"True",
            "type":"ClusterInitializing"
         },
         {
            "lastTransitionTime":"2020-07-24T14:50:00Z",
            "status":"True",
            "type":"ClusterInitializing"
         },
         {
            "lastTransitionTime":"2020-07-24T14:52:31Z",
            "status":"True",
            "type":"ClusterInitializing"
         },
         {
            "lastTransitionTime":"2020-07-24T14:52:43Z",
            "status":"True",
            "type":"Error"
         },
         {
            "lastTransitionTime":"2020-07-24T14:53:01Z",
            "status":"True",
            "type":"ClusterInitializing"
         },
         {
```


(continues on next page)

(continued from previous page)

```
            "lastTransitionTime":"2020-07-24T14:53:05Z",
            "status":"True",
            "type":"ClusterInitializing"
         },
         {
            "lastTransitionTime":"2020-07-24T14:53:05Z",
            "status":"True",
            "type":"ClusterReady"
         }
      ],
      "observedGeneration":1,
      "replsets":{
         "rs0":{
            "ready":3,
            "size":3,
            "status":"ready"
         }
      },
      "state":"ready"
   }
}
```

28.6 Update Percona Server for MongoDB cluster image

Description:

```
Change the image of Percona Server for MongoDB containers inside the cluster
```

Kubectl Command:

```
kubectl patch psmdb my-cluster-name --type=merge --patch '{
"spec": {"psmdb":{ "image": "percona/percona-server-mongodb-operator:1.4.0-mongod4.2"␣
↪}
}}'
```

URL:

```
https://$API_SERVER/apis/psmdb.percona.com/v1/namespaces/default/
↪perconaservermongodbs/my-cluster-name
```

Authentication:

```
Authorization: Bearer $KUBE_TOKEN
```

cURL Request:

```
curl -k -v -XPATCH "https://$API_SERVER/apis/psmdb.percona.com/v1/namespaces/default/
↪perconaservermongodbs/my-cluster-name" \
            -H "Authorization: Bearer $KUBE_TOKEN" \
            -H "Accept: application/json" \
            -H "Content-Type: application/merge-patch+json"
            -d '{
```

(continues on next page)

(continued from previous page)

```
             "spec": {"psmdb":{ "image": "percona/percona-server-mongodb-operator:1.
↪4.0-mongod4.2" }
             }}'
```

Request Body:

JSON:

```
{
"spec": { "image ": "percona/percona-server-mongodb:4.2.8-8" }
}}
```

Input:

spec:

psmdb:

1. image (String, min-length: 1) : `name of the image to update for Percona Server for MongoDB`

Response:

JSON:

```
{
   "apiVersion":"psmdb.percona.com/v1",
   "kind":"PerconaServerMongoDB",
   "metadata":{
      "annotations":{
         "kubectl.kubernetes.io/last-applied-configuration":"{\"apiVersion\":\"psmdb.
↪percona.com/v1-5-0\",\"kind\":\"PerconaServerMongoDB\",\"metadata\":{\"annotations\
↪":{},\"name\":\"my-cluster-name\",\"namespace\":\"default\"},\"spec\":{\
↪"allowUnsafeConfigurations\":false,\"backup\":{\"enabled\":true,\"image\":\"percona/
↪percona-server-mongodb-operator:1.5.0-backup\",\"restartOnFailure\":true,\
↪"serviceAccountName\":\"percona-server-mongodb-operator\",\"storages\":null,\"tasks\
↪":null},\"image\":\"percona/percona-server-mongodb:4.2.8-8\",\"imagePullPolicy\":\
↪"Always\",\"mongod\":{\"net\":{\"hostPort\":0,\"port\":27017},\"operationProfiling\
↪":{\"mode\":\"slowOp\",\"rateLimit\":100,\"slowOpThresholdMs\":100},\"security\":{\
↪"enableEncryption\":true,\"encryptionCipherMode\":\"AES256-CBC\",\
↪"encryptionKeySecret\":\"my-cluster-name-mongodb-encryption-key\",\
↪"redactClientLogData\":false},\"setParameter\":{\"ttlMonitorSleepSecs\":60,\
↪"wiredTigerConcurrentReadTransactions\":128,\"wiredTigerConcurrentWriteTransactions\
↪":128},\"storage\":{\"engine\":\"wiredTiger\",\"inMemory\":{\"engineConfig\":{\
↪"inMemorySizeRatio\":0.9}},\"mmapv1\":{\"nsSize\":16,\"smallfiles\":false},\
↪"wiredTiger\":{\"collectionConfig\":{\"blockCompressor\":\"snappy\"},\"engineConfig\
↪":{\"cacheSizeRatio\":0.5,\"directoryForIndexes\":false,\"journalCompressor\":\
↪"snappy\"},\"indexConfig\":{\"prefixCompression\":true}}}},\"pmm\":{\"enabled\
↪":false,\"image\":\"percona/percona-server-mongodb-operator:1.5.0-pmm\",\
↪"serverHost\":\"monitoring-service\"},\"replsets\":[{\"affinity\":{\
↪"antiAffinityTopologyKey\":\"none\"},\"arbiter\":{\"affinity\":{\
↪"antiAffinityTopologyKey\":\"none\"},\"enabled\":false,\"size\":1},\"expose\":{\
↪"enabled\":false,\"exposeType\":\"LoadBalancer\"},\"name\":\"rs0\",\
↪"podDisruptionBudget\":{\"maxUnavailable\":1},\"resources\":{\"limits\":null},\
↪"size\":3,\"volumeSpec\":{\"persistentVolumeClaim\":{\"accessModes\":[\
↪"ReadWriteOnce\"],\"resources\":{\"requests\":{\"storage\":\"3Gi\"}},\
↪"storageClassName\":\"standard\"}}}],\"secrets\":{\"users\":\"my-cluster-name-
↪secrets\"},\"updateStrategy\":\"SmartUpdate\"}}\n"
      },
```

(continues on next page)

(continued from previous page)


```
    "creationTimestamp":"2020-07-24T14:27:58Z",
    "generation":5,
    "managedFields":[
       {
          "apiVersion":"psmdb.percona.com/v1-5-0",
          "fieldsType":"FieldsV1",
          "fieldsV1":{
             "f:metadata":{
                "f:annotations":{
                   ".":{

                   },
                   "f:kubectl.kubernetes.io/last-applied-configuration":{

                   }
                }
             },
             "f:spec":{
                ".":{

                },
                "f:allowUnsafeConfigurations":{

                },
                "f:backup":{
                   ".":{

                   },
                   "f:enabled":{

                   },
                   "f:image":{

                   },
                   "f:serviceAccountName":{

                   }
                },
                "f:image":{

                },
                "f:imagePullPolicy":{

                },
                "f:mongod":{
                   ".":{

                   },
                   "f:net":{
                      ".":{

                      },
                      "f:port":{

                      }
                   },
                   "f:operationProfiling":{
```


(continues on next page)

(continued from previous page)


```
            ".":{

            },
            "f:mode":{

            },
            "f:rateLimit":{

            },
            "f:slowOpThresholdMs":{

            }
         },
         "f:security":{
            ".":{

            },
            "f:enableEncryption":{

            },
            "f:encryptionCipherMode":{

            },
            "f:encryptionKeySecret":{

            }
         },
         "f:setParameter":{
            ".":{

            },
            "f:ttlMonitorSleepSecs":{

            },
            "f:wiredTigerConcurrentReadTransactions":{

            },
            "f:wiredTigerConcurrentWriteTransactions":{

            }
         },
         "f:storage":{
            ".":{

            },
            "f:engine":{

            },
            "f:inMemory":{
               ".":{

               },
               "f:engineConfig":{
                  ".":{

                  },
                  "f:inMemorySizeRatio":{
```


(continues on next page)

(continued from previous page)


```
                    }
                }
            },
            "f:mmapv1":{
                ".":{

                },
                "f:nsSize":{

                }
            },
            "f:wiredTiger":{
                ".":{

                },
                "f:collectionConfig":{
                    ".":{

                    },
                    "f:blockCompressor":{

                    }
                },
                "f:engineConfig":{
                    ".":{

                    },
                    "f:cacheSizeRatio":{

                    },
                    "f:journalCompressor":{

                    }
                },
                "f:indexConfig":{
                    ".":{

                    },
                    "f:prefixCompression":{

                    }
                }
            }
        }
    },
    "f:pmm":{
        ".":{

        },
        "f:image":{

        },
        "f:serverHost":{

        }
    },
```


(continues on next page)

(continued from previous page)


```
                    "f:secrets":{
                        ".":{

                        },
                        "f:users":{

                        }
                    },
                    "f:updateStrategy":{

                    }
                }
            },
            "manager":"kubectl",
            "operation":"Update",
            "time":"2020-07-24T14:27:58Z"
        },
        {
            "apiVersion":"psmdb.percona.com/v1",
            "fieldsType":"FieldsV1",
            "fieldsV1":{
                "f:spec":{
                    "f:backup":{
                        "f:containerSecurityContext":{
                            ".":{

                            },
                            "f:runAsNonRoot":{

                            },
                            "f:runAsUser":{

                            }
                        },
                        "f:podSecurityContext":{
                            ".":{

                            },
                            "f:fsGroup":{

                            }
                        }
                    },
                    "f:clusterServiceDNSSuffix":{

                    },
                    "f:runUid":{

                    },
                    "f:secrets":{
                        "f:ssl":{

                        },
                        "f:sslInternal":{

                        }
                    }
```


(continues on next page)

(continued from previous page)


```
            },
            "f:status":{
               ".":{

               },
               "f:conditions":{

               },
               "f:observedGeneration":{

               },
               "f:replsets":{
                  ".":{

                  },
                  "f:rs0":{
                     ".":{

                     },
                     "f:ready":{

                     },
                     "f:size":{

                     },
                     "f:status":{

                     }
                  }
               },
               "f:state":{

               }
            }
         },
         "manager":"percona-server-mongodb-operator",
         "operation":"Update",
         "time":"2020-07-24T15:35:14Z"
      },
      {
         "apiVersion":"psmdb.percona.com/v1",
         "fieldsType":"FieldsV1",
         "fieldsV1":{
            "f:spec":{
               "f:image ":{

               },
               "f:replsets":{
                  ".":{

                  },
                  "f:size":{

                  }
               }
            }
         },
```


(continues on next page)

(continued from previous page)


```
            "manager":"kubectl",
            "operation":"Update",
            "time":"2020-07-27T12:21:39Z"
         }
      ],
      "name":"my-cluster-name",
      "namespace":"default",
      "resourceVersion":"1279853",
      "selfLink":"/apis/psmdb.percona.com/v1/namespaces/default/perconaservermongodbs/
↪my-cluster-name",
      "uid":"5207e71a-c83f-4707-b892-63aa93fb615c"
   },
   "spec":{
      "allowUnsafeConfigurations":false,
      "backup":{
         "enabled":true,
         "image":"percona/percona-server-mongodb-operator:1.5.0-backup",
         "restartOnFailure":true,
         "serviceAccountName":"percona-server-mongodb-operator",
         "storages":null,
         "tasks":null
      },
      "image ":"percona/percona-server-mongodb:4.2.8-8",
      "imagePullPolicy":"Always",
      "mongod":{
         "net":{
            "hostPort":0,
            "port":27017
         },
         "operationProfiling":{
            "mode":"slowOp",
            "rateLimit":100,
            "slowOpThresholdMs":100
         },
         "security":{
            "enableEncryption":true,
            "encryptionCipherMode":"AES256-CBC",
            "encryptionKeySecret":"my-cluster-name-mongodb-encryption-key",
            "redactClientLogData":false
         },
         "setParameter":{
            "ttlMonitorSleepSecs":60,
            "wiredTigerConcurrentReadTransactions":128,
            "wiredTigerConcurrentWriteTransactions":128
         },
         "storage":{
            "engine":"wiredTiger",
            "inMemory":{
               "engineConfig":{
                  "inMemorySizeRatio":0.9
               }
            },
            "mmapv1":{
               "nsSize":16,
               "smallfiles":false
            },
            "wiredTiger":{
```


(continues on next page)

(continued from previous page)


```
                "collectionConfig":{
                   "blockCompressor":"snappy"
                },
                "engineConfig":{
                   "cacheSizeRatio":0.5,
                   "directoryForIndexes":false,
                   "journalCompressor":"snappy"
                },
                "indexConfig":{
                   "prefixCompression":true
                }
             }
          }
       },
       "pmm":{
          "enabled":false,
          "image":"percona/percona-server-mongodb-operator:1.5.0-pmm",
          "serverHost":"monitoring-service"
       },
       "replsets":{
          "size":"5"
       },
       "secrets":{
          "users":"my-cluster-name-secrets"
       },
       "updateStrategy":"SmartUpdate"
    },
    "status":{
       "conditions":[
          {
             "lastTransitionTime":"2020-07-24T14:28:03Z",
             "status":"True",
             "type":"ClusterInitializing"
          },
          {
             "lastTransitionTime":"2020-07-24T14:28:39Z",
             "status":"True",
             "type":"Error"
          },
          {
             "lastTransitionTime":"2020-07-24T14:28:41Z",
             "status":"True",
             "type":"ClusterInitializing"
          },
          {
             "lastTransitionTime":"2020-07-24T14:28:41Z",
             "status":"True",
             "type":"Error"
          },
          {
             "lastTransitionTime":"2020-07-24T14:29:10Z",
             "status":"True",
             "type":"ClusterReady"
          },
          {
             "lastTransitionTime":"2020-07-24T14:49:46Z",
             "status":"True",
```


(continues on next page)

(continued from previous page)


```
            "type":"ClusterInitializing"
         },
         {
            "lastTransitionTime":"2020-07-24T14:50:00Z",
            "status":"True",
            "type":"ClusterInitializing"
         },
         {
            "lastTransitionTime":"2020-07-24T14:52:31Z",
            "status":"True",
            "type":"ClusterInitializing"
         },
         {
            "lastTransitionTime":"2020-07-24T14:52:43Z",
            "status":"True",
            "type":"Error"
         },
         {
            "lastTransitionTime":"2020-07-24T14:53:01Z",
            "status":"True",
            "type":"ClusterInitializing"
         },
         {
            "lastTransitionTime":"2020-07-24T14:53:05Z",
            "status":"True",
            "type":"ClusterInitializing"
         },
         {
            "lastTransitionTime":"2020-07-24T14:53:05Z",
            "status":"True",
            "type":"ClusterReady"
         }
      ],
      "observedGeneration":1,
      "replsets":{
         "rs0":{
            "ready":3,
            "size":3,
            "status":"ready"
         }
      },
      "state":"ready"
   }
}
```


28.7 Backup Percona Server for MongoDB cluster

Description:

```
Takes a backup of the Percona Server for MongoDB cluster containers data to be able␣
↪to recover from disasters or make a roll-back later
```

Kubectl Command:

```
kubectl apply -f percona-server-mongodb-operator/deploy/backup/backup.yaml
```

URL:

```
https://$API_SERVER/apis/psmdb.percona.com/v1/namespaces/default/
↪perconaservermongodbbackups
```

Authentication:

```
Authorization: Bearer $KUBE_TOKEN
```

cURL Request:

```
curl -k -v -XPOST "https://$API_SERVER/apis/psmdb.percona.com/v1/namespaces/default/
↪perconaservermongodbbackups" \
            -H "Accept: application/json" \
            -H "Content-Type: application/json" \
            -d "@backup.json" -H "Authorization: Bearer $KUBE_TOKEN"
```

Request Body (backup.json):

JSON:

```
{
   "apiVersion":"psmdb.percona.com/v1",
   "kind":"PerconaServerMongoDBBackup",
   "metadata":{
      "name":"backup1",
      "namespace":"default"
   },
   "spec":{
      "psmdbCluster":"my-cluster-name",
      "storageName":"s3-us-west"
   }
}
```

Input:

1. **metadata**:

   name(String, min-length:1) : `name of backup to create`

2. **spec**:

   1. psmdbCluster(String, min-length:1) : `name of Percona Server for MongoDB cluster`
   2. storageName(String, min-length:1) : `name of storage claim to use`

Response:

JSON:

```
{
   "apiVersion":"psmdb.percona.com/v1",
   "kind":"PerconaServerMongoDBBackup",
   "metadata":{
      "annotations":{
         "kubectl.kubernetes.io/last-applied-configuration":"{\"apiVersion\":\"psmdb.
↪percona.com/v1\",\"kind\":\"PerconaServerMongoDBBackup\",\"metadata\":{\
↪"annotations\":{},\"name\":\"backup1\",\"namespace\":\"default\"},\"spec\":{\
↪"psmdbCluster\":\"my-cluster-name\",\"storageName\":\"s3-us-west\"}}\n"
```

(continues on next page)

(continued from previous page)


```
      },
      "creationTimestamp":"2020-07-27T13:45:43Z",
      "generation":1,
      "managedFields":[
         {
            "apiVersion":"psmdb.percona.com/v1",
            "fieldsType":"FieldsV1",
            "fieldsV1":{
               "f:metadata":{
                  "f:annotations":{
                     ".":{

                     },
                     "f:kubectl.kubernetes.io/last-applied-configuration":{

                     }
                  }
               },
               "f:spec":{
                  ".":{

                  },
                  "f:psmdbCluster":{

                  },
                  "f:storageName":{

                  }
               }
            },
            "manager":"kubectl",
            "operation":"Update",
            "time":"2020-07-27T13:45:43Z"
         }
      ],
      "name":"backup1",
      "namespace":"default",
      "resourceVersion":"1290243",
      "selfLink":"/apis/psmdb.percona.com/v1/namespaces/default/
↪perconaservermongodbbackups/backup1",
      "uid":"e695d1c7-898e-44b0-b356-537284f6c046"
   },
   "spec":{
      "psmdbCluster":"my-cluster-name",
      "storageName":"s3-us-west"
   }
}
```

28.8 Restore Percona Server for MongoDB cluster

Description:

```
Restores Percona Server for MongoDB cluster data to an earlier version to recover␣
↪from a problem or to make a roll-back
```

Kubectl Command:

```
kubectl apply -f percona-server-mongodb-operator/deploy/backup/restore.yaml
```

URL:

```
https://$API_SERVER/apis/psmdb.percona.com/v1/namespaces/default/
↪perconaservermongodbrestores
```

Authentication:

```
Authorization: Bearer $KUBE_TOKEN
```

cURL Request:

```
curl -k -v -XPOST "https://$API_SERVER/apis/psmdb.percona.com/v1/namespaces/default/
↪perconaservermongodbrestores" \
            -H "Accept: application/json" \
            -H "Content-Type: application/json" \
            -d "@restore.json" \
            -H "Authorization: Bearer $KUBE_TOKEN"
```

Request Body (restore.json):

JSON:

```
{
   "apiVersion":"psmdb.percona.com/v1",
   "kind":"PerconaServerMongoDBRestore",
   "metadata":{
      "name":"restore1",
      "namespace":"default"
   },
   "spec":{
      "backupName":"backup1",
      "clusterName":"my-cluster-name"
   }
}
```

Input:

1. **metadata**:

   name(String, min-length:1): `name of restore to create`

2. **spec**:

   1. clusterName(String, min-length:1) : `name of Percona Server for MongoDB cluster`
   2. backupName(String, min-length:1) : `name of backup to restore from`

Response:

JSON:

```
{
   "apiVersion":"psmdb.percona.com/v1",
   "kind":"PerconaServerMongoDBRestore",
   "metadata":{
      "annotations":{
         "kubectl.kubernetes.io/last-applied-configuration":"{\"apiVersion\":\"psmdb.
↪percona.com/v1\",\"kind\":\"PerconaServerMongoDBRestore\",\"metadata\":{\
↪"annotations\":{},\"name\":\"restore1\",\"namespace\":\"default\"},\"spec\":{\
↪"backupName\":\"backup1\",\"clusterName\":\"my-cluster-name\"}}\n"
      },
      "creationTimestamp":"2020-07-27T13:52:56Z",
      "generation":1,
      "managedFields":[
         {
            "apiVersion":"psmdb.percona.com/v1",
            "fieldsType":"FieldsV1",
            "fieldsV1":{
               "f:metadata":{
                  "f:annotations":{
                     ".":{

                     },
                     "f:kubectl.kubernetes.io/last-applied-configuration":{

                     }
                  }
               },
               "f:spec":{
                  ".":{

                  },
                  "f:backupName":{

                  },
                  "f:clusterName":{

                  }
               }
            },
            "manager":"kubectl",
            "operation":"Update",
            "time":"2020-07-27T13:52:56Z"
         }
      ],
      "name":"restore1",
      "namespace":"default",
      "resourceVersion":"1291198",
      "selfLink":"/apis/psmdb.percona.com/v1/namespaces/default/
↪perconaservermongodbrestores/restore1",
      "uid":"17e982fe-ac41-47f4-afba-fea380b0c76e"
   },
   "spec":{
      "backupName":"backup1",
      "clusterName":"my-cluster-name"
   }
}
```

CHAPTER

TWENTYNINE

FREQUENTLY ASKED QUESTIONS

29.1 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 Kubernetes Operator for Percona Server for MongoDB provides out-of-the-box functionality to automate provisioning and management of highly available MongoDB database clusters on Kubernetes.

29.2 How can I contact the developers?

The best place to discuss Percona Kubernetes Operator for Percona Server for MongoDB with developers and other community members is the community forum.

If you would like to report a bug, use the Percona Kubernetes Operator for Percona Server for MongoDB project in JIRA.

29.3 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 Server for MongoDB Operator as a whole.

29.4 Which versions of MongoDB the Operator supports?

Percona Operator for Percona Server for MongoDB provides a ready-to-use installation of the MongoDB-based database cluster inside your Kubernetes installation. It works with Percona Server for MongoDB 3.6, 4.0, 4.2, and 4.4, and the exact version is determined by the Docker image in use.

Percona-certified Docker images used by the Operator are listed here. For example, Percona Server for MongoDB 4.4 is supported with the following recommended version: 4.4.5-7. More details on the exact Percona Server for MongoDB version can be found in the release notes (4.4, 4.2, 4.0, and 3.6).

29.5 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:
  replsets:
  - name: rs0
    ....
    sidecars:
    - image: busybox
      command: ["/bin/sh"]
      args: ["-c", "while true; do echo echo $(date -u) 'test' >> /dev/null; sleep 5;␣
↪done"]
      name: rs-sidecar-1
    ....
```

You can add `sidecars` subsection to `replsets`, `sharding.configsvrReplSet`, and `sharding.mongos` sections.

Note: Custom sidecar containers can easily access other components of your cluster. Therefore they should be used carefully and by experienced users only.

29.6 How to provoke the initial sync of a Pod

There are certain situations where it might be necessary to delete all MongoDB instance data to force the resync. For example, there may be the following reasons:

- rebuilding the node to defragment the database,
- recreating the member failing to sync due to some bug.

In the case of a "regular" MongoDB, wiping the dbpath would trigger such resync. In the case of a MongoDB cluster controlled by the Operator, you will need to do the following steps:

1. Find out the names of the Persistent Volume Claim and Pod you are going to delete (use `kubectl get pvc` command for PVC and `kubectl get pod` one for Pods).
2. Delete the appropriate PVC and Pod. For example, wiping out the `my-cluster-name-rs0-2` Pod should look as follows:

```
kubectl delete pod/my-cluster-name-rs0-2 pvc/mongod-data-my-cluster-name-rs0-2
```

The Operator will automatically recreate the needed Pod and PVC after deletion.

CHAPTER

THIRTY

KUBERNETES OPERATOR FOR PERCONA SERVER FOR MONGODB RELEASE NOTES

30.1 *Percona Kubernetes Operator for Percona Server for MongoDB 1.8.0*

Date May 6, 2021

Installation Installing Percona Kubernetes Operator for Percona Server for MongoDB

30.1.1 Release Highlights

- The support for *Point-in-time recovery* added in this release. Users can now recover to a specific date and time from operations logs stored on S3
- It is now possible to perform a *major version upgrade* for MongoDB (for example, upgrade 4.2 version to 4.4) with no manual steps

30.1.2 New Features

- K8SPSMDB-387: Add support for *point-in-time recovery* to recover to a specific date and time
- K8SPSMDB-284: Add support for automated major version MongoDB upgrades

30.1.3 Improvements

- K8SPSMDB-436: The imagePullPolicy option in the `deploy/cr.yaml` configuration file now is applied to init container as well
- K8SPSMDB-400: Simplify secret change logic to avoid Pod restarts when user changes the credentials
- K8SPSMDB-381: Get credentials directly from Secrets instead of the environment variables when initializing the Replica Set
- K8SPSMDB-352: Restrict running run less than 5 Pods of Replica Sets with enabled arbiter unless the `allowUnsafeConfigurations` option is set to true
- K8SPSMDB-332: Restrict running less than 3 Pods of Config Servers unless the `allowUnsafeConfigurations` option is set to true
- K8SPSMDB-331: Restrict running less than 3 mongos Pods unless the `allowUnsafeConfigurations` option is set to true

30.1.4 Bugs Fixed

- K8SPSMDB-384: Fix a bug due to which mongos Pods were failing readiness probes for some period of time during the cluster initialization
- K8SPSMDB-434: Fix a bug due to which nil pointer dereference error was occurring when switching the `sharding.enabled` option from false to true (thanks to srteam2020 for contributing)
- K8SPSMDB-430: Fix a bug due to which a stale apiserver could trigger undesired StatefulSet and PVC deletion when recreating the cluster with the same name (thanks to srteam2020 for contributing)
- K8SPSMDB-428: Fix a bug which caused mongos to fail in case of the empty name field in configsvrReplSet section of the Custom Resource
- K8SPSMDB-418: Fix a bug due to which `serviceAnnotations` changes in the `deploy/cr.yaml` file were not applied to the running cluster
- K8SPSMDB-364: Fix a bug where liveness probe of a mongo container was always failing if the userAdmin password contained special characters
- K8SPSMDB-43: Fix a bug due to which renaming Replica Set in the Custom Resource caused creating new Replica Set without deleting the old one

30.2 *Percona Kubernetes Operator for Percona Server for MongoDB 1.7.0*

Date March 8, 2021

Installation Installing Percona Kubernetes Operator for Percona Server for MongoDB

30.2.1 Release Highlights

- This release brings full support for the *Percona Server for MongoDB Sharding*. Sharding allows you to scale databases horizontally, distributing data across multiple MongoDB Pods, and so it is extremely useful for large data sets. By default of the `deploy/cr.yaml` configuration file contains only one replica set, but when you *turn sharding on*, you can add more replica sets with different names to the `replsets` section.
- It is now *possible* to clean up Persistent Volume Claims automatically after the cluster deletion event. This feature is off by default. Particularly it is useful to avoid leftovers in testing environments, where the cluster can be re-created and deleted many times. 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. The sidecar container can be added to *replsets*, *sharding.configsvrReplSet*, and *sharding.mongos* sections of the `deploy/cr.yaml` configuration file.

30.2.2 New Features

- K8SPSMDB-121: Add support for *sharding* to scale MongoDB cluster horizontally
- K8SPSMDB-294: Support for *custom sidecar container* to extend the Operator capabilities
- K8SPSMDB-260: Persistent Volume Claims *can now be automatically removed* after MongoDB cluster deletion

30.2.3 Improvements

- K8SPSMDB-335: Operator can now automatically remove old backups from S3 if *retention period* is set
- K8SPSMDB-330: Add support for runtimeClassName Kubernetes feature for selecting the container runtime
- K8SPSMDB-306: It is now possible to explicitly set the version of MongoDB for newly provisioned clusters. Before that, all new clusters were started with the latest MongoDB version if Version Service was enabled
- K8SPSMDB-370: Fix confusing log messages about no backup / restore found which were caused by Percona Backup for MongoDB waiting for the backup metadata
- K8SPSMDB-342: MongoDB container liveness probe will now use TLS to follow best practices and remove noisy log messages from mongod log

30.2.4 Bugs Fixed

- K8SPSMDB-346: Fix a bug which prevented adding/removing labels to Pods without downtime
- K8SPSMDB-366: Fix a bug which prevented enabling Percona Monitoring and Management (PMM) due to incorrect request for the recommended PMM Client image version to the Version Service
- K8SPSMDB-402: running multiple replica sets without sharding enabled should be prohibited
- K8SPSMDB-382: Fix a bug which caused mongos process to fail when using `allowUnsafeConfigurations=true`
- K8SPSMDB-362: Fix a bug due to which changing secrets in a single-shard mode caused mongos Pods to fail

30.3 *Percona Kubernetes Operator for Percona Server for MongoDB* 1.6.0

Date December 22, 2020

Installation Installing Percona Kubernetes Operator for Percona Server for MongoDB

30.3.1 New Features

- K8SPSMDB-273: Add support for `mongos` service to expose a single *shard* of a MongoDB cluster through one entry point instead of provisioning a load-balancer per replica set node. In the following release, we will add support for multiple shards.
- K8SPSMDB-282: 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.

30.3.2 Improvements

- K8SPSMDB-258: Add support for Percona Server for MongoDB version 4.4
- K8SPSMDB-319: Show Endpoint in the `kubectl get psmdb` command output to connect to a MongoDB cluster easily
- K8SPSMDB-257: Store the Operator version as a `crVersion` field in the `deploy/cr.yaml` configuration file
- K8SPSMDB-266: Use plain-text passwords instead of base64-encoded ones when creating *System Users* secrets for simplicity

30.3.3 Bugs Fixed

- K8SPSMDB-268: Fix a bug affecting the support of TLS certificates issued by cert-manager, due to which proper rights were not set for the role-based access control, and Kubernetes versions newer than 1.15 required other certificate issuing sources
- K8SPSMDB-261: Fix a bug due to which cluster pause/resume functionality didn't work in previous releases
- K8SPSMDB-292: Fix a bug due to which not all clusters managed by the Operator were upgraded by the automatic update

30.3.4 Removal

- The MMAPv1 storage engine is no longer supported for all MongoDB versions starting from this version of the Operator. MMAPv1 was already deprecated by MongoDB for a long time. WiredTiger is the default storage engine since MongoDB 3.2, and MMAPv1 was completely removed in MongoDB 4.2.

Note: Upgrade of the Operator from 1.5.0 to 1.6.0 will fail if MMAPv1 is used, but MongoDB cluster will continue to run. It is recommended to migrate your clusters to WiredTiger engine before the upgrade.

30.4 *Percona Kubernetes Operator for Percona Server for MongoDB 1.5.0*

Date September 7, 2020

Installation Installing Percona Kubernetes Operator for Percona Server for MongoDB

30.4.1 New Features

- K8SPSMDB-233: Automatic management of system users for MongoDB on password rotation via Secret
- K8SPSMDB-226: Official Helm chart for the Operator
- K8SPSMDB-199: Support multiple PSMDB minor versions by the Operator
- K8SPSMDB-198: Fully Automate Minor Version Updates (Smart Update)

30.4.2 Improvements

- K8SPSMDB-192: The ability to set the mongod cursorTimeoutMillis parameter in YAML (Thanks to user xprt64 for the contribution)
- K8SPSMDB-234: OpenShift 4.5 support
- K8SPSMDB-197: Additional certificate SANs useful for reverse DNS lookups (Thanks to user phin1x for the contribution)
- K8SPSMDB-190: Direct API quering with "curl" instead of using "kubectl" tool in scheduled backup jobs (Thanks to user phin1x for the contribution)
- K8SPSMDB-133: A special Percona Server for MongoDB debug image which avoids restarting on fail and contains additional tools useful for debugging
- CLOUD-556: Kubernetes 1.17 / Google Kubernetes Engine 1.17 support

30.4.3 Bugs Fixed

- K8SPSMDB-213: Installation instruction not reflecting recent changes in git tags (Thanks to user geraintj for reporting this issue)
- K8SPSMDB-210: Backup documentation not reflecting changes in Percona Backup for MongoDB
- K8SPSMDB-180: Replset and cluster having "ready" status set before mongo initialization and replicasets configuration finished
- K8SPSMDB-179: The "error" cluster status instead of the "initializing" one during the replset initialization
- CLOUD-531: Wrong usage of `strings.TrimLeft` when processing apiVersion

30.5 *Percona Kubernetes Operator for Percona Server for MongoDB* 1.4.0

Date March 31, 2020

Installation Installing Percona Kubernetes Operator for PSMDB

30.5.1 New Features

- K8SPSMDB-89: Amazon Elastic Container Service for Kubernetes (EKS) was added to the list of the officially supported platforms
- K8SPSMDB-113: Percona Server for MongoDB 4.2 is now supported
- OpenShift Container Platform 4.3 is now supported

30.5.2 Improvements

- K8SPSMDB-79: The health check algorithm improvements have increased the overall stability of the Operator
- K8SPSMDB-176: The Operator was updated to use Percona Backup for MongoDB version 1.2
- K8SPSMDB-153: Now the user can adjust securityContext, replacing the automatically generated securityContext with the customized one
- K8SPSMDB-175: Operator now updates observedGeneration status message to allow better monitoring of the cluster rollout or backups/restore process

30.5.3 Bugs Fixed

- K8SPSMDB-182: Setting the `updateStrategy: OnDelete` didn't work if was not specified from scratch in CR
- K8SPSMDB-174: 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.

Help us improve our software quality by reporting any bugs you encounter using our bug tracking system.

30.6 *Percona Kubernetes Operator for Percona Server for MongoDB* 1.3.0

Percona announces the *Percona Kubernetes Operator for Percona Server for MongoDB* 1.3.0 release on December 11, 2019. This release is now the current GA release in the 1.3 series. Install the Kubernetes Operator for Percona Server for MongoDB by following the instructions.

The Operator simplifies the deployment and management of the Percona Server for MongoDB 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.

30.6.1 New Features and Improvements

- CLOUD-415: Non-default cluster domain can now be specified with the new `ClusterServiceDNSSuffix` Operator option.
- CLOUD-395: The Percona Server for MongoDB images size decrease by 42% 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 have been provided.

Percona Server for MongoDB is an enhanced, open source and highly-scalable database that is a fully-compatible, drop-in replacement for MongoDB Community Edition. It supports MongoDB protocols and drivers. Percona Server for MongoDB extends MongoDB Community Edition functionality by including the Percona Memory Engine, as well as several enterprise-grade features. It requires no changes to MongoDB applications or code.

Help us improve our software quality by reporting any bugs you encounter using our bug tracking system.

30.7 *Percona Kubernetes Operator for Percona Server for MongoDB* 1.2.0

Percona announces the *Percona Kubernetes Operator for Percona Server for MongoDB* 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 Server for MongoDB by following the instructions.

The Operator simplifies the deployment and management of the Percona Server for MongoDB 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.

30.7.1 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.
- Data-at-rest encryption is now enabled by default unless `EnableEncryption=false` is explicitly specified in the `deploy/cr.yaml` configuration file.
- 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.
- The resource constraint values were refined for all containers to eliminate the possibility of an out of memory error.

30.7.2 Fixed Bugs

- Oscillations of the cluster status between "initializing" and "ready" took place after an update.
- The Operator was removing other cron jobs in case of the enabled backups without defined tasks (contributed by Marcel Heers).

Percona Server for MongoDB is an enhanced, open source and highly-scalable database that is a fully-compatible, drop-in replacement for MongoDB Community Edition. It supports MongoDB protocols and drivers. Percona Server for MongoDB extends MongoDB Community Edition functionality by including the Percona Memory Engine, as well as several enterprise-grade features. It requires no changes to MongoDB applications or code.

Help us improve our software quality by reporting any bugs you encounter using our bug tracking system.

30.8 *Percona Kubernetes Operator for Percona Server for MongoDB* 1.1.0

Percona announces the general availability of *Percona Kubernetes Operator for Percona Server for MongoDB* 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 Server for MongoDB by following the instructions. Please see the GA release announcement.

The Operator simplifies the deployment and management of the Percona Server for MongoDB 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.

30.8.1 New Features and Improvements

- Now the Percona Kubernetes Operator allows upgrading Percona Server for MongoDB to newer versions, either in semi-automatic or in manual mode.
- Also, two modes are implemented for updating the Percona Server for MongoDB `mongod.conf` configuration file: in *automatic configuration update* mode Percona Server for MongoDB Pods are immediately re-created to populate changed options from the Operator YAML file, while in *manual mode* changes are held until Percona Server for MongoDB Pods are re-created manually.
- Percona Server for MongoDB data-at-rest encryption is now supported by the Operator to ensure that encrypted data files cannot be decrypted by anyone except those with the decryption key.
- 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 the 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 Server for MongoDB is an enhanced, open source and highly-scalable database that is a fully-compatible, drop-in replacement for MongoDB Community Edition. It supports MongoDB protocols and drivers. Percona Server for MongoDB extends MongoDB Community Edition functionality by including the Percona Memory Engine, as well as several enterprise-grade features. It requires no changes to MongoDB applications or code.

Help us improve our software quality by reporting any bugs you encounter using our bug tracking system.

30.9 *Percona Kubernetes Operator for Percona Server for MongoDB* 1.0.0

Percona announces the general availability of *Percona Kubernetes Operator for Percona Server for MongoDB* 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 Server for MongoDB 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 Server for MongoDB automates the lifecycle of your Percona Server for MongoDB environment. The Operator can be used to create a Percona Server for MongoDB replica set, or scale an existing replica set.

The Operator creates a Percona Server for MongoDB replica set with the needed settings and provides a consistent Percona Server for MongoDB instance. The Percona Kubernetes Operators are based on best practices for configuration and setup of the Percona Server for MongoDB.

The Kubernetes Operators provide 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 and may save time while providing a consistent environment.

The advantages are the following:

- Deploy a Percona Server for MongoDB 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 Server for MongoDB size parameter to add or remove Percona Server for MongoDB replica set members
- Integrate with Percona Monitoring and Management (PMM) to seamlessly monitor your Percona Server for MongoDB
- 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 Percona Server for MongoDB replica set member
- 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
- Supports a replica set Arbiter member
- Supports Percona Server for MongoDB versions 3.6 and 4.0

30.9.1 Installation

Installation is performed by following the documentation installation instructions for Kubernetes and OpenShift.