Percona Backup for MongoDB Documentation

Release 1.5.0

May 13, 2021

CONTENTS

Percona Backup for MongoDB is a distributed, low-impact solution for achieving consistent backups of MongoDB sharded clusters and replica sets.

Percona Backup for MongoDB supports Percona Server for MongoDB and MongoDB Community v3.6 or higher with MongoDB Replication enabled.

Note: Percona Backup for MongoDB doesn't work on standalone MongoDB instances. This is because Percona Backup for MongoDB requires an *oplog* to guarantee backup consistency. Oplog is available on nodes with replication enabled.

For testing purposes, you can deploy Percona Backup for MongoDB on a single-node replica set. (Specify the `replication.replSetName` option in the configuration file of the standalone server.)

See also:

MongoDB Documentation: Convert a Standalone to a Replica Set https://docs.mongodb.com/manual/tutorial/convert-standalone-to-replica-set/

The Percona Backup for MongoDB project is inherited from and replaces *mongodb_consistent_backup*, which is no longer actively developed or supported.

Features

- Logical backup and restore. Physical (a.k.a. 'hot') backup and restore are not supported
- Works for both for sharded clusters and classic, non-sharded replica sets.
- Point-in-time Restore
- Simple command-line management utility
- Simple, integrated-with-MongoDB authentication
- Distributed transaction consistency with MongoDB 4.2+
- Use with any S3-compatible storage
- Users with classic, locally-mounted remote filesystem backup servers can use 'filesystem' instead of 's3' storage type.

Part I

Introduction

CHAPTER

ONE

HOW PERCONA BACKUP FOR MONGODB WORKS

Even in a highly-available architecture, such as with MongoDB replication, backups are still required even though losing one server is not fatal. Whether for a complete or partial data disaster, you can use PBM (Percona Backup for MongoDB) to go back in time to the best available backup snapshot.

Percona Backup for MongoDB is a command line interface. It provides the set of commands to make backup and restore operations in your database.

The following example illustrates how to use Percona Backup for MongoDB.

For example, imagine your web application's update was released on February 2nd 23:00 EDT but, by 11:23 the next day, someone realizes that the update has a bug that is wiping the historical data of any user who logged in. Nobody likes to have downtime, but it's time to roll back: what's the best backup to use?

```
$ pbm list
```

Output

```
Backup snapshots:
  2021-02-03T08:08:15Z [complete: 2021-02-03T08:08:35]
  2021-02-05T13:55:55Z [complete: 2021-02-05T13:56:15]
  2021-02-07T13:57:58Z [complete: 2021-02-07T13:58:17]
  2021-02-09T14:06:06Z [complete: 2021-02-09T14:06:26]
  2021-02-11T14:22:41Z [complete: 2021-02-11T14:23:01]
```

The most recent daily backup is 09:22 EDT (14:22 UTC), which would include 4 hours of damage caused by the bug. Let's restore the one before that:

```
$ pbm restore 2021-02-09T14:06:06Z
```

To be on the safe side, it might be best to make an extra backup manually before the next application release:

```
$ pbm backup
```

Find the full set of commands available in Percona Backup for MongoDB in *pbm commands*.

CHAPTER

TWO

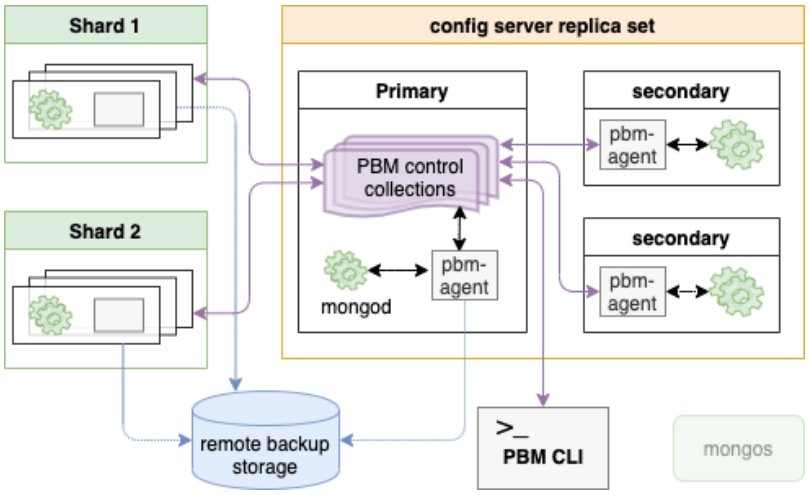
PERCONA BACKUP FOR MONGODB COMPONENTS

Percona Backup for MongoDB consists of the following components:

- *pbm-agent* is a process running on every `mongod` node within the cluster or a replica set that performs backup and restore operations.
- *PBM CLI* is a command-line utility that instructs pbm-agents to perform an operation.

  A single **`pbm-agent`** is only involved with one cluster (or non-sharded replica set). The `pbm` CLI utility can connect to any cluster it has network access to, so it is possible for one user to list and launch backups or restores on many clusters.
- *PBM Control collections* are special collections in MongoDB that store the configuration data and backup states. Both `pbm` CLI and **`pbm-agent`** use PBM Control collections to check backup status in MongoDB and communicate with each other.
- Remote backup storage is where Percona Backup for MongoDB saves backups. It can be either an *S3 compatible storage* or a filesystem-type storage.

The following diagram illustrates how Percona Backup for MongoDB components communicate with MongoDB.



To learn more about Percona Backup for MongoDB architecture, see *Architecture*.

Part II

Getting started

CHAPTER

THREE

INSTALLING PERCONA BACKUP FOR MONGODB

Percona provides and supports installation packages for Percona Backup for MongoDB in the *deb* and *rpm* formats that you can install by using `apt` or `yum` or other interfaces to your package management system.

The Percona Software and Platform Lifecycle page lists Linux distributions for which Percona Backup for MongoDB installation packages are available.

The recommended approach is to install Percona Backup for MongoDB from Percona repositories using the `percona-release` tool.

If you want a full control over the installation, you may build and install Percona Backup for MongoDB from source code.

Regardless of the installation method you choose, the following tools are at your disposal after the installation completes:

Tool	Purpose
pbm	Command-line interface for controlling the backup system
pbm-agent	An agent for running backup/restore actions on a database host

Install **`pbm-agent`** on every server that has `mongod` nodes in the MongoDB cluster (or non-sharded replica set). You don't need to install **`pbm-agent`** on arbiter nodes as they don't have the data set.

You can install `pbm` CLI on any or all servers or desktop computers you wish to use it from, so long as those computers aren't network-blocked from accessing the MongoDB cluster.

3.1 Installing from Percona repositories

To install Percona Backup for MongoDB, first install `percona-release` tool using the package manager of your operating system. This is a repository management tool that automatically configures the required repository for you.

Follow the instructions in Percona Software repositories documentation to install `percona-release`.

Enable the repository. As of version 1.3.0, Percona Backup for MongoDB packages are stored in the *pbm* repository.

```
$ sudo percona-release enable pbm release
```

3.1.1 Installing Percona Backup for MongoDB Using `apt`

```
$ sudo apt-get update
$ sudo apt-get install percona-backup-mongodb
```

3.1.2 Installing Percona Backup for MongoDB Using `yum`

```
$ sudo yum install percona-backup-mongodb
```

3.2 Building from source code

Building the project requires:

- Go 1.11 or above
- make
- git
- `krb5-devel` for Red Hat Enterprise Linux / CentOS or `libkrb5-dev` for Debian / Ubuntu. This package is required for Kerberos authentication in Percona Server for MongoDB.

See also:

Installing and setting up Go tools https://golang.org/doc/install

To build the project (from the project dir):

```
$ git clone https://github.com/<your_name>/percona-backup-mongodb
$ cd percona-backup-mongodb
$ make build
```

After **`make`** completes, you can find `pbm` and **`pbm-agent`** binaries in the `./bin` directory:

```
$ cd bin
$ ./pbm version
```

By running **`pbm version`**, you can verify if Percona Backup for MongoDB has been built correctly and is ready for use.

Output

```
Version:   [pbm version number]
Platform:  linux/amd64
GitCommit: [commit hash]
GitBranch: master
BuildTime: [time when this version was produced in UTC format]
GoVersion: [Go version number]
```

Tip: Instead of specifying the path to pbm binaries, you can add it to the PATH environment variable:

```
export PATH=/percona-backup-mongodb/bin:$PATH
```

After Percona Backup for MongoDB is successfully installed on your system, you have `pbm` and **`pbm-agent`** programs available. See *Initial setup* for guidelines how to set up Percona Backup for MongoDB.

CHAPTER

FOUR

INITIAL SETUP

After you've installed **pbm-agent** on every server with a `mongod` node that is not an arbiter node, perform initial setup to run Percona Backup for MongoDB.

The setup steps are the following:

1. Configure authentication in MongoDB
2. Use `pbm CLI` to insert the config information for remote backup storage
3. Start **pbm-agent** process

4.1 Configure authentication in MongoDB

Percona Backup for MongoDB has no authentication and authorization subsystem of its own - it uses the one of MongoDB. This means that to authenticate Percona Backup for MongoDB, you need to create a corresponding `pbm` user in the `admin` database and set a valid MongoDB connection URI string for both **pbm-agent** and `pbm`.

4.1.1 Create the `pbm` user

First create the role that allows any action on any resource.

```
db.getSiblingDB("admin").createRole({ "role": "pbmAnyAction",
      "privileges": [
         { "resource": { "anyResource": true },
           "actions": [ "anyAction" ]
         }
      ],
```

```
      "roles": []
   });
```

Then create the user and assign the role you created to it.

```
db.getSiblingDB("admin").createUser({user: "pbmuser",
       "pwd": "secretpwd",
       "roles" : [
          { "db" : "admin", "role" : "readWrite", "collection": "" },
          { "db" : "admin", "role" : "backup" },
          { "db" : "admin", "role" : "clusterMonitor" },
          { "db" : "admin", "role" : "restore" },
          { "db" : "admin", "role" : "pbmAnyAction" }
       ]
    });
```

You can specify username and password values and other options of the `createUser` command as you require so long as the roles shown above are granted.

Create the `pbm` user on every replica set. In a sharded cluster, this means every shard replica set and the config server replica set.

Note: In a cluster run *db.getSiblingDB("config").shards.find({}, {"host": true, "_id": false})* to list all the host+port lists for the shard replica sets. The replica set name at the *front* of these "host" strings will have to be placed as a "/?replicaSet=xxxx" argument in the parameters part of the connection URI (see below).

4.1.2 Set the MongoDB connection URI for `pbm-agent`

A **`pbm-agent`** process connects to its localhost `mongod` node with a standalone type of connection. To set the MongoDB URI connection string means to configure a service init script (`pbm-agent.service` systemd unit file) that runs a **`pbm-agent`**.

The `pbm-agent.service` systemd unit file includes the environment file. You set the MongoDB URI connection string for the "PBM_MONGODB_URI" variable within the environment file.

The environment file for Debian and Ubuntu is `/etc/default/pbm-agent`. For Redhat and CentOS, it is `/etc/sysconfig/pbm-agent`.

Edit the environment file and specify MongoDB connection URI string for the `pbm` user to the local `mongod` node. For example, if `mongod` node listens on port 27018, the MongoDB connection URI string will be the following:

```
PBM_MONGODB_URI="mongodb://pbmuser:secretpwd@localhost:27018"
```

Configure the service init script for every **`pbm-agent`**.

Hint: How to find the environment file

The path to the environment file is specified in the `pbm-agent.service` systemd unit file.

In Ubuntu and Debian, the `pbm-agent.service` systemd unit file is at the path `/lib/systemd/system/pbm-agent.service`.

In Red Hat and CentOS, the path to this file is `/usr/lib/systemd/system/pbm-agent.service`.

Example of pbm-agent.service systemd unit file

```
[Unit]
Description=pbm-agent
After=time-sync.target network.target

[Service]
EnvironmentFile=-/etc/default/pbm-agent
Type=simple
User=pbm
Group=pbm
PermissionsStartOnly=true
ExecStart=/usr/bin/pbm-agent

[Install]
WantedBy=multi-user.target
```

See also:

More information about standard MongoDB connection strings *Authentication*

4.1.3 Set the MongoDB connection URI for `pbm CLI`

Provide the MongoDB URI connection string for `pbm` in your shell. This allows you to call `pbm` commands without the `--mongodb-uri` flag.

Use the following command:

```
export PBM_MONGODB_URI="mongodb://pbmuser:secretpwd@localhost:27018?/replSetName=xxxx"
```

For more information what connection string to specify, refer to *The pbm connection string* section.

4.2 Configure remote backup storage

The easiest way to provide remote backup storage configuration is to specify it in a YAML config file and upload this file to Percona Backup for MongoDB using `pbm CLI`.

The storage configuration itself is out of scope of the present document. We assume that you have configured one of the *supported remote backup storages*.

1. Create a config file (e.g. `pbm_config.yaml`).
2. Specify the storage information within.

   The following is the sample configuration for Amazon AWS:

```
storage:
  type: s3
  s3:
    region: us-west-2
    bucket: pbm-test-bucket
    prefix: data/pbm/backup
    credentials:
      access-key-id: <your-access-key-id-here>
      secret-access-key: <your-secret-key-here>
    serverSideEncryption:
```

```
      sseAlgorithm: aws:kms
      kmsKeyID: <your-kms-key-here>
```

This is the sample configuration for Microsoft Azure Blob storage:

```
storage:
  type: azure
  azure:
    account: <your-account>
    container: <your-container>
    prefix: pbm
    credentials:
      key: <your-access-key>
```

This is the sample configuration for filesystem storage:

```
storage:
  type: filesystem
  filesystem:
    path: /data/local_backups
```

Important: When using a filesystem storage, verify that the user running Percona Backup for MongoDB is the owner of the backup folder.

```
$ sudo chown pbm:pbm <backup_directory>
```

See more examples in *Example config files*.

3. Insert the config file

```
$ pbm config --file pbm_config.yaml
```

For a sharded cluster, run this command whilst connecting to config server replica set. Otherwise connect to the non-sharded replica set as normal.

To learn more about Percona Backup for MongoDB configuration, see *Percona Backup for MongoDB configuration in a cluster (or non-sharded replica set)*.

4.3 Start the `pbm-agent` process

Start **`pbm-agent`** on every server with the `mongod` node installed. It is best to use the packaged service scripts to run **`pbm-agent`**.

```
$ sudo systemctl start pbm-agent
$ sudo systemctl status pbm-agent
```

E.g. Imagine you put configsvr nodes (listen port 27019) collocated on the same servers as the first shard's `mongod` nodes (listen port 27018, replica set name "sh1rs"). In this server there should be two **`pbm-agent`** processes, one connected to the shard (e.g. "mongodb://username:password@localhost:27018/") and one to the configsvr node (e.g. "mongodb://username:password@localhost:27019/").

For reference, the following is an example of starting **`pbm-agent`** manually. The output is redirected to a file and the process is backgrounded. Alternatively you can run it on a shell terminal temporarily if you want to observe and/or debug the startup from the log messages.

```
$ nohup pbm-agent --mongodb-uri "mongodb://username:password@localhost:27018/" > /
↪data/mdb_node_xyz/pbm-agent.$(hostname -s).27018.log 2>&1 &
```

Tip: Running as the `mongod` user would be the most intuitive and convenient way. But if you want it can be another user.

4.3.1 How to see the pbm-agent log

With the packaged systemd service, the log output to stdout is captured by systemd's default redirection to systemd-journald. You can view it with the command below. See **`man journalctl`** for useful options such as '–lines', '–follow', etc.

```
~$ journalctl -u pbm-agent.service
-- Logs begin at Tue 2019-10-22 09:31:34 JST. --
Jan 22 15:59:14 akira-x1 systemd[1]: Started pbm-agent.
Jan 22 15:59:14 akira-x1 pbm-agent[3579]: pbm agent is listening for the commands
...
...
```

If you started **`pbm-agent`** manually, see the file you redirected stdout and stderr to.

When a message *"pbm agent is listening for the commands"* is printed to the **`pbm-agent`** log file, it confirms that it has connected to its `mongod` node successfully.

Part III

Usage

CHAPTER

FIVE

RUNNING PERCONA BACKUP FOR MONGODB

This document provides examples of using `pbm` commands to operate your backup system. For detailed description of pbm commands, refer to *pbm commands*.

5.1 Listing backups

To view all completed backups, run the **`pbm list`** command.

```
$ pbm list
```

As of version 1.4.0, the **`pbm list`** output shows the completion time.

Sample output

```
Backup snapshots:
  2021-01-13T15:50:54Z [complete: 2021-01-13T15:53:40]
  2021-01-13T16:10:20Z [complete: 2021-01-13T16:13:00]
  2021-01-20T17:09:46Z [complete: 2021-01-20T17:10:33]
```

5.2 Starting a backup

```
$ pbm backup
```

By default, Percona Backup for MongoDB uses `s2` compression method when making a backup. You can start a backup with a different compression method by passing the `--compression` flag to the **`pbm backup`** command.

For example, to start a backup with gzip compression, use the following command

```
$ pbm backup --compression=gzip
```

Supported compression types are: `gzip`, `snappy`, `lz4`, `pgzip`. The `none` value means no compression is done during backup.

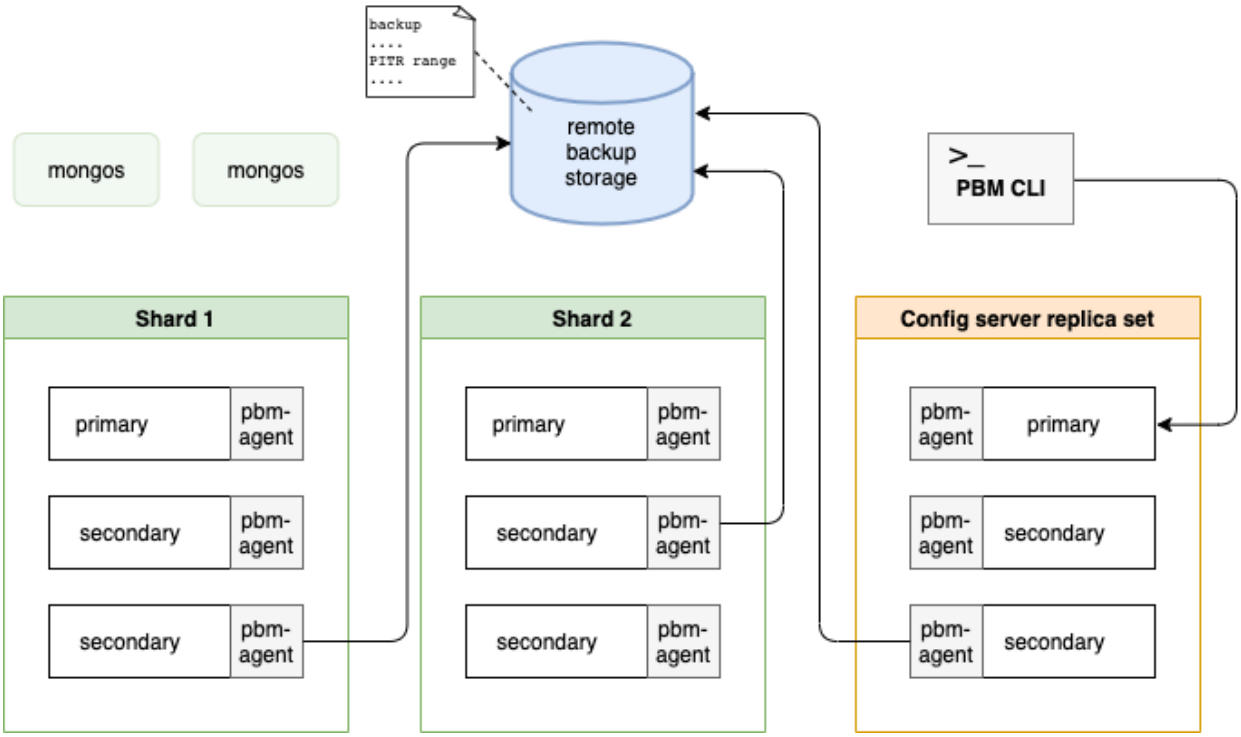
Backup in sharded clusters

Important: For PBM v1.0 (only): before running **`pbm backup`** on a cluster, stop the balancer.

In sharded clusters, one of **`pbm-agent`** processes for every shard and the config server replica set writes backup snapshots and *oplog slices* into the remote backup storage directly. To learn more about oplog slicing, see *Point-in-Time Recovery*.

The `mongos` nodes are not involved in the backup process.

The following diagram illustrates the backup flow.



Adjust node priority for backups

In Percona Backup for MongoDB prior to version 1.5.0, the `pbm-agent` to do a backup is elected randomly among secondary nodes in a replica set. If no secondary node responds in a defined period, then the `pbm-agent` on the primary node is elected to do a backup.

As of version 1.5.0, you can influence the `pbm-agent` election by assigning a priority to `mongod` nodes in the Percona Backup for MongoDB *configuration file*.

```
backup:
  priority:
    "localhost:28019": 2.5
    "localhost:27018": 2.5
    "localhost:27020": 2.0
    "localhost:27017": 0.1
```

The format of the *priority* array is `<hostname:port>:<priority>`.

Note that if you listed only specific nodes, the remaining nodes will be automatically assigned priority 1.0. For example, you assigned priority 2.5 to only one secondary node. Then both the remaining secondaries and the primary node receive priority 1.0.

The `mongod` node with the highest priority makes the backup. If this node is unavailable, next priority node is selected. If there are several nodes with the same priority, one of them is randomly elected to make the backup.

If you haven't listed any nodes for the `priority` option in the config, the nodes have the default priority for making backups as follows:

- hidden nodes - priority 2.0
- secondary nodes - priority 1.0
- primary node - priority 0.5

This ability to adjust node priority helps you manage your backup strategy by selecting specific nodes or nodes from preferred data centers. In geographically distributed infrastructures, you can reduce network latency by making backups from nodes in geographically closest locations.

As soon as you adjust node priorities in the configuration file, it is assumed that you take manual control over them. The default rule to prefer secondary nodes over primary stops working.

5.3 Checking an in-progress backup

Important: As of version 1.4.0, the information about running backups is not available in the **`pbm list`** output. Use the **`pbm status`** command instead to check for running backups. See *Checking status of Percona Backup for MongoDB* for more information.

For Percona Backup for MongoDB version 1.3.4 and earlier, run the **`pbm list`** command and you will see the running backup listed with a 'In progress' label. When that is absent, the backup is complete.

5.4 Restoring a backup

Warning: Backups made with Percona Backup for MongoDB prior to v1.5.0 are incompatible for restore with Percona Backup for MongoDB v1.5.0 and later. This is because processing of system collections `Users` and `Roles` has changed: in v1.5.0, `Users` and `Roles` are copied to temporary collection during backup and must be present in the backup during restore. In earlier versions of Percona Backup for MongoDB, `Users` and `Roles` are copied to a temporary collection during restore. Therefore, restoring from these backups with Percona Backup for MongoDB v1.5.0 isn't possible.

The recommended approach is to make a fresh backup after *upgrading Percona Backup for MongoDB* to version 1.5.0.

To restore a backup that you have made using **`pbm backup`**, use the **`pbm restore`** command supplying the time stamp of the backup that you intend to restore.

Important: Consider these important notes on restore operation:

1. Percona Backup for MongoDB is designed to be a full-database restore tool. As of version <=1.x, it performs a full all-databases, all collections restore and does not offer an option to restore only a subset of collections in the backup, as MongoDB's `mongodump` tool does. But to avoid surprising `mongodump` users, as of versions 1.x, Percona Backup for MongoDB replicates mongodump's behavior to only drop collections in the backup. It does not drop collections that are created new after the time of the backup and before the restore. Run a `db.dropDatabase()` manually in all non-system databases (these are all databases except "local", "config" and "admin") before running **`pbm restore`** if you want to guarantee that the post-restore database only includes collections that are in the backup.
2. Whilst the restore is running, prevent clients from accessing the database. The data will naturally be incomplete whilst the restore is in progress, and writes the clients make cause the final restored data to differ from the backed-up data.
3. If you enabled *Point-in-Time Recovery*, disable it before running **`pbm restore`**. This is because Point-in-Time Recovery incremental backups and restore are incompatible operations and cannot be run together.

```
$ pbm restore 2019-06-09T07:03:50Z
```

New in version 1.3.2: The Percona Backup for MongoDB config includes the restore options to adjust the memory consumption by the **`pbm-agent`** in environments with tight memory bounds. This allows preventing out of memory errors during the restore operation.

```
restore:
  batchSize: 500
  numInsertionWorkers: 10
```

The default values were adjusted to fit the setups with the memory allocation of 1GB and less for the agent.

Note: The lower the values, the less memory is allocated for the restore. However, the performance decreases too.

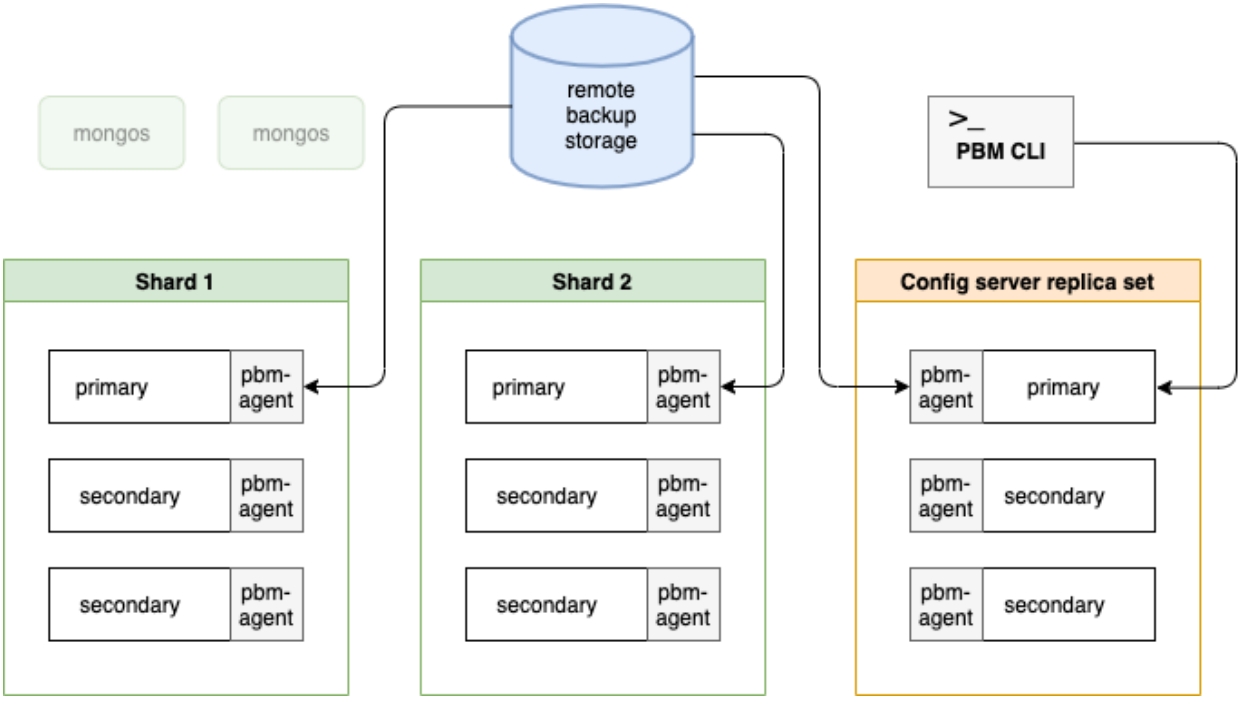
Restoring a backup in sharded clusters

Important: As preconditions for restoring a backup in a sharded cluster, complete the following steps:

1. Stop the balancer.
2. Shut down all `mongos` nodes to stop clients from accessing the database while restore is in progress. This ensures that the final restored data doesn't differ from the backed-up data.
3. Disable point-in-time recovery if it is enabled. To learn more about point-in-time recovery, see *Point-in-Time Recovery*.

Note that you can restore a sharded backup only into a sharded environment. It can be your existing cluster or a new one. To learn how to restore a backup into a new environment, see *Restoring a backup into a new environment*.

During the restore, the **`pbm-agent`** processes write data to primary nodes in the cluster. The following diagram shows the restore flow.



After a cluster's restore is complete, restart all `mongos` nodes to reload the sharding metadata.

Restoring a backup into a new environment

To restore a backup from one environment to another, consider the following key points about the destination environment:

- Replica set names (both the config servers and the shards) in your new destination cluster and in the cluster that was backed up must be exactly the same.
- Percona Backup for MongoDB configuration in the new environment must point to the same remote storage that is defined for the original environment, including the authentication credentials if it is an object store. Once you run **`pbm list`** and see the backups made from the original environment, then you can run the **`pbm restore`** command.

Of course, make sure not to run **pbm backup** from the new environment whilst the Percona Backup for MongoDB config is pointing to the remote storage location of the original environment.

5.5 Canceling a backup

You can cancel a running backup if, for example, you want to do another maintenance of a server and don't want to wait for the large backup to finish first.

To cancel the backup, use the **pbm cancel-backup** command.

```
$ pbm cancel-backup
Backup cancellation has started
```

After the command execution, the backup is marked as canceled in the **pbm list** output:

```
$ pbm list
...
2020-04-30T18:05:26Z  Canceled at 2020-04-30T18:05:37Z
```

5.6 Deleting backups

Use the **pbm delete-backup** command to delete a specified backup or all backups older than the specified time.

The command deletes the backup regardless of the remote storage used: either S3-compatible or a filesystem-type remote storage.

Note: You can only delete a backup that is not running (has the "done" or the "error" state).

As of version 1.4.0, **pbm list** shows only successfully completed backups. To check for backups with other states, run **pbm status**.

To delete a backup, specify the `<backup_name>` as an argument.

```
$ pbm delete-backup 2020-12-20T13:45:59Z
```

By default, the **pbm delete-backup** command asks for your confirmation to proceed with the deletion. To bypass it, add the `-f` or `--force` flag.

```
$ pbm delete-backup --force 2020-04-20T13:45:59Z
```

To delete backups that were created before the specified time, pass the `--older-than` flag to the **pbm delete-backup** command. Specify the timestamp as an argument for **pbm delete-backup** in the following format:

- `%Y-%M-%DT%H:%M:%S` (for example, 2020-04-20T13:13:20) or
- `%Y-%M-%D` (2020-04-20).

```
$ #View backups
$ pbm list
Backup snapshots:
  2020-04-20T20:55:42Z
  2020-04-20T23:47:34Z
```

```
  2020-04-20T23:53:20Z
  2020-04-21T02:16:33Z
$ #Delete backups created before the specified timestamp
$ pbm delete-backup -f --older-than 2020-04-21
Backup snapshots:
  2020-04-21T02:16:33Z
```

5.7 Viewing backup logs

As of version 1.4.0, you can see the logs from all `pbm-agents` in your MongoDB environment using `pbm CLI`. This reduces time for finding required information when troubleshooting issues.

To view **`pbm-agent`** logs, run the **`pbm logs`** command and pass one or several flags to narrow down the search.

The following flags are available:

- `-t`, `--tail` - Show the last N rows of the log
- `-e`, `--event` - Filter logs by all backups or a specific backup
- `-n`, `--node` - Filter logs by a specific node or a replica set
- `-s`, `--severity` - Filter logs by severity level. The following values are supported (from low to high):
- D - Debug
- I - Info
- W - Warning
- E - Error
- F - Fatal
- `-o`, `--output` - Show log information as text (default) or in JSON format.
- `-i`, `--opid` - Filter logs by the operation ID

Examples

The following are some examples of filtering logs:

Show logs for all backups

```
$ pbm logs --event=backup
```

Show the last 100 lines of the log about a specific backup 2020-10-15T17:42:54Z

```
$ pbm logs --tail=100 --event=backup/2020-10-15T17:42:54Z
```

Include only errors from the specific replica set

```
$ pbm logs -n rs1 -s E
```

The output includes log messages of the specified severity type and all higher levels. Thus, when ERROR is specified, both ERROR and FATAL messages are shown in the output.

Implementation details

`pbm-agents` write log information into the `pbmLog` collection in the *PBM Control collections*. Every **`pbm-agent`** also writes log information to stderr so that you can retrieve it when there is no healthy mongod node in your cluster or replica set. For how to view an individual **`pbm-agent`** log, see *How to see the pbm-agent log*.

Note that log information from `pbmLog` collection is shown in the UTC timezone and from the stderr - in the server's time zone.

CHAPTER

SIX

POINT-IN-TIME RECOVERY

Point-in-Time Recovery is restoring a database up to a specific moment. Point-in-Time Recovery includes restoring the data from a backup snapshot and replaying all events that occurred to this data up to a specified moment from *oplog slices*. Point-in-Time Recovery helps you prevent data loss during a disaster such as crashed database, accidental data deletion or drop of tables, unwanted update of multiple fields instead of a single one.

Point-in-Time Recovery is available in Percona Backup for MongoDB starting from v1.3.0. Point-in-Time Recovery is enabled via the `pitr.enabled` config option.

```
$ pbm config --set pitr.enabled=true
```

6.1 Incremental backups

When Point-in-Time Recovery is enabled, **`pbm-agent`** periodically saves consecutive slices of the *oplog*. A method similar to the way replica set nodes elect a new primary is used to select the **`pbm-agent`** that saves the oplog slices. (Find more information in *pbm-agent*.)

A slice covers a 10 minute span of oplog events. It can be shorter if Point-in-Time Recovery is disabled or interrupted by the start of a backup snapshot operation.

The oplog slices are stored in the `pbmPitr` subdirectory in the *remote storage defined in the config*. A slice name reflects the start and end time this slice covers.

The **`pbm list`** output includes the following information:

- Backup snapshots. As of version 1.4.0, it also shows the completion time
- Valid time ranges for recovery
- Point-in-Time Recovery status.

```
$ pbm list

Backup snapshots:
     2020-12-10T12:19:10Z [complete: 2020-12-10T12:23:50]
     2020-12-14T10:44:44Z [complete: 2020-12-14T10:49:03]
     2020-12-14T14:26:20Z [complete: 2020-12-14T14:34:39]
     2020-12-17T16:46:59Z [complete: 2020-12-17T16:51:07]
PITR <on>:
     2020-12-14T14:26:40 - 2020-12-16T17:27:26
     2020-12-17T16:47:20 - 2020-12-17T16:57:55
```

Note: If you just enabled Point-in-Time Recovery, the time range list in the `pbm list` output is empty. It requires 10 minutes for the first chunk to appear in the list.

6.2 Restore to the point in time

A restore and Point-in-Time Recovery incremental backups are incompatible operations and cannot be run simultaneously. You must disable Point-in-Time Recovery before restoring a database:

```
$ pbm config --set pitr.enabled=false
```

Run **`pbm restore`** and specify the timestamp from the valid range:

```
$ pbm restore --time="2020-07-14T14:27:04"
```

Restoring to the point in time requires both a backup snapshot and oplog slices that can be replayed on top of it (the time ranges in PITR section of `pbm list` output must be later then the selected backup snapshot).

See also:

Restoring a backup.

A restore operation changes the time line of oplog events. Therefore, all oplog slices made after the restore time stamp and before the last backup become invalid. After the restore is complete, make a new backup to serve as the starting point for oplog updates:

```
$ pbm backup
```

Re-enable Point-in-Time Recovery to resume saving oplog slices:

```
$ pbm config --set pitr.enabled=true
```

Delete a backup

When you *delete a backup*, all oplog slices that relate to this backup will be deleted too. For example, you delete a backup snapshot 2020-07-24T18:13:09 while there is another snapshot 2020-08-05T04:27:55 created after it. **`pbm-agent`** deletes only oplog slices that relate to 2020-07-24T18:13:09.

The same applies if you delete backups older than the specified time.

Note: When Point-in-Time Recovery is enabled, the most recent backup snapshot and oplog slices that relate to it won't be deleted.

CHAPTER

SEVEN

CHECKING STATUS OF PERCONA BACKUP FOR MONGODB

As of version 1.4.0, you can check the status of Percona Backup for MongoDB running in your MongoDB environment using the **`pbm status`** command.

```
$ pbm status
```

The output provides the information about:

- Your MongoDB deployment and `pbm-agents` running in it: to what `mongod` node each agent is connected, the Percona Backup for MongoDB version it runs and the agent's state
- The currently running backups / restores, if any
- Backups stored in the remote backup storage: backup name, completion time, size and status (complete, canceled, failed)
- *Point-in-Time Recovery* status (enabled or disabled).
- Valid time ranges for point-in-time recovery and the data size

This simplifies troubleshooting since the whole information is provided in one place.

Sample output

```
$ pbm status

Cluster:
========
config:
  - config/localhost:27027: pbm-agent v1.3.2 OK
  - config/localhost:27028: pbm-agent v1.3.2 OK
  - config/localhost:27029: pbm-agent v1.3.2 OK
rs1:
  - rs1/localhost:27018: pbm-agent v1.3.2 OK
  - rs1/localhost:27019: pbm-agent v1.3.2 OK
  - rs1/localhost:27020: pbm-agent v1.3.2 OK
rs2:
  - rs2/localhost:28018: pbm-agent v1.3.2 OK
  - rs2/localhost:28019: pbm-agent v1.3.2 OK
  - rs2/localhost:28020: pbm-agent v1.3.2 OK

PITR incremental backup:
========================
Status [OFF]

Currently running:
==================
```

```
(none)

Backups:
========
S3 us-east-1 https://storage.googleapis.com/backup-test
   Snapshots:
     2020-12-16T10:36:52Z 491.98KB [complete: 2020-12-16T10:37:13]
     2020-12-15T12:59:47Z 284.06KB [complete: 2020-12-15T13:00:08]
     2020-12-15T11:40:46Z 0.00B [canceled: 2020-12-15T11:41:07]
     2020-12-11T16:23:55Z 284.82KB [complete: 2020-12-11T16:24:16]
     2020-12-11T16:22:35Z 284.04KB [complete: 2020-12-11T16:22:56]
     2020-12-11T16:21:15Z 283.36KB [complete: 2020-12-11T16:21:36]
     2020-12-11T16:19:54Z 281.73KB [complete: 2020-12-11T16:20:15]
     2020-12-11T16:19:00Z 281.73KB [complete: 2020-12-11T16:19:21]
     2020-12-11T15:30:38Z 287.07KB [complete: 2020-12-11T15:30:59]
PITR chunks:
     2020-12-16T10:37:13 - 2020-12-16T10:43:26 44.17KB
```

Part IV

Details

CHAPTER

EIGHT

ARCHITECTURE

8.1 pbm-agent

pbm-agent is a process that runs backup, restore, delete and other operations available with Percona Backup for MongoDB.

A **pbm-agent** instance must run on each `mongod` instance. This includes replica set nodes that are currently secondaries and config server replica set nodes in a sharded cluster.

An operation is triggered when the `pbm` CLI makes an update to the PBM Control collection. All `pbm-agents` monitor changes to the PBM control collections but only one **pbm-agent** in each replica set will be elected to execute an operation. The elections are done by a random choice among secondary nodes. If no secondary nodes respond, then the **pbm-agent** on the primary node is elected for an operation.

The elected **pbm-agent** acquires a lock for an operation. This prevents mutually exclusive operations like backup and restore to be executed simultaneously.

When the operation is complete, the **pbm-agent** releases the lock and updates the PBM control collections.

8.2 PBM Command Line Utility (pbm)

`pbm` CLI is the command line tool with which you operate Percona Backup for MongoDB. `pbm` provides the **pbm** command that you will use manually in the shell. It will also work as a command that can be executed in scripts (for example, by `crond`).

The set of *pbm sub-commands* enables you to manage backups in your MongoDB environment.

`pbm` uses *PBM Control collections* to communicate with **pbm-agent** processes. It starts and monitors backup or restore operations by updating and reading the corresponding PBM control collections for operations, log, etc. Likewise, it modifies the PBM config by saving it in the PBM Control collection for config values.

pbm does not have its own config and/or cache files. Setting the PBM_MONGODB_URI environment variable in your shell is a configuration-like step that should be done for practical ease though. (Without PBM_MONGODB_URI the --mongodb-uri command line argument will need to be specified each time.)

To learn how to set the PBM_MONGODB_URI environment variable, see *Set the MongoDB connection URI for pbm CLI*. For more information about MongoDB URI connection strings, see *Authentication*.

8.3 PBM Control Collections

The config and state (current and historical) for backups is stored in collections in the MongoDB cluster or non-sharded replica set itself. These are put in the system admin db to keep them cleanly separated from user db namespaces.

In sharded clusters, this is the admin db of the config server replica set. In a non-sharded replica set, the PBM Control Collections are stored in admin db of the replica set itself.

- *admin.pbmBackups* - Log / status of each backup
- *admin.pbmAgents* - Contains information about pbm-agents statuses and health
- *admin.pbmConfig* - Contains configuration information for Percona Backup for MongoDB
- *admin.pbmCmd* - Is used to define and trigger operations
- *admin.pbmLock* - **pbm-agent** synchronization-lock structure
- *admin.pbmLockOp* - Is used to coordinate operations that are not mutually-exclusive such as make backup and delete backup.
- *admin.pbmLog* - Stores log information from all pbm-agents in the MongoDB environment. Available in Percona Backup for MongoDB as of version 1.4.0
- *admin.pbmOpLog* - Stores *operation IDs*
- *admin.pbmPITRChunks* - Stores *Point-in-Time Recovery* oplog slices
- *admin.pbmPITRState* - Contains current state of Point-in-Time Recovery incremental backups
- *admin.pbmRestores* - Contains restore history and the restore state for all replica sets
- *admin.pbmStatus* - Stores Percona Backup for MongoDB status records

The pbm command line tool creates these collections as needed. You do not have to maintain these collections, but you should not drop them unnecessarily either. Dropping them during a backup will cause an abort of the backup.

Filling the config collection is a prerequisite to using Percona Backup for MongoDB for executing backups or restores. (See config page later.)

8.4 Remote Backup Storage

Percona Backup for MongoDB saves your files to a directory. Conceptually in the case of object store; actually if you are using filesystem-type remote storage. Using **pbm list**, a user can scan this directory to find existing backups even if they never used pbm on their computer before.

The files are prefixed with the (UTC) starting time of the backup. For each backup there is one metadata file. For each replica set, a backup includes the following:

- A mongodump-format compressed archive that is the dump of collections
- A (compressed) BSON file dump of the oplog covering the timespan of the backup.

The end time of the oplog slice(s) is the data-consistent point in time of a backup snapshot.

For details about supported backup storages, see *Remote backup storage*.

CHAPTER

NINE

AUTHENTICATION

Percona Backup for MongoDB has no authentication and authorization subsystem of its own - it uses MongoDB's, i.e. `pbm` and **`pbm-agent`** only require a valid MongoDB connection URI string for the PBM user.

For the S3-compatible remote storage authentication config, see *Percona Backup for MongoDB configuration in a cluster (or non-sharded replica set)*.

9.1 MongoDB connection strings - A Reminder (or Primer)

Percona Backup for MongoDB uses MongoDB Connection URI strings to open MongoDB connections. Neither `pbm` or **`pbm-agent`** accept legacy-style command-line arguments for `--host`, `--port`, `--user`, `--password`, etc. as the `mongo` shell or `mongodump` command does.

```
$ pbm-agent --mongodb-uri "mongodb://pbmuser:secretpwd@localhost:27018/"
$ #Alternatively:
$ export PBM_MONGODB_URI="mongodb://pbmuser:secretpwd@localhost:27018/"
$ pbm-agent
```

```
$ pbm list --mongodb-uri "mongodb://pbmuser:secretpwd@mongocsvr1:27018,
↪mongocsvr2:27018,mongocsvr3:27018/?replicaSet=configrs"
$ #Alternatively:
$ export PBM_MONGODB_URI="mongodb://pbmuser:secretpwd@mongocsvr1:27018,
↪mongocsvr2:27018,mongocsvr3:27018/?replicaSet=configrs"
$ pbm list
```

The connection URI above is the format that MongoDB drivers accept universally since approximately the release time of MongoDB server v3.6. The `mongo` shell accepts it too since v4.0. Using a v4.0+ mongo shell is a recommended way to debug connection URI validity from the command line.

The MongoDB Connection URI specification includes several non-default options you may need to use. For example the TLS certificates/keys needed to connect to a cluster or non-sharded replicaset with network encryption enabled are "tls=true" plus "tlsCAFile" and/or "tlsCertificateKeyFile" (see tls options).

Technical note

As of v1.0 the driver used by Percona Backup for MongoDB is the official v1.1 mongo-go-driver.

9.1.1 The `pbm-agent` connection string

`pbm-agent` processes should connect to their localhost mongod with a standalone type of connection.

9.1.2 The `pbm` connection string

The `pbm` CLI will ultimately connect to the replica set with the *PBM control collections*.

- In a non-sharded replica set it is simply that replica set.
- In a cluster it is the config server replica set.

You do not necessarily have to provide that connection string. If you provide a connection to any live node (shard, configsvr, or non-sharded replicaset member), it will automatically determine the right hosts and establish a new connection to those instead.

Tip: When running `pbm` from an unsupervised script, we recommend using a replica set connection string. A standalone-style connection string will fail if that `mongod` host happens to be down temporarily.

CHAPTER

TEN

PERCONA BACKUP FOR MONGODB CONFIGURATION IN A CLUSTER (OR NON-SHARDED REPLICA SET)

The configuration information is stored in a single document of the *admin.pbmConfig* collection. That single copy is shared by all the **pbm-agent** processes in a cluster (or non-sharded replica set), and can be read or updated using the `pbm` tool.

You can see the whole config by running *db.getSiblingDB("admin").pbmConfig.findOne()*. But you don't have to use the mongo shell; the `pbm` CLI has a "config" subcommand to read and update it.

Percona Backup for MongoDB config contains the following settings:

- *Remote backup storage* configuration is available as of v1.0 or v1.1
- *Point-in-Time Recovery* configuration is available as of v1.3.0
- *Restore options* are available as of v1.3.2

Run **pbm config --list** to see the whole config. (Sensitive fields such as keys will be redacted.)

10.1 Insert the whole Percona Backup for MongoDB config from a YAML file

If you are initializing a cluster or non-sharded replica set for the first time, it is simplest to write the whole config as YAML file and use the **pbm config --file** method to upload all the values in one command.

The *Example config files* section provides config file examples for the remote backup storage (required). For more information about available config file options, see *Configuration file options*.

Use the following command to upload the config file (e.g. `pbm_config.yaml`):

```
$ pbm config --file pbm_config.yaml
```

Execute whilst connecting to config server replica set if it is a cluster. Otherwise just connect to the non-sharded replica set as normal. (See *MongoDB connection strings - A Reminder (or Primer)* if you are not familiar with MongoDB connection strings yet.)

10.2 Accessing or updating single config values

You can set a single value at time. For nested values use dot-concatenated key names as shown in the following example:

```
$ pbm config --set storage.s3.bucket="operator-testing"
```

To list a single value you can specify just the key name by itself and the value will be returned (if set)

```
$ pbm config storage.s3.bucket
operator-testing
$ pbm config storage.s3.INVALID-KEY
Error: unable to get config key: invalid config key
```

CHAPTER

ELEVEN

REMOTE BACKUP STORAGE

Percona Backup for MongoDB supports the following types of remote backup storages:

- S3-compatible storage
- Filesystem type storage
- Microsoft Azure Blob storage

S3 compatible storage

Percona Backup for MongoDB should work with other S3-compatible storages but was only tested with the following ones:

- Amazon Simple Storage Service
- Google Cloud Storage
- MinIO

As of v1.3.2, Percona Backup for MongoDB supports *server-side encryption* for *S3 buckets* with customer managed keys stored in AWS KMS (AWS Key Management Service).

See also:

Protecting Data Using Server-Side Encryption with CMKs Stored in AWS Key Management Service (SSE-KMS)

Remote Filesystem Server Storage

This storage must be a remote fileserver mounted to a local directory. It is the responsibility of the server administrators to guarantee that the same remote directory is mounted at exactly the same local path on all servers in the MongoDB cluster or non-sharded replica set.

Warning: Percona Backup for MongoDB uses the directory as if it were any normal directory, and does not attempt to confirm it is mounted from a remote server. If the path is accidentally a normal local directory, errors will eventually occur, most likely during a restore attempt. This will happen because **`pbm-agent`** processes of other nodes in the same replica set can't access backup archive files in a normal local directory on another server.

Local Filesystem Storage

This cannot be used except if you have a single-node replica set. (See the warning note above as to why). We recommend using any object store you might be already familiar with for testing. If you don't have an object store yet, we recommend using MinIO for testing as it has simple setup. If you plan to use a remote filesytem-type backup server, please see the *Remote Filesystem Server Storage* above.

Microsoft Azure Blob Storage

As of v1.5.0, you can use Microsoft Azure Blob Storage as the remote backup storage for Percona Backup for MongoDB.

This gives users a vendor choice. Companies with Microsoft-based infrastructure can set up Percona Backup for MongoDB with less administrative efforts.

11.1 Example config files

Provide the remote backup storage configuration as a YAML config file. The following are the examples of config fles for supported remote storages. For how to insert the config file, see *Insert the whole Percona Backup for MongoDB config from a YAML file*.

S3-compatible remote storage

Amazon Simple Storage Service

```
storage:
  type: s3
  s3:
    region: us-west-2
    bucket: pbm-test-bucket
    prefix: data/pbm/backup
    credentials:
      access-key-id: <your-access-key-id-here>
      secret-access-key: <your-secret-key-here>
    serverSideEncryption:
      sseAlgorithm: aws:kms
      kmsKeyID: <your-kms-key-here>
```

GCS

```
storage:
 type: s3
 s3:
     region: us-east1
     bucket: pbm-testing
     prefix: pbm/test
     endpointUrl: https://storage.googleapis.com
     credentials:
       access-key-id: <your-access-key-id-here>
       secret-access-key: <your-secret-key-here>
```

MinIO

```
storage:
  type: s3
  s3:
    endpointUrl: "http://localhost:9000"
    region: my-region
    bucket: pbm-example
    prefix: data/pbm/test
    credentials:
      access-key-id: <your-access-key-id-here>
      secret-access-key: <your-secret-key-here>
```

Remote filesystem server storage

```
storage:
  type: filesystem
  filesystem:
    path: /data/local_backups
```

Microsoft Azure Blob Storage

```
storage:
  type: azure
  azure:
    account: <your-account>
    container: <your-container>
    prefix: pbm
    credentials:
      key: <your-access-key>
```

For the description of configuration options, see *Configuration file options*.

Part V

How to

CHAPTER

TWELVE

SCHEDULE BACKUPS

In Percona Backup for MongoDB version 1.4.1 and earlier, we recommend using `crond` or similar services to schedule backup snapshots.

Important: Before configuring `crond`, make sure that you have *installed* and *configured* Percona Backup for MongoDB to make backups in your database. Start a backup manually to verify this: **`pbm backup`**.

The recommended approach is to create a `crontab` file in the `/ect/cron.d` directory and specify the command in it. This simplifies server administration especially if multiple users have access to it.

`pbm` CLI requires a valid MongoDB URI connection string to authenticate in MongoDB. Instead of specifying the MongoDB URI connection string as a command line argument, which is a potential security risk, we recommend creating an environment file and specify the `export PBM_MONGODB_URI=$PBM_MONGODB_URI` statement within.

As an example, let's configure to run backup snapshots on 23:30 every Sunday. The steps are the following:

1. Create an environment file. Let's name it `pbm-cron`. The path for this file on Debian and Ubuntu is `/etc/default/pbm-cron`. For Red Hat Enterprise Linux and CentOS, the path is `/etc/sysconfig/pbm-cron`.
2. Specify the environment variable in `pbm-cron`:

```
export PBM_MONGODB_URI="mongodb://pbmuser:secretpwd@localhost:27018?/
↪replSetName=xxxx"
```

3. Grant access to the `pbm-cron` file for the user that will execute the `cron` task.
4. Create a `crontab` file. Let's name it `pbm-backup`.
5. Specify the command in the file:

```
30 23 * * sun <user-to-execute-cron-task> . /etc/default/pbm-cron; /usr/bin/pbm␣
↪backup
```

   Note the dot `.` before the environment file. It sources (includes) the environment file for the rest of the shell commands.

1. Verify that backups are running in `/var/log/cron` or `/var/log/syslog` logs:

```
$ grep CRON /var/log/syslog
```

Backup storage cleanup

Previous backups are not automatically removed from the backup storage. You need to remove the oldest ones periodically to limit the amount of space used in the backup storage.

We recommend using the **pbm delete backup --older-than <timestamp>** command. You can configure a `cron` task to automate backup deletion by specifying the following command in the `crontab` file:

```
/usr/bin/pbm delete-backup -f --older-than $(date -d '-1 month' +\%Y-\%m-\%d)
```

This command deletes backups that are older than 30 days. You can change the period by specifying a desired interval for the `date` function.

12.1 Schedule backups with Point-in-Time Recovery running

It is convenient to automate making backups on a schedule using `crond` if you enabled *Point-in-Time Recovery*.

You can configure Point-in-Time Recovery and `crond` in any order. Note, however, that Point-in-Time Recovery will only start running after at least one full backup has been made.

- Make a fresh backup manually. It will serve as the starting point for incremental backups
- Enable point-in-time recovery
- Configure `crond` to run backup snapshots on a schedule

When it is time for another backup snapshot, Percona Backup for MongoDB automatically disables Point-in-Time Recovery and re-enables it once the backup is complete.

CHAPTER

THIRTEEN

UPGRADING PERCONA BACKUP FOR MONGODB

Similar to installing, the recommended and most convenient way to upgrade Percona Backup for MongoDB is from the Percona repository.

You can upgrade Percona Backup for MongoDB to the **latest version** or to a **specific version**. Since all packages of Percona Backup for MongoDB are stored in the same repository, the following steps apply to both upgrade scenarios:

1. Enable Percona repository.
2. Stop **`pbm-agent`**.
3. Install new version packages (the old ones are automatically removed).
4. Start **`pbm-agent`**.

Important notes

1. Backward compatibility between data backup and restore is supported for upgrades within one major version only (for example, from 1.1.x to 1.2.y). When you upgrade Percona Backup for MongoDB over several major versions (for example, from 1.0.x to 1.2.y), we recommend to make a backup right after the upgrade.
2. Percona Backup for MongoDB v1.5.0 and later is incompatible with Percona Backup for MongoDB v1.4.1 and earlier due to different processing of system collections `Users` and `Roles` during backup / restore operations. After the upgrade to Percona Backup for MongoDB v1.5.0 and later, make sure to make a fresh backup.
3. Starting from version 1.3.0, Percona Backup for MongoDB packages are stored in the *pbm* repository and the *tools* repository for backward compatibility.
4. Upgrade Percona Backup for MongoDB on all nodes where it is installed.

13.1 Enable Percona repository

Install the `percona-release` utility or update it to the latest version as described in Percona Software Repositories Documentation.

Enable the repository running the command as root or via **sudo**

```
$ sudo percona-release enable pbm
```

Note: For `apt`-based systems, run **apt-get update** to update the local cache.

13.2 Upgrade Percona Backup for MongoDB using `apt`

Important: Run all commands as root or via **sudo**.

13.2.1 Upgrade to the latest version

1. Stop **pbm-agent**

   ```
   $ sudo systemctl stop pbm-agent
   ```

2. Install new packages

   ```
   $ sudo apt-get install percona-backup-mongodb
   ```

3. Start **pbm-agent**

   ```
   $ sudo systemctl start pbm-agent
   ```

13.2.2 Upgrade to a specific version

1. List available options:

   ```
   $ sudo apt-cache madison percona-backup-mongodb
   ```

   Sample output

   ```
   percona-backup-mongodb | 1.3.1-1.stretch | http://repo.percona.com/tools/apt␣
    ↪stretch/main amd64 Packages
   percona-backup-mongodb | 1.3.0-1.stretch | http://repo.percona.com/tools/apt␣
    ↪stretch/main amd64 Packages
   percona-backup-mongodb | 1.2.1-1.stretch | http://repo.percona.com/tools/apt␣
    ↪stretch/main amd64 Packages
   percona-backup-mongodb | 1.2.0-1.stretch | http://repo.percona.com/tools/apt␣
    ↪stretch/main amd64 Packages
   percona-backup-mongodb | 1.1.3-1.stretch | http://repo.percona.com/tools/apt␣
    ↪stretch/main amd64 Packages
   ```

```
percona-backup-mongodb | 1.1.1-1.stretch | http://repo.percona.com/tools/apt␣
↪stretch/main amd64 Packages
percona-backup-mongodb | 1.0.0-1.stretch | http://repo.percona.com/tools/apt␣
↪stretch/main amd64 Packages
percona-backup-mongodb | 1.0-1.stretch | http://repo.percona.com/tools/apt␣
↪stretch/main amd64 Packages
```

2. Stop **pbm-agent**:

```
$ sudo systemctl stop pbm-agent
```

3. Install a specific version packages. For example, to upgrade to Percona Backup for MongoDB 1.3.1, run the following command:

```
$ sudo apt-get install percona-backup-mongodb=1.3.1-1.stretch
```

4. Start **pbm-agent**:

```
$ sudo systemctl start pbm-agent
```

13.3 Upgrade Percona Backup for MongoDB using `yum`

Important: Run all commands as root or via **sudo**.

13.3.1 Upgrade to the latest version

1. Stop **pbm-agent**

```
$ sudo systemctl stop pbm-agent
```

2. Install new packages

```
$ sudo yum install percona-backup-mongodb
```

3. Start **pbm-agent**

```
$ sudo systemctl start pbm-agent
```

13.3.2 Upgrade to a specific version

1. List available versions

```
$ sudo yum list percona-backup-mongodb --showduplicates
```

Sample output

```
Available Packages
percona-backup-mongodb.x86_64    1.0-1.el7           tools-release-x86_64
percona-backup-mongodb.x86_64    1.0.0-1.el7         tools-release-x86_64
percona-backup-mongodb.x86_64    1.1.1-1.el7         tools-release-x86_64
percona-backup-mongodb.x86_64    1.1.3-1.el7         tools-release-x86_64
percona-backup-mongodb.x86_64    1.2.0-1.el7         tools-release-x86_64
percona-backup-mongodb.x86_64    1.2.1-1.el7         tools-release-x86_64
percona-backup-mongodb.x86_64    1.3.0-1.el7         tools-release-x86_64
percona-backup-mongodb.x86_64    1.3.1-1.el7         tools-release-x86_64
```

2. Stop **pbm-agent**:

```
$ sudo systemctl stop pbm-agent
```

3. Install a specific version packages. For example, to upgrade Percona Backup for MongoDB to version 1.3.1, use the following command:

```
$ sudo yum install percona-backup-mongodb-1.3.1-1.el7
```

4. Start **pbm-agent**:

```
$ sudo systemctl start pbm-agent
```

CHAPTER

FOURTEEN

TROUBLESHOOTING PERCONA BACKUP FOR MONGODB

Percona Backup for MongoDB provides troubleshooting tools to operate data backups.

14.1 pbm-speed-test

pbm-speed-test allows field-testing compression and backup upload speed. You can use it:

- to check performance before starting a backup;
- to find out what slows down the running backup.

By default, **pbm-speed-test** operates with fake semi random data documents. To run **pbm-speed-test** on a real collection, provide a valid *MongoDB connection URI string* for the `--mongodb-uri` flag.

Run **pbm-speed-test** for the full set of available commands.

14.1.1 Compression test

```
$ pbm-speed-test compression --compression=s2 --size-gb 10
Test started ....
10.00GB sent in 8s.
Avg upload rate = 1217.13MB/s.
```

pbm-speed-test compression uses the compression library from the config file and sends a fake semi random data document (1 GB by default) to the black hole storage. (Use the `pbm config` command to change the compression library).

To test compression on a real collection, pass the `--sample-collection` flag with the <my_db.my_collection> value.

Run `pbm-speed-test compression --help` for the full set of supported flags:

```
$ pbm-speed-test compression --help
usage: pbm-speed-test compression

Run compression test
```

```
Flags:
      --help                       Show context-sensitive help (also try
                                   --help-long and --help-man).
      --mongodb-uri=MONGODB-URI  MongoDB connection string
  -c, --sample-collection=SAMPLE-COLLECTION
                                   Set collection as the data source
  -s, --size-gb=SIZE-GB          Set data size in GB. Default 1
      --compression=s2           Compression type
                                   <none>/<gzip>/<snappy>/<lz4>/<s2>/<pgzip>
```

14.1.2 Upload speed test

```
$ pbm-speed-test storage --compression=s2
Test started
1.00GB sent in 1s.
Avg upload rate = 1744.43MB/s.
```

`pbm-speed-test storage` sends the semi random data (1 GB by default) to the remote storage defined in the config file. Pass the `--size-gb` flag to change the data size.

To run the test with the real collection's data instead of the semi random data, pass the `--sample-collection` flag with the <my_db.my_collection> value.

Run `pbm-speed-test storage --help` for the full set of available flags:

```
$ pbm-speed-test storage --help
usage: pbm-speed-test storage

Run storage test

Flags:
      --help                       Show context-sensitive help (also try --help-long␣
↪and --help-man).
      --mongodb-uri=MONGODB-URI  MongoDB connection string
  -c, --sample-collection=SAMPLE-COLLECTION
                                   Set collection as the data source
  -s, --size-gb=SIZE-GB          Set data size in GB. Default 1
      --compression=s2           Compression type <none>/<gzip>/<snappy>/<lz4>/<s2>/
↪<pgzip>
```

14.2 Backup progress logs

If you have a large backup you can track backup progress in **`pbm-agent`** logs. A line is appended every minute showing bytes copied vs. total size for the current collection.

```
# Start a backup
$ pbm backup
#Check backup progress
journalctl -u pbm-agent.service
2020/05/06 21:31:12 Backup 2020-05-06T18:31:12Z started on node rs2/localhost:28018
2020-05-06T21:31:14.797+0300 writing admin.system.users to archive on stdout
2020-05-06T21:31:14.799+0300 done dumping admin.system.users (2 documents)
2020-05-06T21:31:14.800+0300 writing admin.system.roles to archive on stdout
2020-05-06T21:31:14.807+0300 done dumping admin.system.roles (1 document)
```

```
2020-05-06T21:31:14.807+0300 writing admin.system.version to archive on stdout
2020-05-06T21:31:14.815+0300 done dumping admin.system.version (3 documents)
2020-05-06T21:31:14.816+0300 writing test.testt to archive on stdout
2020-05-06T21:31:14.829+0300 writing test.testt2 to archive on stdout
2020-05-06T21:31:14.829+0300 writing config.cache.chunks.config.system.sessions to␣
↪archive on stdout
2020-05-06T21:31:14.832+0300 done dumping config.cache.chunks.config.system.sessions␣
↪(1 document)
2020-05-06T21:31:14.834+0300 writing config.cache.collections to archive on stdout
2020-05-06T21:31:14.835+0300 done dumping config.cache.collections (1 document)
2020/05/06 21:31:24 [##......................]  test.testt  130841/1073901  (12.2%)
2020/05/06 21:31:24 [##########..............]  test.testt2  131370/300000  (43.8%)
2020/05/06 21:31:24
2020/05/06 21:31:34 [#####...................]  test.testt  249603/1073901  (23.2%)
2020/05/06 21:31:34 [##################.....]  test.testt2  249603/300000  (83.2%)
2020/05/06 21:31:34
2020/05/06 21:31:37 [########################]  test.testt2  300000/300000  (100.0%)
```

CHAPTER

FIFTEEN

UNINSTALLING PERCONA BACKUP FOR MONGODB

To uninstall Percona Backup for MongoDB perform the following steps:

1. Check no backups are currently in progress in the output of **`pbm list`**.
2. Before the next 2 steps make sure you know where the remote backup storage is, so you can delete backups made by Percona Backup for MongoDB. If it is S3-compatible object storage you will need to use another tool such as Amazon AWS's "aws s3", Minio's `mc`, the web AWS Management Console, etc. to do that once Percona Backup for MongoDB is uninstalled. Don't forget to note the connection credentials before they are deleted too.
3. Uninstall the **`pbm-agent`** and `pbm` executables. If you installed using a package manager, see *Installing Percona Backup for MongoDB* for relevant package names and commands for your OS distribution.
4. Drop the *PBM control collections*.
5. Drop the PBM database user. If this is a cluster the dropUser command will need to be run on each shard as well as in the config server replica set.
6. (Optional) Delete the backups from the remote backup storage.

Part VI

Reference

CHAPTER

SIXTEEN

PBM COMMANDS

pbm CLI is the command line utility to control the backup system. This page describes pbm commands available in Percona Backup for MongoDB.

For how to get started with Percona Backup for MongoDB, see *Initial setup*.

pbm help

Returns the help information about pbm commands.

pbm config

Sets, changes or lists Percona Backup for MongoDB configuration.

The command has the following syntax:

```
$ pbm config [<flags>] [<key>]
```

The command accepts the following flags:

Flag	Description
--force-resync	Resync backup list with the current storage
--list	List current settings
--file=FILE	Upload the config information from a YAML file
--set=SET	Set a new config option value. Specify the option in the <key.name=value> format.

pbm backup

Creates a backup snapshot and saves it in the remote backup storage.

The command has the following syntax:

```
$ pbm backup [<flags>]
```

For more information about using pbm backup, see *Starting a backup*

The command accepts the following flags:

Flag	Description
--compression	Create a backup with compression. Supported compression methods: gzip, snappy, lz4, s2, pgzip. Default: s2 The none value means no compression is done during backup.

pbm restore

Restores database from a specified backup / to a specified point in time.

The command has the following syntax:

```
$ pbm restore [<flags>] [<backup_name>]
```

For more information about using `pbm restore`, see *Restoring a backup*.

The command accepts the following flags:

Flag	Description
`--time=TIME`	Restores the database to the specified point in time. Available if *Point-in-Time Recovery* is enabled.

pbm cancel-backup

Cancels a running backup. The backup is marked as canceled in the backup list.

pbm list

Provides the list of backups. In versions 1.3.4 and earlier, the command lists all backups and their states. Backup states are the following:

- In progress - A backup is running
- Canceled - A backup was canceled
- Error - A backup was finished with an error
- No status means a backup is complete

As of version 1.4.0, only successfully completed backups are listed. To view currently running backup information, run *pbm status*.

When *Point-in-Time Recovery* is enabled, the `pbm list` also provides the list of valid time ranges for recovery and point-in-time recovery status.

The command has the following syntax:

```
$ pbm list [<flags>]
```

The command accepts the following flags:

Flag	Description
`--restore`	Shows last N restores.
`--size=0`	Shows last N backups.

pbm delete-backup

Deletes the specified backup or all backups that are older than the specified time. The command deletes backups that are not running regardless of the remote backup storage being used.

The following is the command syntax:

```
$ pbm delete-backup [<flags>] [<name>]
```

The command accepts the following flags:

Flag	Description
`--older-than=TIMESTAMP`	Deletes backups older than date / time specified in the format: • `%Y-%M-%DT%H:%M:%S` (e.g. 2020-04-20T13:13:20) or • `%Y-%M-%D` (e.g. 2020-04-20)
`--force`	Forcibly deletes backups without asking for user's confirmation

pbm version

Shows the version of Percona Backup for MongoDB.

The command accepts the following flags:

Flag	Description
`--short`	Shows only version info
`--commit`	Shows only git commit info
`--format=""`	Shows version info as a standard output or a JSON object. Supported values: `""`, `json`.

pbm status

Shows the status of Percona Backup for MongoDB. The output provides the following information:

- pbm-agent processes version and state,
- currently running backups or restores
- backups stored in the remote storage
- Point-in-Time Recovery status
- Valid time ranges for point-in-time recovery and the data size

CHAPTER

SEVENTEEN

CONFIGURATION FILE OPTIONS

This page describes configuration file options available in Percona Backup for MongoDB. For how to use configuration file, see *Percona Backup for MongoDB configuration in a cluster (or non-sharded replica set)*.

17.1 Remote backup storage options

Percona Backup for MongoDB supports the following types of remote storages: - S3-compatible storage, - Microsoft Azure Blob storage, and - filesystem.

Percona Backup for MongoDB should work with other S3-compatible storage but was only tested with the following ones:

- Amazon Simple Storage Service,
- Google Cloud Storage,
- MinIO.

`storage.type`

Type string

Required YES

Remote backup storage type. Supported values: s3, filesystem, azure.

17.1.1 S3 type storage options

```
storage:
  type: s3
  s3:
    region: <string>
    bucket: <string>
    prefix: <string>
    credentials:
      access-key-id: <your-access-key-id-here>
      secret-access-key: <your-secret-key-here>
    serverSideEncryption:
      sseAlgorithm: aws:kms
      kmsKeyID: <your-kms-key-here>
```

storage.s3.provider

Type string

Required NO

The storage provider's name. Supported values: aws, gcs

storage.s3.bucket

Type string

Required YES

The name of the storage *bucket*. See the AWS Bucket naming rules and GCS bucket naming guidelines for bucket name requirements

storage.s3.region

Type string

Required YES (for AWS and GCS)

The location of the storage bucket. Use the AWS region list and GCS region list to define the bucket region

storage.s3.prefix

Type string

Required NO

The path to the data directory on the bucket. If undefined, backups are stored in the bucket root directory

storage.s3.endpointUrl

Type string

Required YES (for MinIO and GCS)

The URL to access the bucket. The default value for GCS is `https://storage.googleapis.com`

storage.s3.credentials.access-key-id

Type string

Required YES

Your access key to the storage bucket

storage.s3.credentials.secret-access-key

Type string

Required YES

The key to sign your programmatic requests to the storage bucket

storage.s3.uploadPartSize

Type int

Required NO

The size of data chunks in bytes to be uploaded to the storage bucket. Default: 10MB

Percona Backup for MongoDB automatically increases the `uploadPartSize` value if the size of the file to be uploaded exceeds the max allowed file size. (The max allowed file size is calculated with the default values of uploadPartSize * maxUploadParts and is appr. 97,6 GB).

The `uploadPartSize` value is printed in the *pbm-agent log*.

By setting this option, you can manually adjust the size of data chunks if Percona Backup for MongoDB failed to do it for some reason. The defined `uploadPartSize` value overrides the default value and is used for calculating the max allowed file size

Server-side encryption options

serverSideEncryption.sseAlgorythm

Type string

The key management mode used for server-side encryption

Supported value: `aws:kms`

serverSideEncryption.kmsKeyID

Type string

Your customer-managed key

17.1.2 Filesystem storage options

```
storage:
  type: filesystem
  filesystem:
    path: <string>
```

storage.filesystem.path

Type string

Required YES

The path to the backup directory

17.1.3 Microsoft Azure Blob storage options

```
storage:
  type: azure
  azure:
    account: <string>
    container: <string>
    prefix: <string>
```

```
    credentials:
      key: <your-access-key>
```

storage.azure.account

Type string

Required YES

The name of your storage account.

storage.azure.container

Type string

Required YES

The name of the storage *container*. See the Container names for naming conventions.

storage.azure.prefix

Type string

Required NO

The path (sub-folder) to the backups inside the container. If undefined, backups are stored in the container root directory.

.

storage.azure.credentials.key

Type string

Required YES

Your access key to authorize access to data in your storage account.

17.2 Point-in-time recovery options

```
pitr:
  enabled: <boolean>
```

pitr.enabled

Type boolean

Enables point-in-time recovery

17.3 Backup options

```
backup:
  priority:
    "localhost:28019": 2.5
    "localhost:27018": 2.5
    "localhost:27020": 2.0
    "localhost:27017": 0.1
```

priority

Type array of strings

The list of `mongod` nodes and their priority for making backups. The node with the highest priority is elected for making a backup. If several nodes have the same priority, the one among them is randomly elected to make a backup.

If not set, the replica set nodes have the default priority as follows:

- hidden nodes - 2.0,
- secondary nodes - 1.0,
- primary node - 0.5.

17.4 Restore options

```
restore:
  batchSize: <int>
  numInsertionWorkers: <int>
```

batchSize

Type int

Default 500

The number of documents to buffer.

numInsertionWorkers

Type int

Default 10

The number of workers that add the documents to buffer.

CHAPTER

EIGHTEEN

SUBMITTING BUG REPORTS OR FEATURE REQUESTS

If you find a bug in Percona Backup for MongoDB, you can submit a report to the JIRA issue tracker for Percona Backup for MongoDB.

Start by searching the open tickets for a similar report. If you find that someone else has already reported your problem, then you can upvote that report to increase its visibility.

If there is no existing report, submit a report following these steps:

1. Sign in to JIRA issue tracker. You will need to create an account if you do not have one.
2. In the *Summary*, *Description*, *Steps To Reproduce*, *Affects Version* fields describe the problem you have detected. For PBM the important diagnostic information is: log files from the pbm-agents; a dump of the PBM control collections.

As a general rule of thumb, try to create bug reports that are:

- *Reproducible*: describe the steps to reproduce the problem.
- *Specific*: include the version of Percona Backup for MongoDB, your environment, and so on.
- *Unique*: check if there already exists a JIRA ticket to describe the problem.
- *Scoped to a Single Bug*: only report one bug in one JIRA ticket.

CHAPTER

NINETEEN

GLOSSARY

ACID Set of properties that guarantee database transactions are processed reliably. Stands for *Atomicity*, *Consistency*, *Isolation*, *Durability*.

Amazon S3 Amazon S3 (Simple Storage Service) is an object storage service provided through a web service interface offered by Amazon Web Services.

Atomicity Atomicity means that database operations are applied following a "all or nothing" rule. A transaction is either fully applied or not at all.

Blob A blob stands for Binary Large Object, which includes objects such as images and multimedia files. In other words these are various data files that you store in Microsoft's data storage platform. Blobs are organized in *containers* which are kept in Azure Blob storage under your storage account.

Bucket A bucket is a container on the s3 remote storage that stores backups.

Collection A collection is the way data is organized in MongoDB. It is analogous to a table in relational databases.

Consistency In the context of backup and restore, consistency means that the data restored will be consistent in a given point in time. Partial or incomplete writes to disk of atomic operations (for example, to table and index data structures separately) won't be served to the client after the restore. The same applies to multi-document transactions, that started but didn't complete by the time the backup was finished.

Container A container is like a directory in Azure Blob storage that contains a set of *blobs*.

Durability Once a transaction is committed, it will remain so.

GCP GCP (Google Cloud Platform) is the set of services, including storage service, that runs on Google Cloud infrastructure.

Isolation The Isolation requirement means that no transaction can interfere with another.

Jenkins Jenkins is a continuous integration system that we use to help ensure the continued quality of the software we produce. It helps us achieve the aims of:

- no failed tests in trunk on any platform,
- aid developers in ensuring merge requests build and test on all platforms,
- no known performance regressions (without a damn good explanation).

MinIO MinIO is a cloud storage server compatible with *Amazon S3*, released under Apache License v2.

Oplog Oplog (operations log) is a fixed-size collection that keeps a rolling record of all operations that modify data in the database.

Oplog slice A compressed bundle of *oplog* entries stored in the Oplog Store database in MongoDB. The oplog size captures an approximately 10-minute frame. For a snapshot, the oplog size is defined by the time that the slowest replica set member requires to perform mongodump.

OpID A unique identifier of an operation such as backup, restore, resync. When a pbm-agent starts processing an operation, it acquires a lock and an opID. This prevents processing the same operation twice (for example, if there are network issues in distributed systems). Using opID as a log filter allows viewing logs for an operation in progress.

pbm-agent A `pbm-agent` is a *PBM* process running on the mongod node for backup and restore operations. A pbm-agent instance is required for every mongod node (including replica set secondary members and config server replica set nodes).

pbm CLI Command-line interface for controlling the backup system. PBM CLI can connect to several clusters so that a user can manage backups on many clusters.

PBM Control collections PBM Control collections are *collections* with config, authentication data and backup states. They are stored in the admin db in the cluster or non-sharded replica set and serve as the communication channel between *pbm-agent* and *pbm CLI*. *pbm CLI* creates a new pbmCmd document for a new operation. *pbm-agents* monitor it and update as they process the operation.

Percona Backup for MongoDB Percona Backup for MongoDB (PBM) is a low-impact backup solution for MongoDB non-sharded replica sets and clusters. It supports both *Percona Server for MongoDB* and MongoDB Community Edition.

Percona Server for MongoDB Percona Server for MongoDB is a drop-in replacement for MongoDB Community Edition with enterprise-grade features.

Point-in-Time Recovery Point-in-Time Recovery is restoring the database up to a specific moment in time. The data is restored from the backup snapshot and then events that occurred to the data are replayed from oplog.

Replica set A replica set is a group of mongod nodes that host the same data set.

S3 compatible storage This is the storage that is built on the *S3* API.

Server-side encryption Server-side encryption is the encryption of data by the remote storage server as it receives it. The data is encrypted when it is written to S3 bucket and decrypted when you access the data.

CHAPTER

TWENTY

COPYRIGHT AND LICENSING INFORMATION

20.1 Documentation Licensing

CHAPTER

TWENTYONE

TRADEMARK POLICY

This Trademark Policy is to ensure that users of Percona-branded products or services know that what they receive has really been developed, approved, tested and maintained by Percona. Trademarks help to prevent confusion in the marketplace, by distinguishing one company's or person's products and services from another's.

Percona owns a number of marks, including but not limited to Percona, XtraDB, Percona XtraDB, XtraBackup, Percona XtraBackup, Percona Server, and Percona Live, plus the distinctive visual icons and logos associated with these marks. Both the unregistered and registered marks of Percona are protected.

Use of any Percona trademark in the name, URL, or other identifying characteristic of any product, service, website, or other use is not permitted without Percona's written permission with the following three limited exceptions.

First, you may use the appropriate Percona mark when making a nominative fair use reference to a bona fide Percona product.

Second, when Percona has released a product under a version of the GNU General Public License ("GPL"), you may use the appropriate Percona mark when distributing a verbatim copy of that product in accordance with the terms and conditions of the GPL.

Third, you may use the appropriate Percona mark to refer to a distribution of GPL-released Percona software that has been modified with minor changes for the sole purpose of allowing the software to operate on an operating system or hardware platform for which Percona has not yet released the software, provided that those third party changes do not affect the behavior, functionality, features, design or performance of the software. Users who acquire this Percona-branded software receive substantially exact implementations of the Percona software.

Percona reserves the right to revoke this authorization at any time in its sole discretion. For example, if Percona believes that your modification is beyond the scope of the limited license granted in this Policy or that your use of the Percona mark is detrimental to Percona, Percona will revoke this authorization. Upon revocation, you must immediately cease using the applicable Percona mark. If you do not immediately cease using the Percona mark upon revocation, Percona may take action to protect its rights and interests in the Percona mark. Percona does not grant any license to use any Percona mark for any other modified versions of Percona software; such use will require our prior written permission.

Neither trademark law nor any of the exceptions set forth in this Trademark Policy permit you to truncate, modify or otherwise use any Percona mark as part of your own brand. For example, if XYZ creates a modified version of the Percona Server, XYZ may not brand that modification as "XYZ Percona Server" or "Percona XYZ Server", even if that modification otherwise complies with the third exception noted above.

In all cases, you must comply with applicable law, the underlying license, and this Trademark Policy, as amended from time to time. For instance, any mention of Percona trademarks should include the full trademarked name, with proper spelling and capitalization, along with attribution of ownership to Percona Inc. For example, the full proper name for XtraBackup is Percona XtraBackup. However, it is acceptable to omit the word "Percona" for brevity on the second and subsequent uses, where such omission does not cause confusion.

In the event of doubt as to any of the conditions or exceptions outlined in this Trademark Policy, please contact trademarks@percona.com for assistance and we will do our very best to be helpful.

Part VII

Release notes

CHAPTER

TWENTYTWO

RELEASE NOTES

22.1 *Percona Backup for MongoDB* 1.5.0

Date May 10, 2021

Installation Installing Percona Backup for MongoDB

Percona Backup for MongoDB is a distributed, low-impact solution for consistent backups of MongoDB sharded clusters and replica sets. This is a tool for creating consistent backups across a MongoDB sharded cluster (or a single replica set), and for restoring those backups to a specific point in time.

Important: Backups made with previous versions are incompatible for restore with Percona Backup for MongoDB 1.5.0. This is because processing of system collections `users` and `roles` has changed during backup / restore operations. For details, refer to *Restoring a backup* and PBM-636.

22.1.1 New Features

- PBM-596: Azure Blob Storage support
- PBM-488: Create weight or tag method to influence with `pbm-agent` node will do backups

22.1.2 Improvements

- PBM-662: Show PITR Status based on `admin.pbmLock` instead of config settings
- PBM-494: Prefer a (healthy) hidden secondary to any other node in automatic selection

22.1.3 Bugs Fixed

- PBM-642: Display `priority=0` members on agent list in `pbm status` output
- PBM-636: Different collection UUID after restore (Thanks to Nikolay for reporting this issue and Dmitry Kuzmin for contributing)
- PBM-646: Stop the balancer during backup to make sure it doesn't start running during restore
- PBM-635: Wait for the leader's metadata before starting backups
- PBM-490: Use cluster time for the snapshot start time

22.2 *Percona Backup for MongoDB* 1.4.1

Date January 28, 2021

Installation Installing Percona Backup for MongoDB

Percona Backup for MongoDB is a distributed, low-impact solution for consistent backups of MongoDB sharded clusters and replica sets. This is a tool for creating consistent backups across a MongoDB sharded cluster (or a single replica set), and for restoring those backups to a specific point in time.

22.2.1 Improvements

- PBM-621: Show incomplete backups in `pbm status` output
- PBM-619: Optimise response time from storage for `pbm status`
- PBM-615: Check backup validity for current cluster
- PBM-608: Enable Kerberos authentication for PBM by adding support for GSSAPI
- PBM-478: Prevent restore from incomplete backup
- PBM-610: Fix response time from GCS for `pbm status` command

22.2.2 Bugs Fixed

- PBM-618: Check for the complete file set in backup snapshot before processing it

22.3 *Percona Backup for MongoDB* 1.4.0

Date December 24, 2020

Installation Installing Percona Backup for MongoDB

Percona Backup for MongoDB is a distributed, low-impact solution for consistent backups of MongoDB sharded clusters and replica sets. This is a tool for creating consistent backups across a MongoDB sharded cluster (or a single replica set), and for restoring those backups to a specific point in time.

22.3.1 New Features

- PBM-345: Centralize logs
- PBM-435: `pbm status` command

22.3.2 Improvements

- PBM-572: Change backup 'name' in 'pbm list' etc to be consistent time (~= end time) rather than start time
- PBM-556: Introduce operation ID

22.3.3 Bugs Fixed

- PBM-595: Shard backup with different rset name
- PBM-604: Compression flag for 'pbm list' command doesn't change the output
- PBM-602: Empty PITR files are created on storage if PBM fails to upload oplog chunk due to insufficient range
- PBM-597: Properly handle mongo fail while PITR slicing is enabled

22.4 *Percona Backup for MongoDB* 1.3.4

Date November 19, 2020

Installation Installing Percona Backup for MongoDB

Percona Backup for MongoDB is a distributed, low-impact solution for consistent backups of MongoDB sharded clusters and replica sets. This is a tool for creating consistent backups across a MongoDB sharded cluster (or a single replica set), and for restoring those backups to a specific point in time.

22.4.1 Improvements

- PBM-586: Add a request timeout to the S3 downloader during the restore
- PBM-584: Ignore shard configuration during the restore

22.4.2 Bugs Fixed

- PBM-555: Fix the " error demultiplexing archive" error during restore by downloading backup from s3 storage in chunks
- PBM-460: Restore fails with conflicting namespace destinations (Thanks to user pedroalb for reporting this issue)

22.5 *Percona Backup for MongoDB* 1.3.3

Date November 4, 2020

Installation Installing Percona Backup for MongoDB

Percona Backup for MongoDB is a distributed, low-impact solution for consistent backups of MongoDB sharded clusters and replica sets. This is a tool for creating consistent backups across a MongoDB sharded cluster (or a single replica set), and for restoring those backups to a specific point in time.

22.5.1 Bugs Fixed

- PBM-575: `mongodump` connects to the primary node

22.6 *Percona Backup for MongoDB* 1.3.2

Date October 14, 2020

Installation Installing Percona Backup for MongoDB

Percona Backup for MongoDB is a distributed, low-impact solution for consistent backups of MongoDB sharded clusters and replica sets. This is a tool for creating consistent backups across a MongoDB sharded cluster (or a single replica set), and for restoring those backups to a specific point in time.

22.6.1 New Features

- PBM-426: Add AWS KMS key encryption/decryption for S3 buckets

  Config format

```
storage:
  s3:
    serverSideEncryption:
      sseAlgorithm: "aws:kms"
      kmsKeyID: "........"
```

  (Thanks to user pedroalb for reporting this issue)

22.6.2 Improvements

- PBM-568: Print uploadPartSize value to log during backup
- PBM-560: Use s2 compression as default for `pbm-speed-test` instead of gzip

22.6.3 Bugs Fixed

- PBM-485: Fix backups to S3 failing with `MaxUploadParts` limit by auto-adjusting `uploadPartSize` value (Thanks to user pedroalb for reporting this issue)
- PBM-559: pbm-agent runs out of memory while doing restore of large backup (Thanks to user Simon Bernier St-Pierre for reporting this issue)
- PBM-562: Correct calculation of available PITR time ranges by pbm list
- PBM-561: Fix setting of numeric options in config
- PBM-547: Allow deleting backups from local filesystem by moving delete operations to pbm-agents

22.7 *Percona Backup for MongoDB* 1.3.1

Date September, 2020

Installation Installing Percona Backup for MongoDB

Percona Backup for MongoDB is a distributed, low-impact solution for consistent backups of MongoDB sharded clusters and replica sets. This is a tool for creating consistent backups across a MongoDB sharded cluster (or a single replica set), and for restoring those backups to a specific point in time.

22.7.1 Bugs Fixed

- PBM-542: Fix backup folder permissions on filesystem storage for Point-in-Time recovery

22.8 *Percona Backup for MongoDB* 1.3.0

Date August 26, 2020

Installation Installing Percona Backup for MongoDB

Percona Backup for MongoDB is a distributed, low-impact solution for consistent backups of MongoDB sharded clusters and replica sets. This is a tool for creating consistent backups across a MongoDB sharded cluster (or a single replica set), and for restoring those backups to a specific point in time.

22.8.1 New Features

- PBM-455: Add oplog archiver thread for PITR
- PBM-491: Modify "pbm restore" to accept arbitrary point in time when PITR oplog archives available

22.8.2 Improvements

- PBM-526: Add pbm version information to the backup metadata

22.9 *Percona Backup for MongoDB* 1.2.1

Date July 27, 2020

Installation Installing Percona Backup for MongoDB

Percona Backup for MongoDB is a distributed, low-impact solution for consistent backups of MongoDB sharded clusters and replica sets. This is a tool for creating consistent backups across a MongoDB sharded cluster (or a single replica set).

22.9.1 Bugs Fixed

- PBM-509: Include "pbm-speed-test" binary for debian packages

22.10 *Percona Backup for MongoDB* 1.2.0

Date May 13, 2020

Installation Installing Percona Backup for MongoDB

Percona Backup for MongoDB is a distributed, low-impact solution for consistent backups of MongoDB sharded clusters and replica sets. This is a tool for creating consistent backups across a MongoDB sharded cluster (or a single replica set).

22.10.1 New Features

- PBM-348: Add ability to delete old backups
- PBM-447: pbm-speed-test: Add a tool to field-test compression and upload speeds

22.10.2 Improvements

- PBM-431: Raise dump output speed through compression tuning, parallelization
- PBM-461: s2 is set as the default compression mechanism
- PBM-429: Periodic backup progress messages added to pbm-agent logs
- PBM-140: Added ability to cancel a backup

22.10.3 Bugs Fixed

- PBM-451: Resync didn't work if storage type was set to filesystem

22.11 Percona Backup for MongoDB 1.1.3

Date April 14, 2020

Installation *Installing Percona Backup for MongoDB*

22.11.1 Improvements

- PBM-424: Remove the `--mongodb-uri` arg from `pbm-agent.service` unit file
- PBM-419: Resolve restore-blocking issues related to admin.system.version
- PBM-417: Improve pbm control collection etc. metadata for restores

22.11.2 Bugs Fixed

- PBM-425: pbm-agent could fail when restoring
- PBM-430: S3 store resync didn't work if the store had a prefix
- PBM-438: `pbm list --size=5` worked in reverse

22.12 Percona Backup for MongoDB 1.1.1

Date January 31, 2020

Installation *Installing Percona Backup for MongoDB*

Percona Backup for MongoDB is a distributed, low-impact solution for consistent backups of MongoDB sharded clusters and replica sets. This is a tool for creating consistent backups across a MongoDB sharded cluster (or a single replica set), and for restoring those backups to a specific point in time. The project was inspired by (and intends to replace) the Percona-Lab/mongodb_consistent_backup tool.

Percona Backup for MongoDB supports Percona Server for MongoDB or MongoDB Community Server version 3.6 or higher with MongoDB replication enabled. Binaries for the supported platforms as well as the tarball with source code are available from the Percona Backup for MongoDB download page. For more information about Percona Backup for MongoDB and the installation steps, see the *documentation*.

22.12.1 Bugs Fixed

- PBM-407: Very large collections experienced timeout due to full-collection scan for a preliminary count
- PBM-414: The upload on Google cloud storage was broken with "InvalidArgument: Invalid argument. status code: 400"
- PBM-409: Restore failed with "incompatible auth version with target server"

22.13 Percona Backup for MongoDB 1.1.0

Percona is happy to announce the release of Percona Backup for MongoDB 1.1.0 on January 16, 2020.

Percona Backup for MongoDB is a distributed, low-impact solution for consistent backups of MongoDB sharded clusters and replica sets. This is a tool for creating consistent backups across a MongoDB sharded cluster (or a single replica set), and for restoring those backups to a specific point in time. The project was inspired by (and intends to replace) the Percona-Lab/mongodb_consistent_backup tool.

Percona Backup for MongoDB supports Percona Server for MongoDB or MongoDB Community Server version 3.6 or higher with MongoDB replication enabled. Binaries for the supported platforms as well as the tarball with source code are available from the Percona Backup for MongoDB download page. For more information about Percona Backup for MongoDB and the installation steps, see the *documentation*.

Percona Backup for MongoDB 1.1.0 introduces the new `pbm config` command to enable configuring the store from the command line in addition to the configuration file. This command effectively replaces *pbm store* which was only able to read store configuration from the configuration file.

```
$ pbm config --set storage.s3.bucket="operator-testing"
```

22.13.1 New Features

- PBM-344: New *pbm config* command to support configuring the store from the command line.

22.13.2 Improvements

- PBM-361: Improved the processing of timestamps when using oplog.

22.13.3 Bugs Fixed

- PBM-214: `pbm-agent` could crash with restore command running forever, if the primary node became unavailable during the *restore* operation.
- PBM-279: `pbm-agent` could be started with an invalid config file.
- PBM-338: Backups that failed could appear in the output of the *pbm list* command.
- PBM-362: The `pbm backup` could fail when called from the primary node if there were no healthy secondaries.

- PBM-369: ReplicaSets could not establish connections when TLS was used in the cluster.

22.14 Percona Backup for MongoDB 1.0.0

Percona is happy to announce the GA release of our latest software product Percona Backup for MongoDB 1.5 on September 19, 2019.

Percona Backup for MongoDB is a distributed, low-impact solution for consistent backups of MongoDB sharded clusters and replica sets. This is a tool for creating consistent backups across a MongoDB sharded cluster (or a single replica set), and for restoring those backups to a specific point in time. The project was inspired by (and intends to replace) the Percona-Lab/mongodb_consistent_backup tool.

Percona Backup for MongoDB supports Percona Server for MongoDB or MongoDB Community Server version 3.6 or higher with MongoDB replication enabled. Binaries for the supported platforms as well as the tarball with source code are available from the Percona Backup for MongoDB download page. For more information about Percona Backup for MongoDB and the installation steps, see the *documentation*.

Percona Backup for MongoDB 1.0.0 features the following:

- The architecture and the authentication of Percona Backup for MongoDB have been simplified compared to the previous release.
- Stores backup data on Amazon Simple Storage Service or compatible storages, such as MinIO.
- The output of **`pbm list`** shows all backups created from the connected MongoDB sharded cluster or replica set.

22.15 Percona Backup for MongoDB 0.5.0

Percona is pleased to announce the early release of Percona Backup for MongoDB 0.5.0 of our latest software product on June 17, 2019. The GA version of Percona Backup for MongoDB is scheduled to be released later in 2019.

Percona Backup for MongoDB is a distributed, low-impact solution for consistent backups of MongoDB sharded clusters and replica sets. This is a tool for creating consistent backups across a MongoDB sharded cluster (or a single replica set), and for restoring those backups to a specific point in time. Percona Backup for MongoDB uses a distributed client/server architecture to perform backup/restore actions.

The project was inspired by (and intends to replace) the Percona-Lab/mongodb_consistent_backup tool.

Percona Backup for MongoDB supports Percona Server for MongoDB or MongoDB Community Server version 3.6 or higher with MongoDB replication enabled. Binaries for the supported platforms as well as the tarball with source code are available from the Percona Backup for MongoDB download page <https://www.percona.com/downloads/percona-backup-mongodb/LATEST/>‘_. For more information about Percona Backup for MongoDB and the installation steps, see the documentation.

Percona Backup for MongoDB 0.5.0 features the following:

- Enables storing backup metadata on Amazon Simple Storage Service storages.
- The API of Percona Backup for MongoDB introduces HTTP basic authentication to prevent an unauthorized user from running backups or restoring data if they manage to access the API port.
- To optimize the usage of network resources, the pbm-agent on `mongos` is not needed any more and backup-coordinator automatically establishes connection to the appropriate `mongos` instance.
- The output of `pbmctl list nodes` now includes the replica set name and informs the backup status of the node.

Percona doesn't recommend this release for production as its API and configuration fields are still likely to change. It only features a basic API level security. Please report any bugs you encounter in our bug tracking system.

22.15.1 New Features and Improvements

- 93: Support storage of backup metadata on AWS S3.
- 99: **`pbm-agent`** is deprecated on `mongos`.
- 105: Log a warning if a Primary node-type is used for a backup
- 122: Include the replica set name to the output of `pmbctl list nodes`
- 130: Add HTTP Basic Authentication to gRPC servers (API and RPC)
- 139: Support listing backup status in the output of `pmbctl list nodes`
- 170: Enable setting the 'stopOnError' attribute in `mongorestore` to ensure consistency of the data being restored.

INDEX