Percona Backup for MongoDB Documentation

Release 1.2.0

May 13, 2020

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.

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

Features

- Backup and restore for both classic, non-sharded replicasets and clusters
- Simple command-line management utility
- Oplog capture that provides data consistency for any replica set
- Oplog capture synchronization in cluster backups provides cross-cluster consistency.
- Simple, integrated-with-MongoDB authentication
- No need to install a coordination service on a separate server.
- Use any S3-compatible storage
- Users with classic, locally-mounted remote filesystem backup servers can use 'filesystem' instead of 's3' storage type.

Part I

Percona Backup for MongoDB Intro

How to use Percona Backup for MongoDB: going back in time (`pbm restore`)

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.

For example, imagine your web application's update was released on Sunday, June 9th 23:00 EDT but, by 11:23 Monday, 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 --mongodb-uri="mongodb://pbmuser:secretpwd@mongocsvr1:27018,
↪mongocsvr2:27018,mongocsvr3:27018/?replicaSet=configrs"
```

Output

```
$ pbm list
2019-09-10T07:04:14Z
2019-09-09T07:03:50Z
2019-09-08T07:04:21Z
2019-09-07T07:04:18Z
```

The most recent daily backup is 03:04 EDT (07:04 UTC), which would include 4 hours of damage caused by the bug. Let's restore the one before that:

```
$ pbm restore 2019-06-09T07:03:50Z --mongodb-uri="mongodb://
↪pbmuser:secretpwd@mongocsvr1:27018,mongocsvr2:27018,mongocsvr3:27018/?
↪replicaSet=configrs"
```

Next time there is an application release, it might be best to make an extra backup manually just before:

```
$ pbm backup --mongodb-uri="mongodb://pbmuser:secretpwd@mongocsvr1:27018,
↪mongocsvr2:27018,mongocsvr3:27018/?replicaSet=configrs"
```

Percona Backup for MongoDB is an uncomplicated command-line tool by design. The full set of commands:

```
$ pbm help
usage: pbm [<flags>] <command> [<args> ...]

Percona Backup for MongoDB

Flags:
  --help                     Show context-sensitive help (also try --help-long
                             and --help-man).
  --mongodb-uri=MONGODB-URI  MongoDB connection string. Default value read from
                             environment variable PBM_MONGODB_URI.

Commands:
  help [<command>...]
    Show help.

  config [<flags>]
    Set, change or list the config

  backup
    Make backup
```

```
  restore <backup_name>
    Restore backup

  delete-backup <backup_name> [<flags>]
    Delete backup(s)

  cancel-backup
    Cancel backup

  list [<flags>]
    Backup list

  version [<flags>]
    PBM version info
```

Part II

Architecture

Percona Backup for MongoDB uses one **pbm-agent** process per `mongod` node. The PBM control collections in the MongoDB cluster or non-sharded replicaset itself serve as the central configuration, authentication and coordination channel. Administrators observe and control the backups or restores with a `pbm` CLI command that they can run from any host with the access to the MongoDB cluster.

A single **pbm-agent** is only involved with one cluster (or non-sharded replica set). The `pbm` CLI tool 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.

CHAPTER

ONE

PBM-AGENT

Percona Backup for MongoDB requires one instance of **pbm-agent** to be attached locally to each `mongod` instance. This includes replicaset nodes that are currently secondaries and config server replicaset nodes in a cluster.

There is no **pbm-agent** config file. Some configuration is required for the service script (e.g. `systemd` unit file) that will run it though. See *Configuring service init scripts*.

The **pbm-agent**'s backup and restore operations are triggered when it observes updates made to the PBM control collections by the `pbm` command line utility. In a method similar to the way replica set members elect a new primary the **pbm-agent** processes in the same replica set 'elect' one to do the backup or restore for that replica set.

CHAPTER

TWO

PBM COMMAND LINE UTILITY (PBM)

pbm is the command you will use manually in the shell, and it will also work as a command that can be executed in scripts (for example, by crond). It manages your backups through a set of sub-commands:

```
$ pbm help
usage: pbm [<flags>] <command> [<args> ...]

Percona Backup for MongoDB

Flags:
  --help                     Show context-sensitive help (also try --help-long
                             and --help-man).
  --mongodb-uri=MONGODB-URI  MongoDB connection string. Default value read from
                             environment variable PBM_MONGODB_URI.

Commands:
  help [<command>...]
    Show help.

  config [<flags>]
    Set, change or list the config

  backup
    Make backup

  restore <backup_name>
    Restore backup

  delete-backup <backup_name> [<flags>]
    Delete backup(s)

  cancel-backup
    Cancel backup

  list [<flags>]
    Backup list

  version [<flags>]
    PBM version info
```

pbm modifies the PBM config by saving it in the PBM control collection for config values. Likewise it starts and monitors backup or restore operations by updating and reading other PBM control collections for operations, log, etc.

pbm does not have its own config and/or cache files per se. 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.)

CHAPTER

THREE

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 of the config server replica set to keep them cleanly separated from user db namespaces. (In a non-sharded replicaset the `admin` db of the replica set itself is used.)

- *admin.pbmConfig*
- *admin.pbmCmd* (Used to define and trigger operations)
- *admin.pbmLock* (**`pbm-agent`** synchronization-lock structure)
- *admin.pbmBackup* (Log / status of each backup)

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 PBM for executing backups or restores. (See config page later.)

CHAPTER

FOUR

REMOTE BACKUP STORAGE

Conceptually or actually PBM 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 replicaset in the backup:

- 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.

Part III

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.

For your convenience, we recommend that you install the `percona-release` utility which makes it easy to install any Percona product on your system.

You may also build and install Percona Backup for MongoDB from source code in case you require a fully controlled installation method.

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

You should install **`pbm-agent`** on every server that has mongod nodes in the MongoDB cluster (or non-sharded replica set). The `pbm` CLI can be installed 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.

See also:

More information about `percona-release` https://www.percona.com/doc/percona-repo-config/percona-release.html

CHAPTER

FIVE

PREREQUISITES

It is recommended to install Percona Backup for MongoDB from official Percona repositories by using the `percona-release` utility.

```
$ percona-release enable tools
```

Percona Backup for MongoDB is available for installation from your package management system when you enable the *tools* repository.

See also:

Configuring Percona repositories https://www.percona.com/doc/percona-repo-config/index.html

CHAPTER

SIX

INSTALLING PERCONA BACKUP FOR MONGODB USING `APT`

Percona provides packages for the following systems that use the **`apt`** to interface the package management system:

- Debian 8 (“jessie”)
- Debian 9 (“stretch”)
- Debian 10 (“buster”)
- Ubuntu 16.04 LTS (Xenial Xerus)
- Ubuntu 18.04 LTS (Bionic Beaver)
- Ubuntu 20.04 LTS (Focal Fossa)

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

CHAPTER

SEVEN

INSTALLING PERCONA BACKUP FOR MONGODB USING `YUM`

Percona provides packages for the following systems that use the **`yum`** to interface the package management system:

- Red Hat Enterprise Linux / CentOS 6 (current stable release)
- Red Hat Enterprise Linux / CentOS 7 (current stable release)
- Red Hat Enterprise Linux / CentOS 8 (current stable release)

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

CHAPTER

EIGHT

BUILDING FROM SOURCE CODE

Building the project requires:

- Go 1.11 or above
- make

See also:

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

To build the project (from the project dir):

```
$ go get -d github.com/percona/percona-backup-mongodb
$ cd "$(go env GOPATH)/src/github.com/percona/percona-backup-mongodb"
$ make
```

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]
```

Percona Backup for MongoDB services and location of configuration files

After Percona Backup for MongoDB is successfully installed on your system, you have pbm and **pbm-agent** programs on your system.

CHAPTER

NINE

CONFIGURING SERVICE INIT SCRIPTS

The MongoDB connection URI string to the local mongod node should be set in the environment file that the *pbm-agent.service* systemd unit file includes.

With the current systemd unit file (see below), this means setting the "PBM_MONGODB_URI" environment variable in `/etc/default/pbm-agent` (for Debian and Ubuntu) or `/etc/sysconfig/pbm-agent` (for Red Hat or CentOS).

The *Running Percona Backup for MongoDB* section, explains in detail how to start **`pbm-agent`** and provides examples how to use `pbm` commands.

Hint: In Ubuntu and Debian `pbm-agent.service` is located in the `/lib/systemd/system/` directory. In Red Hat and CentOS, this file is found in `/usr/lib/systemd/system/pbm-agent.service`.

```
[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*

Part IV

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 config in a Cluster (or Non-sharded Replicaset)*.

CHAPTER

TEN

CREATE THE PBM USER

To run Percona Backup for MongoDB a user must be created in the `admin` db that has the role *grants* as shown below.

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

User name and password values and other options of the createUser command can be set as you require so long as the roles shown above are granted.

This user must be created on every replicaset, i.e. it must be created on the shard replicasets as well as the config server replicaset.

Note: In a cluster run *db.getSiblingDB("config").shards.find({}, {"host": true, "_id": false})* to list all the host+port lists for the shard replicasets. The replicaset 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).

CHAPTER

ELEVEN

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, replicaset, user, password, etc. as, say, 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.

The `pbm-agent` connection string

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

The `pbm` connection string

The `pbm` CLI should 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.

To make sure `pbm` always automatically connects to the current primary node use a replica set connection string. (A standalone-style connection string will only work successfully if it happens to be to the current primary.)

Part V

Percona Backup for MongoDB config in a Cluster (or Non-sharded Replicaset)

The config 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 replicaset), and can be read or updated using the `pbm` tool.

In short: 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.

As of v1.0 or v1.1 the config only contains the remote storage information.

CHAPTER

TWELVE

INSERT THE WHOLE PERCONA BACKUP FOR MONGODB CONFIG FROM A YAML FILE

If you are initializing a cluster or non-sharded replicaset 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.

```
$ pbm config --file pbm_config.yaml --mongodb-uri="mongodb://
↪pbmuser:secretpwd@mongocsvr1:27018,mongocsvr2:27018,mongocsvr3:27018/?
↪replicaSet=configrs"
```

Execute whilst connecting to config server replicaset if it is 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.)

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

CHAPTER

THIRTEEN

EXAMPLE CONFIG FILES

S3-compatible remote storage

Amazon Simple Storage Service

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

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 replicaset.

Warning: PBM uses the directory as if it was 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 replicaset can't access backup archive files in a normal local directory on another server.

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

Local Filesystem Storage

This cannot be used except if you have a single-node replicaset. (See 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 "Remote Filesystem Server Storage" above.

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
```

Part VI

Running Percona Backup for MongoDB

Please see *Authentication* if you have not already. This will explain the MongoDB user that needs to be created, and the connection method used by Percona Backup for MongoDB.

CHAPTER

FOURTEEN

INITIAL SETUP

1. Determine the right MongoDB connection string for the `pbm` CLI. (See *MongoDB connection strings - A Reminder (or Primer)*)
2. Use the `pbm` CLI to insert the config (especially the Remote Storage location and credentials information). See *Insert the whole Percona Backup for MongoDB Config from a YAML file*
3. Start (or restart) the **`pbm-agent`** processes for all mongod nodes.

Start the `pbm-agent` processes

After installing **`pbm-agent`** on the all the servers that have mongod nodes make sure one instance of it is started for each mongod node.

E.g. Imagine you put configsvr nodes (listen port 27019) colocated 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/").

It is best to use the packaged service scripts to run **`pbm-agent`**. After adding the database connection configuration for them (see :ref: pbm.installation.service_init_scripts) you can start the **`pbm-agent`** service as below:

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

For reference an example of starting pbm-agent manually is shown below. 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.

You can confirm the **`pbm-agent`** connected to its mongod and started OK by confirming *"pbm agent is listening for the commands"* is printed to the log file.

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.

CHAPTER

FIFTEEN

RUNNING PERCONA BACKUP FOR MONGODB

CHAPTER

SIXTEEN

RUNNING PBM COMMANDS

pbm is the command line utility to control the backup system.

Configuring a Remote Store for Backup and Restore Operations

This must be done once, at installation or re-installaton time, before backups can be listed, made, or restored. Please see *Percona Backup for MongoDB config in a Cluster (or Non-sharded Replicaset)*.

Listing all backups

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

Sample output

```
2019-09-10T07:04:14Z
2019-09-09T07:03:50Z
2019-09-08T07:04:21Z
2019-09-07T07:04:18Z
```

Starting a backup

```
$ pbm backup --mongodb-uri="mongodb://pbmuser:secretpwd@mongocsvr1:27018,
↪mongocsvr2:27018,mongocsvr3:27018/?replicaSet=configrs"
```

Starting a backup with compression

```
$ pbm backup --compression=s2  --mongodb-uri="mongodb://
↪pbmuser:secretpwd@mongocsvr1:27018,mongocsvr2:27018,mongocsvr3:27018/?
↪replicaSet=configrs"
```

s2 is the default compression type. Other supported compression types are: gzip, snappy, lz4, pgzip. The none value means no compression is done during backup.

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

Checking an in-progress backup

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.

Restoring a backup

To restore a backup that you have made using `pbm backup` you should use the `pbm restore` command supplying the timestamp of the backup that you intend to restore.

Important: Before running `pbm restore` on a cluster stop the balancer.

Important: Whilst the restore is running, clients should be stopped from accessing the database. The data will naturally be incomplete whilst the restore is in progress, and writes they make will cause the final restored data to differ from the backed-up data. In a cluster's restore the simplest way would be to shutdown all mongos nodes.

Important: Percona Backup for MongoDB is designed to be a full-database restore tool. As of version <=1.x it will perform 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 Percona Backup for MongoDB as of now (versions 1.x) replicates mongodump's behaviour 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 (i.e. all databases except "local", "config" and "admin") before running `pbm restore` if you want to guarantee the post-restore database only includes collections that are in the backup.

```
$ pbm restore 2019-06-09T07:03:50Z --mongodb-uri="mongodb://
↪pbmuser:secretpwd@mongocsvr1:27018,mongocsvr2:27018,mongocsvr3:27018/?
↪replicaSet=configrs"
```

After a cluster's restore is complete all mongos nodes will need to be restarted to reload the sharding metadata.

Cancelling a backup

You can cancel a running backup if, for example, you want to do another maintenance 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 cancelation has started
```

After the command execution, the backup is marked as cancelled in the `pbm list` output:

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

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).

To delete a backup, specify the `<backup_name>` from the the `pbm list` output as an argument.

```
#Get the backup name
$ pbm list
Backup history:
  2020-04-20T10:55:42Z
  2020-04-20T13:07:34Z
  2020-04-20T13:13:20Z
  2020-04-20T13:45:59Z

#Delete a backup
$ pbm delete-backup 2020-04-20T13:45:59Z
```

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 the `pbm delete-backup` command in the following format:

- `%Y-%M-%DT%H:%M:%S` (e.g. 2020-04-20T13:13:20) or
- `%Y-%M-%D` (e.g. 2020-04-20).

```
#Get the backup name
$ pbm list
Backup history:
  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 history:
  2020-04-21T02:16:33Z
```

Part VII

Troubleshooting Percona Backup for MongoDB

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

CHAPTER

SEVENTEEN

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.

The full set of commands:

```
$ /usr/bin/pbm-speed-test --help
usage: pbm-speed-test --mongodb-uri=MONGODB-URI [<flags>] <command> [<args> ...]

Percona Backup for MongoDB compression and upload speed 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>

Commands:
  help [<command>...]
    Show help.

  compression
    Run compression test

  storage
    Run storage test

  version [<flags>]
    PBM version info
```

By default, **pbm-speed-test** operates with fake semi random data documents. To run **pbm-speed-test** on a real collection, you require a connection to the MongoDB. See *MongoDB connection strings - A Reminder (or Primer)* for details.

Compression test

```
/usr/bin/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:

```
/usr/bin/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>
```

Upload speed test

```
/usr/bin/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:

```
/usr/bin/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>
```

CHAPTER

EIGHTEEN

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%)
```

Part VIII

Release notes

CHAPTER

NINETEEN

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).

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

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

Bugs Fixed

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

CHAPTER

TWENTY

PERCONA BACKUP FOR MONGODB 1.1.3

Date April 14, 2020

Installation *Installing Percona Backup for MongoDB*

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

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

CHAPTER

TWENTYONE

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*.

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"

CHAPTER

TWENTYTWO

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"
```

New Features

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

Improvements

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

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.

CHAPTER

TWENTYTHREE

PERCONA BACKUP FOR MONGODB 1.0.0

Percona is happy to announce the GA release of our latest software product Percona Backup for MongoDB 1.2 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.

CHAPTER

TWENTYFOUR

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.

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.

Part IX

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 X

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.

Contact Us

Bugs and feature requests can be opened as JIRA tickets at https://jira.percona.com/projects/PBM/issues .

Separate to using JIRA the email address for this project is (mongodb-backup@percona.com). Lastly you can also use the contact form on the site (https://www.percona.com/about-percona/contact).