PERCONA

Percona XtraBackup 8.0 Documentation

Release 8.0.27-19.0

Percona LLC and/or its affiliates

Feb 02, 2022

CONTENTS

Percona XtraBackup is an open-source hot backup utility for *MySQL* - based servers that doesn't lock your database during the backup.

Whether it is a 24x7 highly loaded server or a low-transaction-volume environment, *Percona XtraBackup* is designed to make backups a seamless procedure without disrupting the performance of the server in a production environment. Commercial support contracts are available.

Percona XtraBackup can back up data from *InnoDB*, *XtraDB*, *MyISAM*, and MyRocks tables on *MySQL* 8.0 servers as well as *Percona Server for MySQL* with *XtraDB*, *Percona Server for MySQL* 8.0, and *Percona XtraDB Cluster* 8.0.

Important: The support of the MyRocks storage engine was added in version 8.0.6.

Percona XtraBackup 8.0 does not support the TokuDB storage engine.

See also:

Percona TokuBackup https://www.percona.com/doc/percona-server/LATEST/tokudb/toku_backup.html

---

Percona XtraBackup 8.0 does not support making backups of databases created in versions prior to 8.0 of *MySQL*, *Percona Server for MySQL* or *Percona XtraDB Cluster*. As the changes that *MySQL* 8.0 introduced in *data dictionaries*, *redo log* and *undo log* are incompatible with previous versions, it is currently impossible for *Percona XtraBackup* 8.0 to also support versions prior to 8.0.

Due to changes in MySQL 8.0.20 released by Oracle at the end of April 2020, *Percona XtraBackup* 8.0, up to version 8.0.11, is not compatible with MySQL version 8.0.20 or higher, or Percona products that are based on it: Percona Server for MySQL and Percona XtraDB Cluster.

For more information, see Percona XtraBackup 8.x and MySQL 8.0.20

For a high-level overview of many of its advanced features, including a feature comparison, please see *About Percona XtraBackup*.

Part I

Introduction

CHAPTER

ONE

ABOUT PERCONA XTRABACKUP

Percona XtraBackup is the world's only open-source, free *MySQL* hot backup software that performs non-blocking backups for *InnoDB* and *XtraDB* databases. With *Percona XtraBackup*, you can achieve the following benefits:

- Backups that complete quickly and reliably
- Uninterrupted transaction processing during backups
- Savings on disk space and network bandwidth
- Automatic backup verification
- Higher uptime due to faster restore time

Percona XtraBackup makes *MySQL* hot backups for all versions of *Percona Server for MySQL*, and *MySQL*. It performs streaming, compressed, and incremental *MySQL* backups.

Important: With the introduction of *Percona XtraBackup* 8.0, *Percona XtraBackup* 2.4 will continue to support MySQL and Percona Server 5.6 and 5.7 databases. Due to the new MySQL redo log and data dictionary formats the Percona XtraBackup 8.0.x versions will only be compatible with MySQL 8.0.x and the upcoming Percona Server for MySQL 8.0.x

---

Percona's enterprise-grade commercial MySQL Support contracts include support for *Percona XtraBackup*. We recommend support for critical production deployments. Percona XtraDB Backup supports encryption.

Supported storage engines

Percona XtraBackup works with MySQL and Percona Server. It supports completely non-blocking backups of InnoDB, XtraDB, and MyRocks storage engines. Fast incremental backups are supported for Percona Server with the XtraDB changed page tracking enabled.

In addition, it can back up the following storage engines by briefly pausing writes at the end of the backup: MyISAM and *Merge*, including partitioned tables, triggers, and database options. InnoDB tables are still locked while copying non-InnoDB data.

Important: The support of the MyRocks storage engine was added in version 8.0.6.

Percona XtraBackup 8.0 does not support the TokuDB storage engine.

See also:

Percona TokuBackup https://www.percona.com/doc/percona-server/LATEST/tokudb/toku_backup.html

---

1.1 What are the features of Percona XtraBackup?

Here is a short list of the Percona XtraBackup features. See the documentation for more.

- Create hot InnoDB backups without pausing your database
- Make incremental backups of MySQL
- Stream compressed MySQL backups to another server
- Move tables between MySQL servers on-line
- Create new MySQL replication replicas easily
- Backup MySQL without adding load to the server
- Percona XtraBackup performs throttling based on the number of IO operations per second
- Percona XtraBackup skips secondary index pages and recreates them when a compact backup is prepared
- Percona XtraBackup can export individual tables even from a full backup, regardless of the InnoDB version
- Backup locks is a lightweight alternative to `FLUSH TABLES WITH READ LOCK` available in Percona Server. Percona XtraBackup uses them automatically to copy non-InnoDB data to avoid blocking DML queries that modify InnoDB tables.

See also:

For more information see *How Percona XtraBackup Works*

Additional information

InnoDB tables are still locked while copying non-*InnoDB* data.

CHAPTER

TWO

HOW *PERCONA XTRABACKUP* WORKS

Percona XtraBackup is based on *InnoDB*'s crash-recovery functionality. It copies your *InnoDB* data files, which results in data that is internally inconsistent; but then it performs crash recovery on the files to make them a consistent, usable database again.

This works because *InnoDB* maintains a redo log, also called the transaction log. This contains a record of every change to InnoDB data. When *InnoDB* starts, it inspects the data files and the transaction log, and performs two steps. It applies committed transaction log entries to the data files, and it performs an undo operation on any transactions that modified data but did not commit.

Percona XtraBackup works by remembering the log sequence number (*LSN*) when it starts, and then copying away the data files. It takes some time to do this, so if the files are changing, then they reflect the state of the database at different points in time. At the same time, *Percona XtraBackup* runs a background process that watches the transaction log files, and copies changes from it. *Percona XtraBackup* needs to do this continually because the transaction logs are written in a round-robin fashion, and can be reused after a while. *Percona XtraBackup* needs the transaction log records for every change to the data files since it began execution.

Percona XtraBackup uses Backup locks where available as a lightweight alternative to `FLUSH TABLES WITH READ LOCK`. This feature is available in *Percona Server for MySQL* 5.6+. *MySQL* 8.0 allows acquiring an instance level backup lock via the `LOCK INSTANCE FOR BACKUP` statement.

Locking is only done for *MyISAM* and other non-InnoDB tables **after** *Percona XtraBackup* finishes backing up all InnoDB/XtraDB data and logs. *Percona XtraBackup* uses this automatically to copy non-InnoDB data to avoid blocking DML queries that modify *InnoDB* tables.

Important: To use effectively either `LOCK INSTANCE FOR BACKUP` or `LOCK TABLES FOR BACKUP`, the `BACKUP_ADMIN` privilege is needed in order to query `performance_schema.log_status`.

---

xtrabackup tries to avoid backup locks and `FLUSH TABLES WITH READ LOCK` when the instance contains only InnoDB tables. In this case, **xtrabackup** obtains binary log coordinates from `performance_schema.log_status`. `FLUSH TABLES WITH READ LOCK` is still required in MySQL 8.0 when xtrabackup is started with the *`--slave-info`*. The `log_status` table in *Percona Server for MySQL* 8.0 is extended to include the relay log coordinates, so no locks are needed even with the *`--slave-info`* option.

See also:

***MySQL* Documentation: More information about `LOCK INSTANCE FOR BACKUP`** https://dev.mysql.com/doc/refman/8.0/en/lock-instance-for-backup.html

When backup locks are supported by the server, **xtrabackup** first copies *InnoDB* data, runs the `LOCK TABLES FOR BACKUP` and then copies the *MyISAM* tables. Once this is done, the backup of the files will begin. It will backup *.frm*, *.MRG*, *.MYD*, *.MYI*, .ARM, .ARZ, *.CSM*, *.CSV*, `.sdi` and `.par` files.

After that **xtrabackup** will use `LOCK BINLOG FOR BACKUP` to block all operations that might change either binary log position or `Exec_Master_Log_Pos` or `Exec_Gtid_Set` (i.e. source binary log coordinates corre-

sponding to the current SQL thread state on a replication replica) as reported by `SHOW MASTER/SLAVE STATUS`. **xtrabackup** will then finish copying the REDO log files and fetch the binary log coordinates. After this is completed **xtrabackup** will unlock the binary log and tables.

Finally, the binary log position will be printed to `STDERR` and **xtrabackup** will exit returning 0 if all went OK.

Note that the `STDERR` of **xtrabackup** is not written in any file. You will have to redirect it to a file, e.g., `xtrabackup OPTIONS 2> backupout.log`.

It will also create the *following files* in the directory of the backup.

During the prepare phase, *Percona XtraBackup* performs crash recovery against the copied data files, using the copied transaction log file. After this is done, the database is ready to restore and use.

The backed-up *MyISAM* and *InnoDB* tables will be eventually consistent with each other, because after the prepare (recovery) process, *InnoDB*'s data is rolled forward to the point at which the backup completed, not rolled back to the point at which it started. This point in time matches where the `FLUSH TABLES WITH READ LOCK` was taken, so the *MyISAM* data and the prepared *InnoDB* data are in sync.

The **xtrabackup** offers many features not mentioned in the preceding explanation. The functionality of each tool is explained in more detail further in this manual. In brief, though, the tools enable you to do operations such as streaming and incremental backups with various combinations of copying the data files, copying the log files, and applying the logs to the data.

2.1 Restoring a backup

To restore a backup with **xtrabackup** you can use the *--copy-back* or *--move-back* options.

xtrabackup will read from the `my.cnf` the variables *datadir*, *innodb_data_home_dir*, *innodb_data_file_path*, *innodb_log_group_home_dir* and check that the directories exist.

It will copy the *MyISAM* tables, indexes, etc. (*.MRG*, *.MYD*, *.MYI*, .ARM, .ARZ, *.CSM*, *.CSV*, `.sdi`, and `par` files) first, *InnoDB* tables and indexes next and the log files at last. It will preserve file's attributes when copying them, you may have to change the files' ownership to `mysql` before starting the database server, as they will be owned by the user who created the backup.

Alternatively, the *--move-back* option may be used to restore a backup. This option is similar to *--copy-back* with the only difference that instead of copying files it moves them to their target locations. As this option removes backup files, it must be used with caution. It is useful in cases when there is not enough free disk space to hold both data files and their backup copies.

Part II

Installation

CHAPTER

THREE

INSTALLING *PERCONA XTRABACKUP* 8.0

This page provides the information on how to install *Percona XtraBackup*. Following options are available:

- *Installing Percona XtraBackup from Repositories* (recommended)
- *Installing Percona XtraBackup on Debian and Ubuntu*
- *Installing Percona XtraBackup on Red Hat Enterprise Linux and CentOS*
- *Installing Percona XtraBackup using downloaded rpm packages*
- *Installing Percona XtraBackup using downloaded deb packages*
- *Installing Percona XtraBackup from a Binary Tarball*
- *Compiling and Installing from Source Code*
- pxc.installing/docker.running

Before installing, you might want to read the *Percona XtraBackup 8.0 Release Notes*.

3.1 Installing *Percona XtraBackup* from Repositories

Percona provides repositories for **yum** (RPM packages for *Red Hat*, *CentOS* and *Amazon Linux AMI*) and **apt** (.deb packages for *Ubuntu* and *Debian*) for software such as *Percona Server for MySQL*, *Percona XtraBackup*, and *Percona Toolkit*. This makes it easy to install and update your software and its dependencies through your operating system's package manager. This is the recommended way of installing *Percona XtraBackup*.

The following guides describe the installation process for using the official Percona repositories for .deb and .rpm packages.

Note: For experimental migrations from earlier database server versions, you will need to backup and restore using XtraBackup 2.4 and then use mysql_upgrade from MySQL 8.0.x

See also:

Supported MySQL and Percona Server for MySQL versions in *Percona XtraBackup* 2.4 and 8.0 *About Percona XtraBackup*

3.2 Installing *Percona XtraBackup* from a Binary Tarball

Binary tarballs are available for download and installation. Select the *Percona XtraBackup* 8.0 version, the software or the operating system, and the type of tarball for your installation.

The following table lists the tarball types available in `Linux - Generic`. Both types support all distributions.

Type	Name	Description
Full	percona-xtrabackup-<version number>-Linux.x86_64.glibc2.12.tar.gz	Contains binaries, libraries, test files, and debug symbols
Minimal	percona-xtrabackup-<version number>-Linux.x86_64.glibc2.12-minimal.tar.gz	Contains binaries, and libraries but does not include test files, or debug symbols

Selecting a different software, such as Ubuntu 20.04 (Focal Fossa), provides a tarball for that operating system. You can download the packages together or separately.

The following link is an example of downloading the full tarball for Linux/Generic:

```
$ wget https://downloads.percona.com/downloads/Percona-XtraBackup-LATEST/Percona-
↪XtraBackup-8.0.23-16/binary/tarball/percona-xtrabackup-8.0.23-16-Linux-x86_64.
↪glibc2.17.tar.gz
```

3.3 Compiling and Installing from Source Code

Percona XtraBackup is open source and the code is available on Github. The following guide describes the compiling and installation process from source code.

3.3.1 Compiling and Installing from Source Code

The source code is available from the *Percona XtraBackup Github* project. The easiest way to get the code is by using the **`git clone`** command. Then, switch to the release branch that you want to install, such as **8.0**.

```
$ git clone https://github.com/percona/percona-xtrabackup.git
$ cd percona-xtrabackup
$ git checkout 8.0
```

Step 1: Installing prerequisites

The following packages and tools must be installed to compile *Percona XtraBackup* from source. These might vary from system to system.

Important: In order to build *Percona XtraBackup* v8.0 from source, you must use *cmake* version 3. In your distribution, it may be available either as a separate package `cmake3` or as `cmake`. To check which version is installed, run `cmake --version` and if it does report a version 3, install `cmake3` for your system.

See also:

https://cmake.org/

Debian or Ubuntu using `apt`

```
sudo apt install bison pkg-config cmake devscripts debconf \
debhelper automake bison ca-certificates \
libcurl4-openssl-dev cmake debhelper libaio-dev \
libncurses-devlibssl-dev libtool libz-dev libgcrypt-dev libev-dev \
lsb-release python-docutils build-essential rsync \
libdbd-mysql-perl libnuma1 socat librtmp-dev libtinfo5 \
qpress liblz4-tool liblz4-1 liblz4-dev vim-common
```

To be able to install the man pages, install the `python3-sphinx` package:

```
$ sudo apt install python3-sphinx
```

CentOS or Red Hat using `yum`

Percona XtraBackup requires GCC version 5.3 or higher. If the version of GCC installed on your system is lower then you may need to install and enable the Developer Toolset 7 on `RPM`-based distributions to make sure that you use the latest GCC compiler and development tools. Then, install `cmake` and other dependencies:

```
$ sudo yum install cmake openssl-devel libaio libaio-devel automake autoconf \
bison libtool ncurses-devel libgcrypt-devel libev-devel libcurl-devel zlib-devel \
vim-common
```

To be able to install the man pages, install the `python3-sphinx` package:

```
$ sudo yum install python3-sphinx
```

Step 2: Generating the build pipeline

At this step, you have `cmake` run the commands in the `CMakeList.txt` file to generate the build pipeline, i.e. a native build environment that will be used to compile the source code).

1. Change to the directory where you cloned the `percona-xtrabackup` repository

   ```
   $ cd percona-xtrabackup
   ```

2. Create a directory to store the compiled files and then change to that directory:

   ```
   $ mkdir build
   $ cd build
   ```

3. Run *cmake* or *cmake3*. In either case, the options you need to use are the same.

Note: You can build *Percona XtraBackup* with man pages but this requires `python-sphinx` package which isn't available from that main repositories for every distribution. If you installed the `python-sphinx` package you need to remove the `-DWITH_MAN_PAGES=OFF` from previous command.

```
$ cmake -DWITH_BOOST=PATH-TO-BOOST-LIBRARY -DDOWNLOAD_BOOST=ON \
-DBUILD_CONFIG=xtrabackup_release -DWITH_MAN_PAGES=OFF -B ..
```

More information about parameters

-DWITH_BOOST For the `-DWITH_BOOST` parameter, specify the name of a directory to download the boost library to. This directory will be created automatically in your current directory.

-B (–build) *Percona XtraBackup* is configured to forbid generating the build pipeline for `make` in the same directory where you store your sources. The `-B` parameter refers to the directory that contains the source code. In this example we use the relative path to the parent directory (..).

Important: CMake Error at CMakeLists.txt:367 (MESSAGE): Please do not build in-source. Out-of source builds are highly recommended: you can have multiple builds for the same source, and there is an easy way to do cleanup, simply remove the build directory (note that 'make clean' or 'make distclean' does *not* work)

You *can* force in-source build by invoking cmake with -DFORCE_INSOURCE_BUILD=1

-DWITH_MAN_PAGES To build *Percona XtraBackup* man pages, use `ON` or remove this parameter from the command line (it is `ON` by default).

To be able to install the man pages, install the `python3-sphinx` package.

See also:

Step 1: Installing prerequisites

Step 2: Compiling the source code

To compile the source code in your `build` directory, use the `make` command.

Important: The computer where you intend to compile *Percona XtraBackup* 8.0 must have at least 2G of RAM available.

---

1. Change to the `build` directory (created at *Step 2: Generating the build pipeline*).
2. Run the `make` command. This command may take a long time to complete.

```
$ make
```

Step 3: Installing on the target system

The following command installs all *Percona XtraBackup* binaries **`xtrabackup`** and tests to default location on the target system: `/usr/local/xtrabackup`.

Run `make install` to install *Percona XtraBackup* to the default location.

```
$ sudo make install
```

Installing to a non-default location

You may use the *DESTDIR* parameter with `make install` to install *Percona XtraBackup* to another location. Make sure that the effective user is able to write to the destination you choose.

```
$ sudo make DESTDIR=<DIR_NAME> install
```

In fact, the destination directory is determined by the installation layout (`-DINSTALL_LAYOUT`) that `cmake` applies (see *Step 2: Generating the build pipeline*). In addition to the installation directory, this parameter controls a number of other destinations that you can adjust for your system.

By default, this parameter is set to `STANDALONE`, which implies the installation directory to be `/usr/local/xtrabackup`.

See also:

MySQL Documentation: -DINSTALL_LAYOUT

Step 4: Running

After *Percona XtraBackup* is installed on your system, you may run it by using the full path to the `xtrabackup` command:

```
$ /usr/local/xtrabackup/bin/xtrabackup
```

Update your PATH environment variable if you would like to use the command on the command line directly.

```
$# Setting $PATH on the command line
$ PATH=$PATH:/usr/local/xtrabackup/bin/xtrabackup

$# Run xtrabackup directly
$ xtrabackup
```

Alternatively, you may consider placing a soft link (using `ln -s`) to one of the locations listed in your `PATH` environment variable.

See also:

`man ln`

To view the documentation with `man`, update the `MANPATH` variable.

3.4 Running *Percona XtraBackup* in a Docker container

Docker images of *Percona XtraBackup* 8.0 are hosted publicly on Docker Hub at https://hub.docker.com/r/percona/percona-xtradb-cluster/.

For more information about how to use Docker, see the Docker Docs.

Note: Make sure that you are using the latest version of Docker. The ones provided by the operating system package manager may be outdated and may cause errors.

3.4.1 Running Percona XtraBackup in a *Docker* container

You may run Percona XtraBackup in a *Docker* container without having to install it. All required libraries come already installed in the container.

Being a lightweight execution environment, *Docker* containers enable creating configurations where each program runs in a separate container. You may run Percona Server for MySQL in one container and Percona XtraBackup in another.

You create a new *Docker* container based on a *Docker* image, which works as a template for newly created containers. *Docker* images for Percona XtraBackup are hosted publicly on Docker Hub here.

```
$ sudo docker create ... percona/percona-xtrabackup --name xtrabackup ...
```

Scope of this section

Docker containers offer a range of different options effectively allowing to create quite complex setup. This section demonstrates how to backup data on a Percona Server for MySQL running in another *Docker* container.

Installing *Docker*

Your operating system may already provide a package for **`docker`**. However, the versions of *Docker* provided by your operating system are likely to be outdated.

Use the installation instructions for your operating system available from the *Docker* site to set up the latest version of **`docker`**.

See also:

***Docker* Documentation:**

- How to use Docker
- Installing
- Getting started

Connecting to a Percona Server for MySQL container

Percona XtraBackup works in combination with a database server. When running a *Docker* container for Percona XtraBackup, you can make backups for a database server either installed on the host machine or running in a separate *Docker* container.

To set up a database server on a host machine or in *Docker* container, follow the documentation of the supported product that you intend to use with Percona XtraBackup.

See also:

Percona Server for MySQL Documentation:

- Installing on a host machine
- Running in a Docker container

```
$ sudo docker run -d --name percona-server-mysql \
-e MYSQL_ROOT_PASSWORD=root percona/percona-server:8.0
```

As soon as Percona Server for MySQL runs, add some data to it. Now, you are ready to make backups with Percona XtraBackup.

Creating a *Docker* container from Percona XtraBackup image

You can create a *Docker* container based on Percona XtraBackup image with either `docker create` or the `docker run` command. `docker create` creates a *Docker* container and makes it available for starting later.

Docker downloads the Percona XtraBackup image from the Docker Hub. If it is not the first time you use the selected image, *Docker* uses the image available locally.

```
$ sudo docker create --name percona-xtrabackup --volumes-from percona-server-mysql \
percona/percona-xtrabackup  \
xtrabackup --backup --datadir=/var/lib/mysql/ --target-dir=/backup \
--user=root --password=mysql
```

With parameter name you give a meaningful name to your new *Docker* container so that you could easily locate it among your other containers.

The `volumes-from` flag refers to Percona Server for MySQL and indicates that you indend to use the same data as the Percona Server for MySQL container.

Run the container with exactly the same parameters that were used when the container was created:

```
$ sudo docker start -ai percona-xtrabackup
```

This command starts the *percona-xtrabackup* container, attaches to its input/output streams, and opens an interactive shell.

The `docker run` is a shortcut command that creates a *Docker* container and then immediately runs it.

```
$ sudo docker run --name percona-xtrabackup --volumes-from percona-server-mysql \
percona/percona-xtrabackup
xtrabackup --backup --data-dir=/var/lib/mysql --target-dir=/backup --user=root --
↪password=mysql
```

See also:

More in *Docker* documentation

- Docker volumes as persistent data storage for containers
- More information about containers

CHAPTER

FOUR

INSTALLING *PERCONA XTRABACKUP* ON *DEBIAN* AND *UBUNTU*

Ready-to-use packages are available from the *Percona XtraBackup* software repositories and the download page.

Specific information on the supported platforms, products, and versions is described in Percona Software and Platform Lifecycle.

4.1 What's in each DEB package?

The `percona-xtrabackup-80` package contains the latest *Percona XtraBackup* GA binaries and associated files.

The `percona-xtrabackup-dbg-80` package contains the debug symbols for binaries in `percona-xtrabackup-80`.

The `percona-xtrabackup-test-80` package contains the test suite for *Percona XtraBackup*.

The `percona-xtrabackup` package contains the older version of the *Percona XtraBackup*.

4.2 Installing *Percona XtraBackup* via `percona-release`

Percona XtraBackup, like many other *Percona* products, is installed via the `percona-release` package configuration tool.

1. Download a deb package for `percona-release` the repository packages from Percona web:

   ```
   $ wget https://repo.percona.com/apt/percona-release_latest.$(lsb_release -sc)_all.
   ↪deb
   ```

2. Install the downloaded package with **`dpkg`**. To do that, run the following commands as root or with **`sudo`**: `dpkg -i percona-release_latest.$(lsb_release -sc)_all.deb`

   Once you install this package the Percona repositories should be added. You can check the repository setup in the `/etc/apt/sources.list.d/percona-release.list` file.

3. Enable the repository: `percona-release enable-only tools release`

   If *Percona XtraBackup* is intented to be used in combination with the upstream MySQL Server, you only need to enable the `tools` repository: `percona-release enable-only tools`.

4. Remember to update the local cache: `apt update`

5. After that you can install the `percona-xtrabackup-80` package:

   ```
   $ sudo apt install percona-xtrabackup-24
   ```

6. In order to make compressed backups, install the `qpress` package:

```
$ sudo apt install qpress
```

See also:

Compressed Backup

4.3 Apt-Pinning the packages

In some cases you might need to "pin" the selected packages to avoid the upgrades from the distribution repositories. You'll need to make a new file `/etc/apt/preferences.d/00percona.pref` and add the following lines in it:

```
Package: *
Pin: release o=Percona Development Team
Pin-Priority: 1001
```

For more information about the pinning you can check the official debian wiki.

4.4 Installing *Percona XtraBackup* using downloaded deb packages

Download the packages of the desired series for your architecture from the download page. The following example will download *Percona XtraBackup* 8.0.4-1 release package for *Debian* 8.0:

```
$ wget https://www.percona.com/downloads/XtraBackup/Percona-XtraBackup-8.0.4/binary/
↪debian/stretch/x86_64/percona-xtrabackup-80_8.0.4-1.stretch_amd64.deb
```

Now you can install *Percona XtraBackup* by running:

```
$ sudo dpkg -i percona-xtrabackup-80_0.4-1.stretch_amd64.deb
```

Note: When installing packages manually like this, you'll need to make sure to resolve all the dependencies and install missing packages yourself.

4.5 Uninstalling *Percona XtraBackup*

To uninstall *Percona XtraBackup* you'll need to remove all the installed packages.

1. Remove the packages

```
$ sudo apt remove percona-xtrabackup-24
```

CHAPTER

FIVE

INSTALLING *PERCONA XTRABACKUP* ON *RED HAT ENTERPRISE LINUX* AND *CENTOS*

Ready-to-use packages are available from the *Percona XtraBackup* software repositories and the download page. The *Percona* **yum** repository supports popular *RPM*-based operating systems, including the *Amazon Linux AMI*.

The easiest way to install the *Percona Yum* repository is to install an *RPM* that configures **yum** and installs the Percona GPG key.

Specific information on the supported platforms, products, and versions is described in Percona Software and Platform Lifecycle.

5.1 What's in each RPM package?

The `percona-xtrabackup-80` package contains the latest *Percona XtraBackup* GA binaries and associated files.

Package	Contains
`percona-xtrabackup-80-debuginfo`	The debug symbols for binaries in `percona-xtrabackup-80`
`percona-xtrabackup-test-80`	The test suite for *Percona XtraBackup*
`percona-xtrabackup`	The older version of the *Percona XtraBackup*

5.2 Installing *Percona XtraBackup* from Percona `yum` repository

1. Install the Percona yum repository by running the following command as the `root` user or with **sudo**: `yum install https://repo.percona.com/yum/percona-release-latest.noarch.rpm`
2. Enable the repository: `percona-release enable-only tools release`

   If *Percona XtraBackup* is intented to be used in combination with the upstream MySQL Server, you only need to enable the `tools` repository: `percona-release enable-only tools`.
3. Install *Percona XtraBackup* by running: `yum install percona-xtrabackup-80`

Warning: Make sure that you have the `libev` package installed before installing *Percona XtraBackup* on CentOS 6. For this operating system, the `libev` package is available from the EPEL repositories.

1. To be able to make compressed backups, install the `qpress` package:

   $ yum install qpress

See also:

Compressed Backup

5.3 Installing *Percona XtraBackup* using downloaded rpm packages

Download the packages of the desired series for your architecture from the download page. The following example downloads *Percona XtraBackup* 8.0.4 release package for *CentOS* 7:

```
$ wget https://www.percona.com/downloads/XtraBackup/Percona-XtraBackup-8.0.4/binary/
↪redhat/7/x86_64/percona-xtrabackup-80-8.0.4-1.el7.x86_64.rpm
```

Now you can install *Percona XtraBackup* by running `yum localinstall`:

```
$ yum localinstall percona-xtrabackup-80-8.0.4-1.el7.x86_64.rpm
```

Note: When installing packages manually like this, you'll need to make sure to resolve all the dependencies and install missing packages yourself.

5.4 Uninstalling *Percona XtraBackup*

To completely uninstall *Percona XtraBackup* you'll need to remove all the installed packages: `yum remove percona-xtrabackup`

Part III

Prerequisites

CHAPTER

SIX

CONNECTION AND PRIVILEGES NEEDED

Percona XtraBackup needs to be able to connect to the database server and perform operations on the server and the *datadir* when creating a backup, when preparing in some scenarios and when restoring it. In order to do so, there are privileges and permission requirements on its execution that must be fulfilled.

Privileges refers to the operations that a system user is permitted to do in the database server. **They are set at the database server and only apply to users in the database server**.

Permissions are those which permits a user to perform operations on the system, like reading, writing or executing on a certain directory or start/stop a system service. **They are set at a system level and only apply to system users**.

When **xtrabackup** is used, there are two actors involved: the user invoking the program - *a system user* - and the user performing action in the database server - *a database user*. Note that these are different users in different places, even though they may have the same username.

All the invocations of **xtrabackup** in this documentation assume that the system user has the appropriate permissions and you are providing the relevant options for connecting the database server - besides the options for the action to be performed - and the database user has adequate privileges.

6.1 Connecting to the server

The database user used to connect to the server and its password are specified by the `--user` and `--password` option:

```
$ xtrabackup --user=DVADER --password=14MY0URF4TH3R --backup \
--target-dir=/data/bkps/
```

If you don't use the `--user` option, *Percona XtraBackup* will assume the database user whose name is the system user executing it.

6.1.1 Other Connection Options

According to your system, you may need to specify one or more of the following options to connect to the server:

Option	Description
–port	The port to use when connecting to the database server with TCP/IP.
–socket	The socket to use when connecting to the local database.
–host	The host to use when connecting to the database server with TCP/IP.

These options are passed to the **mysql** child process without alteration, see `mysql --help` for details.

Note: In case of multiple server instances, the correct connection parameters (port, socket, host) must be specified in order for **xtrabackup** to talk to the correct server.

6.2 Permissions and Privileges Needed

Once connected to the server, in order to perform a backup you will need `READ` and `EXECUTE` permissions at a filesystem level in the server's *datadir*.

The database user needs the following privileges on the tables or databases to be backed up:

- `RELOAD` and `LOCK TABLES` (unless the *--no-lock* option is specified) in order to run `FLUSH TABLES WITH READ LOCK` and `FLUSH ENGINE LOGS` prior to start copying the files, and requires this privilege when Backup Locks are used
- `BACKUP_ADMIN` privilege is needed to query the performance_schema.log_status table, and run `LOCK INSTANCE FOR BACKUP`, `LOCK BINLOG FOR BACKUP`, or `LOCK TABLES FOR BACKUP`.
- `REPLICATION CLIENT` in order to obtain the binary log position,
- `CREATE TABLESPACE` in order to import tables (see *Restoring Individual Tables*),
- `PROCESS` in order to run `SHOW ENGINE INNODB STATUS` (which is mandatory), and optionally to see all threads which are running on the server (see *Handling FLUSH TABLES WITH READ LOCK*),
- `SUPER` in order to start/stop the replication threads in a replication environment, use XtraDB Changed Page Tracking for *Incremental Backups* and for *handling FLUSH TABLES WITH READ LOCK*,
- `CREATE` privilege in order to create the *PERCONA_SCHEMA.xtrabackup_history* database and table,
- `ALTER` privilege in order to upgrade the *PERCONA_SCHEMA.xtrabackup_history* database and table,
- `INSERT` privilege in order to add history records to the *PERCONA_SCHEMA.xtrabackup_history* table,
- `SELECT` privilege in order to use *--incremental-history-name* or *--incremental-history-uuid* in order for the feature to look up the `innodb_to_lsn` values in the *PERCONA_SCHEMA.xtrabackup_history* table.
- `SELECT` privilege on the keyring_component_status table to view the attributes and status of the installed keyring component when in use.

The explanation of when these are used can be found in *How Percona XtraBackup Works*.

An SQL example of creating a database user with the minimum privileges required to full backups would be:

```
mysql> CREATE USER 'bkpuser'@'localhost' IDENTIFIED BY 's3cr%T';
mysql> GRANT BACKUP_ADMIN, PROCESS, RELOAD, LOCK TABLES, REPLICATION CLIENT ON *.* TO
↪'bkpuser'@'localhost';
mysql> GRANT SELECT ON performance_schema.log_status TO 'bkpuser'@'localhost';
mysql> GRANT SELECT ON performance_schema.keyring_component_status TO bkpuser@
↪'localhost'
mysql> FLUSH PRIVILEGES;
```

CHAPTER

SEVEN

CONFIGURING XTRABACKUP

All of the **xtrabackup** configuration is done through options, which behave exactly like standard *MySQL* program options: they can be specified either at the command-line, or through a file such as `/etc/my.cnf`.

The **xtrabackup** binary reads the `[mysqld]` and `[xtrabackup]` sections from any configuration files, in that order. That is so that it can read its options from your existing *MySQL* installation, such as the *datadir* or some of the *InnoDB* options. If you want to override these, just specify them in the `[xtrabackup]` section, and because it is read later, it will take precedence.

You don't need to put any configuration in your `my.cnf` if you don't want to. You can simply specify the options on the command-line. Normally, the only thing you might find convenient to place in the `[xtrabackup]` section of your `my.cnf` file is the `target_dir` option to default the directory in which the backups will be placed, for example:

```
[xtrabackup]
target_dir = /data/backups/mysql/
```

This manual will assume that you do not have any file-based configuration for **xtrabackup**, so it will always show command-line options being used explicitly. Please see the *option and variable reference* for details on all of the configuration options.

The **xtrabackup** binary does not accept exactly the same syntax in the `my.cnf` file as the **mysqld** server binary does. For historical reasons, the **mysqld** server binary accepts parameters with a `--set-variable=<variable>=<value>` syntax, which **xtrabackup** does not understand. If your `my.cnf` file has such configuration directives, you should rewrite them in the `--variable=value` syntax.

7.1 System Configuration and NFS Volumes

The **xtrabackup** tool requires no special configuration on most systems. However, the storage where the *`--target-dir`* is located must behave properly when `fsync()` is called. In particular, we have noticed that NFS volumes not mounted with the `sync` option might not really sync the data. As a result, if you back up to an NFS volume mounted with the async option, and then try to prepare the backup from a different server that also mounts that volume, the data might appear to be corrupt. You can use the `sync` mount option to avoid this problem.

CHAPTER

EIGHT

SERVER VERSION AND BACKUP VERSION COMPARISON

Percona XtraBackup 8.0.21 adds the `--no-server-version-check` parameter. This parameter compares the source system version to the *Percona XtraBackup* version.

The parameter checks for the following scenarios:

- The source system and the PXB version are the same, the backup proceeds
- The source system is less than the PXB version, the backup proceeds
- The source system is greater than the PXB version, and the parameter is not overridden, the backup is stopped and returns an error message
- The source system is greater than the PXB version, and the parameter is overridden, the backup proceeds

Explicitly adding the `--no-server-version-check` parameter, like the example, overrides the parameter and the backup proceeds.

```
$ xtrabackup --backup --no-server-version-check --target-dir=$mysql/backup1
```

When you override the parameter, the following events can happen:

- Backup fails
- Creates a corrupted backup
- Backup successful

Part IV

Backup Scenarios

CHAPTER

NINE

THE BACKUP CYCLE - FULL BACKUPS

9.1 Creating a backup

To create a backup, run **xtrabackup** with the *--backup* option. You also need to specify the *--target-dir* option, which is where the backup will be stored, if the *InnoDB* data or log files are not stored in the same directory, you might need to specify the location of those, too. If the target directory does not exist, **xtrabackup** creates it. If the directory does exist and is empty, **xtrabackup** will succeed.

xtrabackup will not overwrite existing files, it will fail with operating system error 17, `file exists`.

To start the backup process run:

```
$ xtrabackup --backup --target-dir=/data/backups/
```

This will store the backup at `/data/backups/`. If you specify a relative path, the target directory will be relative to the current directory.

During the backup process, you should see a lot of output showing the data files being copied, as well as the log file thread repeatedly scanning the log files and copying from it. Here is an example that shows the log thread scanning the log in the background, and a file copying thread working on the `ibdata1` file:

```
160906 10:19:17 Finished backing up non-InnoDB tables and files
160906 10:19:17 Executing FLUSH NO_WRITE_TO_BINLOG ENGINE LOGS...
xtrabackup: The latest check point (for incremental): '62988944'
xtrabackup: Stopping log copying thread.
.160906 10:19:18 >> log scanned up to (137343534)
160906 10:19:18 Executing UNLOCK TABLES
160906 10:19:18 All tables unlocked
160906 10:19:18 Backup created in directory '/data/backups/'
160906 10:19:18 [00] Writing backup-my.cnf
160906 10:19:18 [00]        ...done
160906 10:19:18 [00] Writing xtrabackup_info
160906 10:19:18 [00]        ...done
xtrabackup: Transaction log of lsn (26970807) to (137343534) was copied.
160906 10:19:18 completed OK!
```

The last thing you should see is something like the following, where the value of the `<LSN>` will be a number that depends on your system:

```
$ xtrabackup: Transaction log of lsn (<SLN>) to (<LSN>) was copied.
```

Note: Log copying thread checks the transactional log every second to see if there were any new log records written that need to be copied, but there is a chance that the log copying thread might not be able to keep up with the amount

of writes that go to the transactional logs, and will hit an error when the log records are overwritten before they could be read.

After the backup is finished, the target directory will contain files such as the following, assuming you have a single InnoDB table `test.tbl1` and you are using MySQL's *innodb_file_per_table* option:

```
$ ls -lh /data/backups/
total 182M
drwx------  7 root root 4.0K Sep  6 10:19 .
drwxrwxrwt 11 root root 4.0K Sep  6 11:05 ..
-rw-r-----  1 root root  387 Sep  6 10:19 backup-my.cnf
-rw-r-----  1 root root  76M Sep  6 10:19 ibdata1
drwx------  2 root root 4.0K Sep  6 10:19 mysql
drwx------  2 root root 4.0K Sep  6 10:19 performance_schema
drwx------  2 root root 4.0K Sep  6 10:19 sbtest
drwx------  2 root root 4.0K Sep  6 10:19 test
drwx------  2 root root 4.0K Sep  6 10:19 world2
-rw-r-----  1 root root  116 Sep  6 10:19 xtrabackup_checkpoints
-rw-r-----  1 root root  433 Sep  6 10:19 xtrabackup_info
-rw-r-----  1 root root 106M Sep  6 10:19 xtrabackup_logfile
```

The backup can take a long time, depending on how large the database is. It is safe to cancel at any time, because **xtrabackup** does not modify the database.

The next step is getting your backup ready to be restored.

9.2 Preparing a backup

After making a backup with the *--backup* option, you need need to prepare it in order to restore it. Data files are not point-in-time consistent until they are *prepared*, because they were copied at different times as the program ran, and they might have been changed while this was happening.

If you try to start InnoDB with these data files, it will detect corruption and stop working to avoid running on damaged data. The *--prepare* step makes the files perfectly consistent at a single instant in time, so you can run *InnoDB* on them.

You can run the *prepare* operation on any machine; it does not need to be on the originating server or the server to which you intend to restore. You can copy the backup to a utility server and prepare it there.

Note that *Percona XtraBackup* 8.0 can only prepare backups of *MySQL* 8.0, *Percona Server for MySQL* 8.0, and *Percona XtraDB Cluster* 8.0 databases. Releases prior to 8.0 are not supported.

During the *prepare* operation, **xtrabackup** boots up a kind of modified embedded InnoDB (the libraries **xtrabackup** was linked against). The modifications are necessary to disable InnoDB standard safety checks, such as complaining about the log file not being the right size. This warning is not appropriate for working with backups. These modifications are only for the xtrabackup binary; you do not need a modified *InnoDB* to use **xtrabackup** for your backups.

The *prepare* step uses this "embedded InnoDB" to perform crash recovery on the copied data files, using the copied log file. The `prepare` step is very simple to use: you simply run **xtrabackup** with the *--prepare* option and tell it which directory to prepare, for example, to prepare the previously taken backup run:

```
$ xtrabackup --prepare --target-dir=/data/backups/
```

When this finishes, you should see an `InnoDB shutdown` with a message such as the following, where again the value of *LSN* will depend on your system:

```
InnoDB: Shutdown completed; log sequence number 137345046
160906 11:21:01 completed OK!
```

All following prepares will not change the already prepared data files, you'll see that output says:

```
xtrabackup: This target seems to be already prepared.
xtrabackup: notice: xtrabackup_logfile was already used to '--prepare'.
```

It is not recommended to interrupt xtrabackup process while preparing backup because it may cause data files corruption and backup will become unusable. Backup validity is not guaranteed if prepare process was interrupted.

Note: If you intend the backup to be the basis for further incremental backups, you should use the *--apply-log-only* option when preparing the backup, or you will not be able to apply incremental backups to it. See the documentation on preparing *incremental backups* for more details.

9.3 Restoring a Backup

Warning: Backup needs to be *prepared* before it can be restored.

For convenience, **xtrabackup** binary has the *--copy-back* option to copy the backup to the *datadir* of the server:

```
$ xtrabackup --copy-back --target-dir=/data/backups/
```

If you don't want to save your backup, you can use the *--move-back* option which will move the backed up data to the *datadir*.

If you don't want to use any of the above options, you can additionally use **rsync** or **cp** to restore the files.

Note: The *datadir* must be empty before restoring the backup. Also it's important to note that MySQL server needs to be shut down before restore is performed. You cannot restore to a *datadir* of a running mysqld instance (except when importing a partial backup).

Example of the **rsync** command that can be used to restore the backup can look like this:

```
$ rsync -avrP /data/backup/ /var/lib/mysql/
```

You should check that the restored files have the correct ownership and permissions.

As files' attributes will be preserved, in most cases you will need to change the files' ownership to `mysql` before starting the database server, as they will be owned by the user who created the backup:

```
$ chown -R mysql:mysql /var/lib/mysql
```

Data is now restored and you can start the server.

CHAPTER

TEN

INCREMENTAL BACKUP

xtrabackup supports incremental backups, which means that they can copy only the data that has changed since the last backup.

You can perform many incremental backups between each full backup, so you can set up a backup process such as a full backup once a week and an incremental backup every day, or full backups every day and incremental backups every hour.

Incremental backups work because each *InnoDB* page contains a log sequence number, or *LSN*. The *LSN* is the system version number for the entire database. Each page's *LSN* shows how recently it was changed.

An incremental backup copies each page whose *LSN* is newer than the previous incremental or full backup's *LSN*. There are two algorithms in use to find the set of such pages to be copied. The first one, available with all the server types and versions, is to check the page *LSN* directly by reading all the data pages. The second one, available with *Percona Server for MySQL*, is to enable the changed page tracking feature on the server, which will note the pages as they are being changed. This information will be then written out in a compact separate so-called bitmap file. The **xtrabackup** binary will use that file to read only the data pages it needs for the incremental backup, potentially saving many read requests. The latter algorithm is enabled by default if the **xtrabackup** binary finds the bitmap file. It is possible to specify `--incremental-force-scan` to read all the pages even if the bitmap data is available.

Important: Incremental backups do not actually compare the data files to the previous backup's data files. For this reason, running an incremental backup after a *partial backup* may lead to inconsistent data.

Incremental backups simply read the pages and compare their *LSN* to the last backup's *LSN*. You still need a full backup to recover the incremental changes, however; without a full backup to act as a base, the incremental backups are useless.

You can use the `--incremental-lsn` option to perform an incremental backup without even having the previous backup, if you know its *LSN*.

See also:

Partial Backups

10.1 Creating an Incremental Backup

To make an incremental backup, begin with a full backup as usual. The **xtrabackup** binary writes a file called `xtrabackup_checkpoints` into the backup's target directory. This file contains a line showing the `to_lsn`, which is the database's *LSN* at the end of the backup. *Create the full backup* with a following command:

```
$ xtrabackup --backup --target-dir=/data/backups/base
```

If you look at the `xtrabackup_checkpoints` file, you should see similar content depending on your LSN nuber:

```
backup_type = full-backuped
from_lsn = 0
to_lsn = 1626007
last_lsn = 1626007
compact = 0
recover_binlog_info = 1
```

Now that you have a full backup, you can make an incremental backup based on it. Use the following command:

```
$ xtrabackup --backup --target-dir=/data/backups/inc1 \
--incremental-basedir=/data/backups/base
```

The `/data/backups/inc1/` directory should now contain delta files, such as `ibdata1.delta` and `test/table1.ibd.delta`. These represent the changes since the `LSN 1626007`. If you examine the `xtrabackup_checkpoints` file in this directory, you should see similar content to the following:

```
backup_type = incremental
from_lsn = 1626007
to_lsn = 4124244
last_lsn = 4124244
compact = 0
recover_binlog_info = 1
```

`from_lsn` is the starting LSN of the backup and for incremental it has to be the same as `to_lsn` (if it is the last checkpoint) of the previous/base backup.

It's now possible to use this directory as the base for yet another incremental backup:

```
$ xtrabackup --backup --target-dir=/data/backups/inc2 \
--incremental-basedir=/data/backups/inc1
```

This folder also contains the `xtrabackup_checkpoints`:

```
backup_type = incremental
from_lsn = 4124244
to_lsn = 6938371
last_lsn = 7110572
compact = 0
recover_binlog_info = 1
```

Note: In this case you can see that there is a difference between the `to_lsn` (last checkpoint LSN) and `last_lsn` (last copied LSN), this means that there was some traffic on the server during the backup process.

10.2 Preparing the Incremental Backups

The *--prepare* step for incremental backups is not the same as for full backups. In full backups, two types of operations are performed to make the database consistent: committed transactions are replayed from the log file against the data files, and uncommitted transactions are rolled back. You must skip the rollback of uncommitted transactions when preparing an incremental backup, because transactions that were uncommitted at the time of your backup may be in progress, and it's likely that they will be committed in the next incremental backup. You should use the *--apply-log-only* option to prevent the rollback phase.

Warning: **If you do not use the *--apply-log-only* option to prevent the rollback phase, then your incremental backups will be useless**. After transactions have been rolled back, further incremental backups cannot be applied.

Beginning with the full backup you created, you can prepare it, and then apply the incremental differences to it. Recall that you have the following backups:

```
/data/backups/base
/data/backups/inc1
/data/backups/inc2
```

To prepare the base backup, you need to run *--prepare* as usual, but prevent the rollback phase:

```
$ xtrabackup --prepare --apply-log-only --target-dir=/data/backups/base
```

The output should end with text similar to the following:

```
InnoDB: Shutdown completed; log sequence number 1626007
161011 12:41:04 completed OK!
```

The log sequence number should match the `to_lsn` of the base backup, which you saw previously.

Note: This backup is actually safe to *restore* as-is now, even though the rollback phase has been skipped. If you restore it and start *MySQL*, *InnoDB* will detect that the rollback phase was not performed, and it will do that in the background, as it usually does for a crash recovery upon start. It will notify you that the database was not shut down normally.

To apply the first incremental backup to the full backup, run the following command:

```
$ xtrabackup --prepare --apply-log-only --target-dir=/data/backups/base \
--incremental-dir=/data/backups/inc1
```

This applies the delta files to the files in `/data/backups/base`, which rolls them forward in time to the time of the incremental backup. It then applies the redo log as usual to the result. The final data is in `/data/backups/base`, not in the incremental directory. You should see an output similar to:

```
incremental backup from 1626007 is enabled.
xtrabackup: cd to /data/backups/base
xtrabackup: This target seems to be already prepared with --apply-log-only.
xtrabackup: xtrabackup_logfile detected: size=2097152, start_lsn=(4124244)
...
xtrabackup: page size for /tmp/backups/inc1/ibdata1.delta is 16384 bytes
Applying /tmp/backups/inc1/ibdata1.delta to ./ibdata1...
```

(continues on next page)

(continued from previous page)

```
...
161011 12:45:56 completed OK!
```

Again, the *LSN* should match what you saw from your earlier inspection of the first incremental backup. If you restore the files from `/data/backups/base`, you should see the state of the database as of the first incremental backup.

Warning: *Percona XtraBackup* does not support using the same incremental backup directory to prepare two copies of backup. Do not run `--prepare` with the same incremental backup directory (the value of *–incremental-dir*) more than once.

Preparing the second incremental backup is a similar process: apply the deltas to the (modified) base backup, and you will roll its data forward in time to the point of the second incremental backup:

```
$ xtrabackup --prepare --target-dir=/data/backups/base \
--incremental-dir=/data/backups/inc2
```

Note: `--apply-log-only` should be used when merging all incrementals except the last one. That's why the previous line doesn't contain the `--apply-log-only` option. Even if the `--apply-log-only` was used on the last step, backup would still be consistent but in that case server would perform the rollback phase.

Once prepared incremental backups are the same as the *full backups* and they can be *restored* in the same way.

CHAPTER

ELEVEN

COMPRESSED BACKUP

Percona XtraBackup supports compressed backups: a local or streaming backup can be compressed or decompressed with *xbstream*.

11.1 Creating Compressed Backups

In order to make a compressed backup you'll need to use the `--compress` option:

```
$ xtrabackup --backup --compress --target-dir=/data/compressed/
```

The `--compress` uses the `qpress` tool that you can install via the `percona-release` package configuration tool as follows:

```
$ sudo percona-release enable tools
$ sudo apt update
$ sudo apt install qpress
```

Note: Enable the repository: `percona-release enable-only tools release`

If *Percona XtraBackup* is intented to be used in combination with the upstream MySQL Server, you only need to enable the `tools` repository: `percona-release enable-only tools`.

See also:

Installing Percona XtraBackup from Repositories

If you want to speed up the compression you can use the parallel compression, which can be enabled with `--compress-threads` option. Following example will use four threads for compression:

```
$ xtrabackup --backup --compress --compress-threads=4 \
--target-dir=/data/compressed/
```

Output should look like this

```
...
170223 13:00:38 [01] Compressing ./test/sbtest1.frm to /tmp/compressed/test/sbtest1.
↪frm.qp
170223 13:00:38 [01]        ...done
170223 13:00:38 [01] Compressing ./test/sbtest2.frm to /tmp/compressed/test/sbtest2.
↪frm.qp
170223 13:00:38 [01]        ...done
```

(continues on next page)

(continued from previous page)

```
...
170223 13:00:39 [00] Compressing xtrabackup_info
170223 13:00:39 [00]        ...done
xtrabackup: Transaction log of lsn (9291934) to (9291934) was copied.
170223 13:00:39 completed OK!
```

11.1.1 Preparing the backup

Before you can prepare the backup you'll need to uncompress all the files. *Percona XtraBackup* has implemented *--decompress* option that can be used to decompress the backup.

```
$ xtrabackup --decompress --target-dir=/data/compressed/
```

Note: *--parallel* can be used with *--decompress* option to decompress multiple files simultaneously.

Percona XtraBackup doesn't automatically remove the compressed files. In order to clean up the backup directory you should use *--remove-original* option. Even if they're not removed these files will not be copied/moved over to the datadir if *--copy-back* or *--move-back* are used.

When the files are uncompressed you can prepare the backup with the *--prepare* option:

```
$ xtrabackup --prepare --target-dir=/data/compressed/
```

You should check for a confirmation message:

```
InnoDB: Starting shutdown...
InnoDB: Shutdown completed; log sequence number 9293846
170223 13:39:31 completed OK!
```

Now the files in `/data/compressed/` are ready to be used by the server.

11.1.2 Restoring the backup

xtrabackup has a *--copy-back* option, which performs the restoration of a backup to the server's *datadir*:

```
$ xtrabackup --copy-back --target-dir=/data/backups/
```

It will copy all the data-related files back to the server's *datadir*, determined by the server's `my.cnf` configuration file. You should check the last line of the output for a success message:

```
170223 13:49:13 completed OK!
```

You should check the file permissions after copying the data back. You may need to adjust them with something like:

```
$ chown -R mysql:mysql /var/lib/mysql
```

Now that the *datadir* contains the restored data. You are ready to start the server.

Part V

User's Manual

CHAPTER

TWELVE

PERCONA XTRABACKUP USER MANUAL

12.1 The xtrabackup Binary

The **xtrabackup** binary is a compiled C program that is linked with the *InnoDB* libraries and the standard *MySQL* client libraries.

xtrabackup enables point-in-time backups of *InnoDB* / *XtraDB* tables together with the schema definitions, *MyISAM* tables, and other portions of the server.

The *InnoDB* libraries provide the functionality to apply a log to data files. The *MySQL* client libraries are used to parse command-line options and configuration file.

The tool runs in either `--backup` or `--prepare` mode, corresponding to the two main functions it performs. There are several variations on these functions to accomplish different tasks, and there are two less commonly used modes, `--stats` and `--print-param`.

12.1.1 Other Types of Backups

Incremental Backups

xtrabackup supports incremental backups. It copies only the data that has changed since the last full backup. You can perform many incremental backups between each full backup, so you can set up a backup process such as a full backup once a week and an incremental backup every day, or full backups every day and incremental backups every hour.

Incremental backups work because each InnoDB page (usually 16kb in size) contains a log sequence number, or *LSN*. The *LSN* is the system version number for the entire database. Each page's *LSN* shows how recently it was changed. An incremental backup copies each page whose *LSN* is newer than the previous incremental or full backup's *LSN*. There are two algorithms in use to find the set of such pages to be copied. The first one, available with all the server types and versions, is to check the page *LSN* directly by reading all the data pages. The second one, available with *Percona Server for MySQL*, is to enable the changed page tracking feature on the server, which will note the pages as they are being changed. This information will be then written out in a compact separate so-called bitmap file. The **xtrabackup** binary will use that file to read only the data pages it needs for the incremental backup, potentially saving many read requests. The latter algorithm is enabled by default if the **xtrabackup** binary finds the bitmap file. It is possible to specify `--incremental-force-scan` to read all the pages even if the bitmap data is available.

Incremental backups do not actually compare the data files to the previous backup's data files. In fact, you can use `--incremental-lsn` to perform an incremental backup without even having the previous backup, if you know its *LSN*. Incremental backups simply read the pages and compare their *LSN* to the last backup's *LSN*. You still need a full backup to recover the incremental changes, however; without a full backup to act as a base, the incremental backups are useless.

Creating an Incremental Backup

To make an incremental backup, begin with a full backup as usual. The **xtrabackup** binary writes a file called `xtrabackup_checkpoints` into the backup's target directory. This file contains a line showing the `to_lsn`, which is the database's *LSN* at the end of the backup. *Create the full backup* with a command such as the following:

```
$ xtrabackup --backup --target-dir=/data/backups/base --datadir=/var/lib/mysql/
```

If you look at the `xtrabackup_checkpoints` file, you should see contents similar to the following:

```
backup_type = full-backuped
from_lsn = 0
to_lsn = 1291135
```

Now that you have a full backup, you can make an incremental backup based on it. Use a command such as the following:

```
$ xtrabackup --backup --target-dir=/data/backups/inc1 \
--incremental-basedir=/data/backups/base --datadir=/var/lib/mysql/
```

The `/data/backups/inc1/` directory should now contain delta files, such as `ibdata1.delta` and `test/table1.ibd.delta`. These represent the changes since the `LSN 1291135`. If you examine the `xtrabackup_checkpoints` file in this directory, you should see something similar to the following:

```
backup_type = incremental
from_lsn = 1291135
to_lsn = 1291340
```

The meaning should be self-evident. It's now possible to use this directory as the base for yet another incremental backup:

```
$ xtrabackup --backup --target-dir=/data/backups/inc2 \
--incremental-basedir=/data/backups/inc1 --datadir=/var/lib/mysql/
```

Preparing the Incremental Backups

The *--prepare* step for incremental backups is not the same as for normal backups. In normal backups, two types of operations are performed to make the database consistent: committed transactions are replayed from the log file against the data files, and uncommitted transactions are rolled back. You must skip the rollback of uncommitted transactions when preparing a backup, because transactions that were uncommitted at the time of your backup may be in progress, and it is likely that they will be committed in the next incremental backup. You should use the *--apply-log-only* option to prevent the rollback phase.

Note: If you do not use the *--apply-log-only* option to prevent the rollback phase, then your incremental backups will be useless. After transactions have been rolled back, further incremental backups cannot be applied.

Beginning with the full backup you created, you can prepare it, and then apply the incremental differences to it. Recall that you have the following backups:

```
/data/backups/base
/data/backups/inc1
/data/backups/inc2
```

To prepare the base backup, you need to run *--prepare* as usual, but prevent the rollback phase:

```
xtrabackup --prepare --apply-log-only --target-dir=/data/backups/base
```

The output should end with some text such as the following:

```
101107 20:49:43  InnoDB: Shutdown completed; log sequence number 1291135
```

The log sequence number should match the `to_lsn` of the base backup, which you saw previously.

This backup is actually safe to *restore* as-is now, even though the rollback phase has been skipped. If you restore it and start *MySQL*, *InnoDB* will detect that the rollback phase was not performed, and it will do that in the background, as it usually does for a crash recovery upon start. It will notify you that the database was not shut down normally.

To apply the first incremental backup to the full backup, you should use the following command:

```
xtrabackup --prepare --apply-log-only --target-dir=/data/backups/base \
--incremental-dir=/data/backups/inc1
```

This applies the delta files to the files in `/data/backups/base`, which rolls them forward in time to the time of the incremental backup. It then applies the redo log as usual to the result. The final data is in `/data/backups/base`, not in the incremental directory. You should see some output such as the following:

```
incremental backup from 1291135 is enabled.
xtrabackup: cd to /data/backups/base/
xtrabackup: This target seems to be already prepared.
xtrabackup: xtrabackup_logfile detected: size=2097152, start_lsn=(1291340)
Applying /data/backups/inc1/ibdata1.delta ...
Applying /data/backups/inc1/test/table1.ibd.delta ...
.... snip
101107 20:56:30  InnoDB: Shutdown completed; log sequence number 1291340
```

Again, the *LSN* should match what you saw from your earlier inspection of the first incremental backup. If you restore the files from `/data/backups/base`, you should see the state of the database as of the first incremental backup.

Preparing the second incremental backup is a similar process: apply the deltas to the (modified) base backup, and you will roll its data forward in time to the point of the second incremental backup:

```
xtrabackup --prepare --target-dir=/data/backups/base \
--incremental-dir=/data/backups/inc2
```

Note: *`--apply-log-only`* should be used when merging all incrementals except the last one. That's why the previous line doesn't contain the *`--apply-log-only`* option. Even if the *`--apply-log-only`* was used on the last step, backup would still be consistent but in that case server would perform the rollback phase.

If you wish to avoid the notice that *InnoDB* was not shut down normally, when you applied the desired deltas to the base backup, you can run *`--prepare`* again without disabling the rollback phase.

Restoring Incremental Backups

After preparing the incremental backups, the base directory contains the same data as the full backup. To restoring this backup, you can use this command: `xtrabackup --copy-back --target-dir=BASE-DIR`

You may have to change the ownership as detailed on *Restoring a Backup*.

Incremental Streaming Backups Using xbstream

Incremental streaming backups can be performed with the *xbstream* streaming option. Currently backups are packed in custom **xbstream** format. With this feature, you need to take a BASE backup as well.

Making a base backup

```
$ xtrabackup --backup --target-dir=/data/backups
```

Taking a local backup

```
$ xtrabackup --backup --incremental-lsn=LSN-number --stream=xbstream --target-dir=./ >
↪ incremental.xbstream
```

Unpacking the backup

```
$ xbstream -x < incremental.xbstream
```

Taking a local backup and streaming it to the remote server and unpacking it

```
$ xtrabackup --backup --incremental-lsn=LSN-number --stream=xbstream --target-dir=./
$ ssh user@hostname " cat - | xbstream -x -C > /backup-dir/"
```

Partial Backups

xtrabackup supports taking partial backups when the *innodb_file_per_table* option is enabled. There are three ways to create partial backups:

1. matching the tables names with a regular expression
2. providing a list of table names in a file
3. providing a list of databases

Warning: Do not copy back the prepared backup.

Restoring partial backups should be done by importing the tables, not by using the *–copy-back* option. It is not recommended to run incremental backups after running a partial backup.

Although there are some scenarios where restoring can be done by copying back the files, this may lead to database inconsistencies in many cases and it is not a recommended way to do it.

For the purposes of this manual page, we will assume that there is a database named `test` which contains tables named `t1` and `t2`.

Warning: If any of the matched or listed tables is deleted during the backup, **`xtrabackup`** will fail.

Creating Partial Backups

There are multiple ways of specifying which part of the whole data is backed up:

- Use the *`--tables`* option to list the table names
- Use the *`--tables-file`* option to list the tables in a file
- Use the *`--databases`* option to list the databases
- Use the *`--databases-file`* option to list the databases

The *–tables* Option

The first method involves the xtrabackup *–tables* option. The option's value is a regular expression that is matched against the fully-qualified database name and table name using the `databasename.tablename` format.

To back up only tables in the `test` database, use the following command:

```
$ xtrabackup --backup --datadir=/var/lib/mysql --target-dir=/data/backups/ \
--tables="^test[.].*"
```

To back up only the `test.t1` table, use the following command:

```
$ xtrabackup --backup --datadir=/var/lib/mysql --target-dir=/data/backups/ \
--tables="^test[.]t1"
```

The *–tables-file* Option

The `--tables-file` option specifies a file that can contain multiple table names, one table name per line in the file. Only the tables named in the file will be backed up. Names are matched exactly, case-sensitive, with no pattern or regular expression matching. The table names must be fully-qualified in `databasename.tablename` format.

```
$ echo "mydatabase.mytable" > /tmp/tables.txt
$ xtrabackup --backup --tables-file=/tmp/tables.txt
```

The *–databases* and *–databases-file* options

The ` –databases` option accepts a space-separated list of the databases and tables to backup in the `databasename[.tablename]` format. In addition to this list, make sure to specify the `mysql`, `sys`, and

`performance_schema` databases. These databases are required when restoring the databases using *xtrabackup –copy-back*.

Note: Tables processed during the –prepare step may also be added to the backup even if they are not explicitly listed by the parameter if they were created after the backup started.

```
$ xtrabackup --databases='mysql sys performance_schema test ...'
```

The `--databases-file` Option

The *–databases-file* option specifies a file that can contain multiple databases and tables in the `databasename[.tablename]` format, one element name per line in the file. Names are matched exactly, case-sensitive, with no pattern or regular expression matching.

Note: Tables processed during the –prepare step may also be added to the backup even if they are not explicitly listed by the parameter if they were created after the backup started.

Preparing Partial Backups

The procedure is analogous to *restoring individual tables* : apply the logs and use the *–export* option:

```
$ xtrabackup --prepare --export --target-dir=/path/to/partial/backup
```

When you use the *–prepare* option on a partial backup, you will see warnings about tables that don't exist. This is because these tables exist in the data dictionary inside InnoDB, but the corresponding *.ibd* files don't exist. They were not copied into the backup directory. These tables will be removed from the data dictionary, and when you restore the backup and start InnoDB, they will no longer exist and will not cause any errors or warnings to be printed to the log file.

> Could not find any file associated with the tablespace ID: 5
>
> Use –innodb-directories to find the tablespace files. If that fails then use –innodb-force-recovery=1 to ignore this and to permanently lose all changes to the missing tablespace(s).

Restoring Partial Backups

Restoring should be done by *restoring individual tables* in the partial backup to the server.

It can also be done by copying back the prepared backup to a "clean" *datadir* (in that case, make sure to include the `mysql` database) to the datadir you are moving the backup to. A system database can be created with the following:

```
$ sudo mysql --initialize --user=mysql
```

Once you start the server, you may see mysql complaining about missing tablespaces:

```
2021-07-19T12:42:11.077200Z 1 [Warning] [MY-012351] [InnoDB] Tablespace 4, name
↪'test1/t1', file './d2/test1.ibd' is missing!
2021-07-19T12:42:11.077300Z 1 [Warning] [MY-012351] [InnoDB] Tablespace 4, name
↪'test1/t1', file './d2/test1.ibd' is missing!
```

In order to clean the orphan database from the data dictionary, you must manually create the missing database directory and then `DROP` this database from the server.

Example of creating the missing database:

```
$ mkdir /var/lib/mysql/test1/d2
```

Example of dropping the database from the server:

```
mysql> DROP DATABASE d2;
Query OK, 2 rows affected (0.5 sec)
```

12.1.2 Advanced Features

Analyzing Table Statistics

The **xtrabackup** binary can analyze InnoDB data files in read-only mode to give statistics about them. To do this, you should use the *--stats* option. You can combine this with the *--tables* option to limit the files to examine. It also uses the *--use-memory* option.

You can perform the analysis on a running server, with some chance of errors due to the data being changed during analysis. Or, you can analyze a backup copy of the database. Either way, to use the statistics feature, you need a clean copy of the database including correctly sized log files, so you need to execute with *--prepare* twice to use this functionality on a backup.

The result of running on a backup might look like the following:

```
<INDEX STATISTICS>
  table: test/table1, index: PRIMARY, space id: 12, root page 3
  estimated statistics in dictionary:
    key vals: 25265338, leaf pages 497839, size pages 498304
  real statistics:
     level 2 pages: pages=1, data=5395 bytes, data/pages=32%
     level 1 pages: pages=415, data=6471907 bytes, data/pages=95%
        leaf pages: recs=25958413, pages=497839, data=7492026403 bytes, data/pages=91%
```

This can be interpreted as follows:

- The first line simply shows the table and index name and its internal identifiers. If you see an index named `GEN_CLUST_INDEX`, that is the table's clustered index, automatically created because you did not explicitly create a `PRIMARY KEY`.
- The estimated statistics in dictionary information is similar to the data that's gathered through `ANALYZE TABLE` inside of *InnoDB* to be stored as estimated cardinality statistics and passed to the query optimizer.
- The real statistics information is the result of scanning the data pages and computing exact information about the index.
- `The level <X> pages`: output means that the line shows information about pages at that level in the index tree. The larger `<X>` is, the farther it is from the leaf pages, which are level 0. The first line is the root page.
- The `leaf pages` output shows the leaf pages, of course. This is where the table's data is stored.
- The `external pages`: output (not shown) shows large external pages that hold values too long to fit in the row itself, such as long `BLOB` and `TEXT` values.
- The `recs` is the real number of records (rows) in leaf pages.
- The `pages` is the page count.
- The `data` is the total size of the data in the pages, in bytes.
- The `data/pages` is calculated as (`data` / (`pages` * `PAGE_SIZE`)) * 100%. It will never reach 100% because of space reserved for page headers and footers.

A more detailed example is posted as a MySQL Performance Blog post.

Script to Format Output

The following script can be used to summarize and tabulate the output of the statistics information:

tabulate-xtrabackup-stats.pl

```
#!/usr/bin/env perl
use strict;
use warnings FATAL => 'all';
my $script_version = "0.1";

my $PG_SIZE = 16_384; # InnoDB defaults to 16k pages, change if needed.
my ($cur_idx, $cur_tbl);
my (%idx_stats, %tbl_stats);
my ($max_tbl_len, $max_idx_len) = (0, 0);
while ( my $line = <> ) {
   if ( my ($t, $i) = $line =~ m/table: (.*), index: (.*), space id:/ ) {
      $t =~ s!/!.!;
      $cur_tbl = $t;
      $cur_idx = $i;
      if ( length($i) > $max_idx_len ) {
         $max_idx_len = length($i);
      }
      if ( length($t) > $max_tbl_len ) {
         $max_tbl_len = length($t);
      }
   }
   elsif ( my ($kv, $lp, $sp) = $line =~ m/key vals: (\d+), \D*(\d+), \D*(\d+)/ ) {
      @{$idx_stats{$cur_tbl}->{$cur_idx}}{qw(est_kv est_lp est_sp)} = ($kv, $lp, $sp);
      $tbl_stats{$cur_tbl}->{est_kv} += $kv;
      $tbl_stats{$cur_tbl}->{est_lp} += $lp;
      $tbl_stats{$cur_tbl}->{est_sp} += $sp;
   }
   elsif ( my ($l, $pages, $bytes) = $line =~ m/(?:level (\d+)|leaf) pages:.*pages=(\
↪d+), data=(\d+) bytes/ ) {
      $l ||= 0;
      $idx_stats{$cur_tbl}->{$cur_idx}->{real_pages} += $pages;
      $idx_stats{$cur_tbl}->{$cur_idx}->{real_bytes} += $bytes;
      $tbl_stats{$cur_tbl}->{real_pages} += $pages;
      $tbl_stats{$cur_tbl}->{real_bytes} += $bytes;
   }
}

my $hdr_fmt = "%${max_tbl_len}s %${max_idx_len}s %9s %10s %10s\n";
my @headers = qw(TABLE INDEX TOT_PAGES FREE_PAGES PCT_FULL);
printf $hdr_fmt, @headers;

my $row_fmt = "%${max_tbl_len}s %${max_idx_len}s %9d %10d %9.1f%%\n";
foreach my $t ( sort keys %tbl_stats ) {
   my $tbl = $tbl_stats{$t};
   printf $row_fmt, $t, "", $tbl->{est_sp}, $tbl->{est_sp} - $tbl->{real_pages},
      $tbl->{real_bytes} / ($tbl->{real_pages} * $PG_SIZE) * 100;
   foreach my $i ( sort keys %{$idx_stats{$t}} ) {
      my $idx = $idx_stats{$t}->{$i};
      printf $row_fmt, $t, $i, $idx->{est_sp}, $idx->{est_sp} - $idx->{real_pages},
         $idx->{real_bytes} / ($idx->{real_pages} * $PG_SIZE) * 100;
   }
}
```

Sample Script Output

The output of the above Perl script, when run against the sample shown in the previously mentioned blog post, will appear as follows:

```
          TABLE            INDEX TOT_PAGES FREE_PAGES    PCT_FULL
art.link_out104                     832383      38561       86.8%
art.link_out104          PRIMARY    498304         49       91.9%
art.link_out104        domain_id     49600       6230       76.9%
art.link_out104     domain_id_2     26495       3339       89.1%
art.link_out104 from_message_id     28160        142       96.3%
art.link_out104    from_site_id     38848       4874       79.4%
art.link_out104   revert_domain    153984      19276       71.4%
art.link_out104    site_message     36992       4651       83.4%
```

The columns are the table and index, followed by the total number of pages in that index, the number of pages not actually occupied by data, and the number of bytes of real data as a percentage of the total size of the pages of real data. The first line in the above output, in which the `INDEX` column is empty, is a summary of the entire table.

Working with Binary Logs

The `xtrabackup` binary integrates with the `log_status table`. This integration enables `xtrabackup` to print out the backup's corresponding binary log position, so that you can use this binary log position to provision a new replica or perform point-in-time recovery.

Finding the Binary Log Position

You can find the binary log position corresponding to a backup after the backup has been taken. If your backup is from a server with binary logging enabled, `xtrabackup` creates a file named `xtrabackup_binlog_info` in the target directory. This file contains the binary log file name and position of the exact point when the backup was taken.

The output is similar to the following during the backup stage:

```
210715 14:14:59 Backup created in directory '/backup/'
MySQL binlog position: filename 'binlog.000002', position '156'
. . .
210715 14:15:00 completed OK!
```

Note: As of Percona XtraBackup 8.0.26-18.0, xtrabackup no longer creates the `xtrabackup_binlog_pos_innodb` file. This change is because MySQL and Percona Server no longer update the binary log information on global transaction system section of `ibdata`. You should rely on `xtrabackup_binlog_info` regardless of the storage engine in use.

Point-In-Time Recovery

To perform a point-in-time recovery from an `xtrabackup` backup, you should prepare and restore the backup, and then replay binary logs from the point shown in the `xtrabackup_binlog_info` file.

A more detailed procedure is found *here*.

Setting Up a New Replication Replica

To set up a new replica, you should prepare the backup, and restore it to the data directory of your new replication replica. If you are using version 8.0.22 or earlier, in your `CHANGE MASTER TO` command, use the binary log filename and position shown in the `xtrabackup_binlog_info` file to start replication.

If you are using 8.0.23 or later, use the CHANGE_REPLICATION_SOURCE_TO and the appropriate options. `CHANGE_MASTER_TO` is deprecated.

A more detailed procedure is found in *How to setup a replica for replication in 6 simple steps with Percona XtraBackup*.

Restoring Individual Tables

With *Percona XtraBackup*, you can export individual tables from any *InnoDB* database, and import them into *Percona Server for MySQL* with *XtraDB* or *MySQL* 8.0. (The source doesn't have to be *XtraDB* or *MySQL* 8.0, but the destination does.) This only works on individual *.ibd* files, and cannot export a table that is not contained in its own *.ibd* file.

Let's see how to export and import the following table:

```
CREATE TABLE export_test (
a int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Exporting the Table

This table should be created in *innodb_file_per_table* mode, so after taking a backup as usual with *--backup*, the *.ibd* file should exist in the target directory:

```
$ find /data/backups/mysql/ -name export_test.*
/data/backups/mysql/test/export_test.ibd
```

when you prepare the backup, add the extra parameter *--export* to the command. Here is an example:

```
$ xtrabackup --prepare --export --target-dir=/data/backups/mysql/
```

Note: If you're trying to restore *encrypted InnoDB tablespace* table you'll need to specify the keyring file as well:

```
$ xtrabackup --prepare --export --target-dir=/tmp/table \
--keyring-file-data=/var/lib/mysql-keyring/keyring
```

Now you should see a *.exp* file in the target directory:

```
$ find /data/backups/mysql/ -name export_test.*
/data/backups/mysql/test/export_test.exp
/data/backups/mysql/test/export_test.ibd
/data/backups/mysql/test/export_test.cfg
```

These three files are all you need to import the table into a server running *Percona Server for MySQL* with *XtraDB* or *MySQL* 8.0. In case server is using InnoDB Tablespace Encryption additional `.cfp` file be listed for encrypted tables.

Note: *MySQL* uses `.cfg` file which contains *InnoDB* dictionary dump in special format. This format is different from the `.exp`` one which is used in *XtraDB* for the same purpose. Strictly speaking, a `.cfg`` file is not required to import a tablespace to *MySQL* 8.0 or *Percona Server for MySQL* 8.0. A tablespace will be imported successfully even if it is from another server, but *InnoDB* will do schema validation if the corresponding `.cfg` file is present in the same directory.

Importing the Table

On the destination server running *Percona Server for MySQL* with *XtraDB* and innodb_import_table_from_xtrabackup option enabled, or *MySQL* 8.0, create a table with the same structure, and then perform the following steps:

1. Run the `ALTER TABLE test.export_test DISCARD TABLESPACE;` command. If you see this error then you must enable *innodb_file_per_table* and create the table again.

   Error

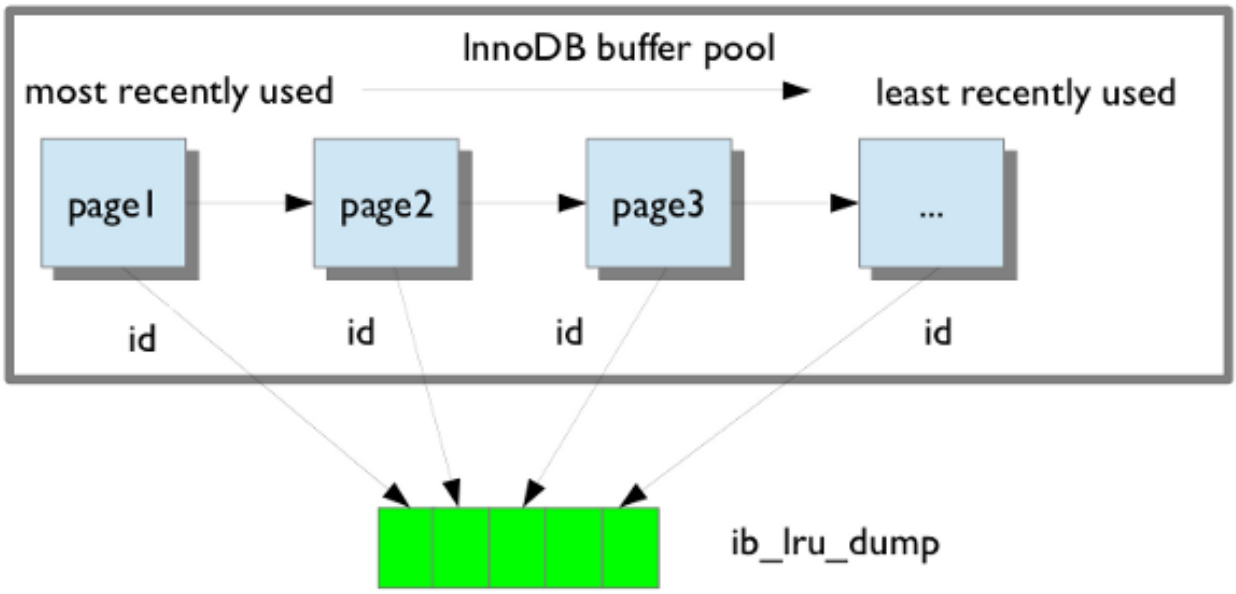
   ERROR 1030 (HY000): Got error -1 from storage engine

2. Copy the exported files to the `test/` subdirectory of the destination server's data directory
3. Run `ALTER TABLE test.export_test IMPORT TABLESPACE;`

The table should now be imported, and you should be able to `SELECT` from it and see the imported data.

LRU dump backup

Percona XtraBackup includes a saved buffer pool dump into a backup to enable reducing the warm up time. It restores the buffer pool state from `ib_buffer_pool` file after restart. *Percona XtraBackup* discovers `ib_buffer_pool` and backs it up automatically.



If the buffer restore option is enabled in `my.cnf` buffer pool will be in the warm state after backup is restored.

See also:

***MySQL* Documentation: Saving and Restoring the Buffer Pool State** https://dev.mysql.com/doc/refman/8.0/en/innodb-preload-buffer-pool.html

Streaming Backups

Streaming mode, supported by *Percona XtraBackup*, sends backup to `STDOUT` in the *xbstream* format instead of copying files to the backup directory.

This allows you to use other programs to filter the output of the backup, providing greater flexibility for storage of the backup. For example, compression is achieved by piping the output to a compression utility. One of the benefits of streaming backups and using Unix pipes is that the backups can be automatically encrypted.

To use the streaming feature, you must use the *--stream*, providing the format of the stream (`xbstream`) and where to store the temporary files:

```
$ xtrabackup --stream=xbstream --target-dir=/tmp
```

`xtrabackup` uses *xbstream* to stream all of the data files to `STDOUT`, in a special `xbstream` format. After it finishes streaming all of the data files to `STDOUT`, it stops xtrabackup and streams the saved log file too.

See also:

More information about *xbstream* *The xbstream binary*

When compression is enabled, **`xtrabackup`** compresses all output data, except the meta and non-InnoDB files which are not compressed, using the specified compression algorithm. The only currently supported algorithm is `quicklz`.

The resulting files have the `qpress` archive format, i.e. every *.qp file produced by xtrabackup is essentially a one-file qpress archive and can be extracted and uncompressed by the qpress file archiver which is available from *Percona Software repositories*.

Using *xbstream* as a stream option, backups can be copied and compressed in parallel which can significantly speed up the backup process. In case backups were both compressed and encrypted, they'll need to decrypted first in order to be uncompressed.

Task	Command
Stream the backup into an archive named `backup.xbstream`	`xtrabackup --backup --stream=xbstream --target-dir=./ > backup.xbstream`
Stream the backup into a *compressed* archive named `backup.xbstream`	`xtrabackup --backup --stream=xbstream --compress --target-dir=./ > backup.xbstream`
Encrypt the backup	$ xtrabackup –backup –stream=xbstream ./ > backup.xbstream gzip - | openssl des3 -salt -k "password" > backup.xbstream.gz.des3
Unpack the backup to the current directory	`xbstream -x < backup.xbstream`
Send the backup compressed directly to another host and unpack it	`xtrabackup --backup --compress --stream=xbstream --target-dir=./ | ssh user@otherhost "xbstream -x"`

Send the backup to another server using `netcat`.	On the destination host:
	`$ nc -l 9999 | cat - > /data/backups/backup.xbstream`
	On the source host:
	`$ xtrabackup --backup --stream=xbstream ./ | nc desthost 9999`

Send the backup to another server using a one-liner:	$ ssh user@desthost "(nc -l 9999 > /data/backups/backup.xbstream &)" && xtrabackup –backup –stream=xbstream ./ | nc desthost 9999
Throttle the throughput to 10MB/sec using the `pipe viewer` tool[1]	$ xtrabackup –backup –stream=xbstream ./ | pv -q -L10m ssh user@desthost "cat - > /data/backups/backup.xbstream"

Checksumming the backup during the streaming:	On the destination host:
	`$ nc -l 9999 | tee >(sha1sum > destination_checksum) > /data/backups/backup.xbstream`
	On the source host:
	`$ xtrabackup --backup --stream=xbstream ./ | tee >(sha1sum > source_checksum) | nc desthost 9999`
	Compare the checksums on the source host:
	`$ cat source_checksum`
	`65e4f916a49c1f216e0887ce54cf59bf3934dbad  -`
	Compare the checksums on the destination host:
	`$ cat destination_checksum`
	`65e4f916a49c1f216e0887ce54cf59bf3934dbad  -`

Parallel compression with parallel copying backup	`xtrabackup --backup --compress --compress-threads=8 --stream=xbstream --parallel=4 --target-dir=./ > backup.xbstream`

[1] Install from the official site or from the distribution package (`apt install pv`)

Note that the streamed backup will need to be prepared before restoration. Streaming mode does not prepare the backup.

Encrypting Backups

Percona XtraBackup supports encrypting and decrypting local and streaming backups with *xbstream* option adding another layer of protection. The encryption is implemented using the `libgcrypt` library from GnuPG.

Creating Encrypted Backups

To make an encrypted backup the following options need to be specified (options *--encrypt-key* and *--encrypt-key-file* are mutually exclusive, i.e. just one of them needs to be provided):

- *--encrypt*
- :option:` –encrypt-key`
- :option:` –encrypt-key-file`

Both the *--encrypt-key* option and *--encrypt-key-file* option can be used to specify the encryption key. An encryption key can be generated with a command like `openssl rand -base64 24`

Example output of that command should look like this:

```
GCHFLrDFVx6UAsRb88uLVbAVWbK+Yzfs
```

This value then can be used as the encryption key

The `--encrypt-key` Option

Example of the **xtrabackup** command using the *--encrypt-key* should look like this:

```
$  xtrabackup --backup --encrypt=AES256 --encrypt-key=
↪"GCHFLrDFVx6UAsRb88uLVbAVWbK+Yzfs" --target-dir=/data/backup
```

The `--encrypt-key-file` Option

Use the *--encrypt-key-file* option as follows:

```
$ xtrabackup --backup --encrypt=AES256 --encrypt-key-file=/data/backups/keyfile --
↪target-dir=/data/backup
```

Note: Depending on the text editor that you use to make the `KEYFILE`, the editor can automatically insert the CRLF (end of line) character. This will cause the key size to grow and thus making it invalid. The suggested way to create the file is by using the command line: `echo -n "GCHFLrDFVx6UAsRb88uLVbAVWbK+Yzfs" > /data/backups/keyfile`.

Optimizing the encryption process

Two new options are available for encrypted backups that can be used to speed up the encryption process. These are *--encrypt-threads* and *--encrypt-chunk-size*. By using the *--encrypt-threads* option multiple threads can be specified to be used for encryption in parallel. Option *--encrypt-chunk-size* can be used to specify the size (in bytes) of the working encryption buffer for each encryption thread (default is 64K).

Decrypting Encrypted Backups

Backups can be decrypted with *The xbcrypt binary*. The following one-liner can be used to encrypt the whole folder:

```
$ for i in `find . -iname "*\.xbcrypt"`; do xbcrypt -d --encrypt-key-file=/root/secret_
↪key --encrypt-algo=AES256 < $i > $(dirname $i)/$(basename $i .xbcrypt) && rm $i;␣
↪done
```

Percona XtraBackup *--decrypt* option has been implemented that can be used to decrypt the backups:

```
$ xtrabackup --decrypt=AES256 --encrypt-key="GCHFLrDFVx6UAsRb88uLVbAVWbK+Yzfs" --
↪target-dir=/data/backup/
```

Percona XtraBackup doesn't automatically remove the encrypted files. In order to clean up the backup directory users should remove the `*.xbcrypt` files.

Note: *--parallel* can be used with *--decrypt* option to decrypt multiple files simultaneously.

When the files are decrypted, the backup can be prepared.

Preparing Encrypted Backups

After the backups have been decrypted, they can be prepared in the same way as the standard full backups with the *--prepare* option:

```
$ xtrabackup --prepare --target-dir=/data/backup/
```

Restoring Encrypted Backups

xtrabackup offers the *--copy-back* option to restore a backup to the server's *datadir*:

```
$ xtrabackup --copy-back --target-dir=/data/backup/
```

It will copy all the data-related files back to the server's *datadir*, determined by the server's `my.cnf` configuration file. You should check the last line of the output for a success message:

```
150318 11:08:13  xtrabackup: completed OK!
```

See also:

GnuPG Documentation: `libgcrypt` library http://www.gnupg.org/documentation/manuals/gcrypt/

Handling `FLUSH TABLES WITH READ LOCK`

When making backups, `FLUSH TABLES WITH READ LOCK` is used before the non-InnoDB files are backed up to ensure that the backup is consistent. `FLUSH TABLES WITH READ LOCK` can be run even though there may be a running query that has been executing for hours.

In this case, everything is locked in the `Waiting for table flush` or `Waiting for master to send event` state. Killing the `FLUSH TABLES WITH READ LOCK` does not correct this problem. The only way to get the server operating normally again is to kill off the long running queries that blocked it to begin with. This means that if there are long running queries `FLUSH TABLES WITH READ LOCK` can get stuck, leaving server in read-only mode until waiting for these queries to complete.

Note: All described in this section has no effect when backup locks are used. *Percona XtraBackup* will use Backup locks where available as a lightweight alternative to `FLUSH TABLES WITH READ LOCK`. This feature is available in *Percona Server for MySQL* 5.6+. *Percona XtraBackup* uses this automatically to copy non-InnoDB data to avoid blocking DML queries that modify InnoDB tables.

In order to prevent this from happening two things have been implemented:

- **xtrabackup** can wait for a good moment to issue the global lock.
- **xtrabackup** can kill all or only SELECT queries which are preventing the global lock from being acquired

Waiting for queries to finish

Good moment to issue a global lock is the moment when there are no long queries running. But waiting for a good moment to issue the global lock for extended period of time isn't always good approach, as it can extend the time needed for backup to take place. To prevent **xtrabackup** from waiting to issue `FLUSH TABLES WITH READ LOCK` for too long, new option has been implemented: *`--ftwrl-wait-timeout`* option can be used to limit the waiting time. If the good moment to issue the lock did not happen during this time, **xtrabackup** will give up and exit with an error message and backup will not be taken. Zero value for this option turns off the feature (which is default).

Another possibility is to specify the type of query to wait on. In this case *`--ftwrl-wait-query-type`*. Possible values are `all` and `update`. When `all` is used **xtrabackup** will wait for all long running queries (execution time longer than allowed by *`--ftwrl-wait-threshold`*) to finish before running the `FLUSH TABLES WITH READ LOCK`. When `update` is used **xtrabackup** will wait on `UPDATE/ALTER/REPLACE/INSERT` queries to finish.

Although the time needed for a specific query to complete is hard to predict, we can assume that the queries that have been running for a long time are not likely to finish soon. The queries which are running for a short time are likely to finish shortly. **xtrabackup** can use the value of *`--ftwrl-wait-threshold`* option to specify which query is long running and will likely block global lock for a while. In order to use this option xtrabackup user should have `PROCESS` and `SUPER` privileges.

Killing the blocking queries

The second option is to kill all the queries which prevent from acquiring the global lock. In this case, all queries which run longer than `FLUSH TABLES WITH READ LOCK` are potential blockers. Although all queries can be killed, additional time can be specified for the short running queries to finish using the *`--kill-long-queries-timeout`* option. This option specifies the time for queries to complete, after the value is reached, all the running queries will be killed. The default value is zero, which turns this feature off.

The *`--kill-long-query-type`* option can be used to specify all or only `SELECT` queries that are preventing global lock from being acquired. In order to use this option xtrabackup user should have `PROCESS` and `SUPER` privileges.

Options summary

- *`--ftwrl-wait-timeout`* (seconds) - how long to wait for a good moment. Default is 0, not to wait.
- *`--ftwrl-wait-query-type`* - which long queries should be finished before `FLUSH TABLES WITH READ LOCK` is run. Default is all.
- *`--ftwrl-wait-threshold`* (seconds) - how long query should be running before we consider it long running and potential blocker of global lock.
- *`--kill-long-queries-timeout`* (seconds) - how many time we give for queries to complete after `FLUSH TABLES WITH READ LOCK` is issued before start to kill. Default if `0`, not to kill.
- *`--kill-long-query-type`* - which queries should be killed once `kill-long-queries-timeout` has expired.

Example

Running the **`xtrabackup`** with the following options will cause **`xtrabackup`** to spend no longer than 3 minutes waiting for all queries older than 40 seconds to complete.

```
$  xtrabackup --backup --ftwrl-wait-threshold=40 \
--ftwrl-wait-query-type=all --ftwrl-wait-timeout=180 \
--kill-long-queries-timeout=20 --kill-long-query-type=all \
--target-dir=/data/backups/
```

After `FLUSH TABLES WITH READ LOCK` is issued, **`xtrabackup`** will wait for 20 seconds for lock to be acquired. If lock is still not acquired after 20 seconds, it will kill all queries which are running longer that the `FLUSH TABLES WITH READ LOCK`.

Accelerating the backup process

Copying with the `--parallel` and *–compress-threads* Options

When making a local or streaming backup with *xbstream* option, multiple files can be copied at the same time when using the *`--parallel`* option. This option specifies the number of threads created by **`xtrabackup`** to copy data files.

To take advantage of this option either the multiple tablespaces option must be enabled (*innodb_file_per_table*) or the shared tablespace must be stored in multiple *ibdata* files with the *innodb_data_file_path* option. Having multiple files for the database (or splitting one into many) doesn't have a measurable impact on performance.

As this feature is implemented **at the file level**, concurrent file transfer can sometimes increase I/O throughput when doing a backup on highly fragmented data files, due to the overlap of a greater number of random read requests. You should consider tuning the filesystem also to obtain the maximum performance (e.g. checking fragmentation).

If the data is stored on a single file, this option will have no effect.

To use this feature, simply add the option to a local backup, for example:

```
$ xtrabackup --backup --parallel=4 --target-dir=/path/to/backup
```

By using the *xbstream* in streaming backups, you can additionally speed up the compression process with the *--compress-threads* option. This option specifies the number of threads created by **xtrabackup** for for parallel data compression. The default value for this option is 1.

To use this feature, simply add the option to a local backup, for example:

```
$ xtrabackup --backup --stream=xbstream --compress --compress-threads=4 --target-dir=.
↪/ > backup.xbstream
```

Before applying logs, compressed files will need to be uncompressed.

The `--rsync` Option

In order to speed up the backup process and to minimize the time `FLUSH TABLES WITH READ LOCK` is blocking the writes, the option *--rsync* should be used. When this option is specified, **xtrabackup** uses `rsync` to copy all non-InnoDB files instead of spawning a separate `cp` for each file, which can be much faster for servers with a large number of databases or tables. **xtrabackup** will call the `rsync` twice, once before the `FLUSH TABLES WITH READ LOCK` and once during to minimize the time the read lock is being held. During the second `rsync` call, it will only synchronize the changes to non-transactional data (if any) since the first call performed before the `FLUSH TABLES WITH READ LOCK`. Note that *Percona XtraBackup* will use Backup locks where available as a lightweight alternative to `FLUSH TABLES WITH READ LOCK`. This feature is available in *Percona Server for MySQL* 5.6+. *Percona XtraBackup* uses this automatically to copy non-InnoDB data to avoid blocking DML queries that modify InnoDB tables.

Note: This option cannot be used together with the *--stream* option.

Point-In-Time recovery

Recovering up to particular moment in database's history can be done with **xtrabackup** and the binary logs of the server.

Note that the binary log contains the operations that modified the database from a point in the past. You need a full *datadir* as a base, and then you can apply a series of operations from the binary log to make the data match what it was at the point in time you want.

```
$ xtrabackup --backup --target-dir=/path/to/backup
$ xtrabackup --prepare --target-dir=/path/to/backup
```

For more details on these procedures, see *Creating a backup* and *Preparing a backup*.

Now, suppose that some time has passed, and you want to restore the database to a certain point in the past, having in mind that there is the constraint of the point where the snapshot was taken.

To find out what is the situation of binary logging in the server, execute the following queries:

```
mysql> SHOW BINARY LOGS;
+------------------+-----------+
| Log_name         | File_size |
+------------------+-----------+
| mysql-bin.000001 |       126 |
| mysql-bin.000002 |      1306 |
| mysql-bin.000003 |       126 |
| mysql-bin.000004 |       497 |
+------------------+-----------+
```

and

```
mysql> SHOW MASTER STATUS;
+------------------+----------+--------------+------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| mysql-bin.000004 |      497 |              |                  |
+------------------+----------+--------------+------------------+
```

The first query will tell you which files contain the binary log and the second one which file is currently being used to record changes, and the current position within it. Those files are stored usually in the *datadir* (unless other location is specified when the server is started with the `--log-bin=` option).

To find out the position of the snapshot taken, see the `xtrabackup_binlog_info` at the backup's directory:

```
$ cat /path/to/backup/xtrabackup_binlog_info
mysql-bin.000003      57
```

This will tell you which file was used at moment of the backup for the binary log and its position. That position will be the effective one when you restore the backup:

```
$ xtrabackup --copy-back --target-dir=/path/to/backup
```

As the restoration will not affect the binary log files (you may need to adjust file permissions, see *Restoring a Backup*), the next step is extracting the queries from the binary log with **`mysqlbinlog`** starting from the position of the snapshot and redirecting it to a file

```
$ mysqlbinlog /path/to/datadir/mysql-bin.000003 /path/to/datadir/mysql-bin.000004 \
    --start-position=57 > mybinlog.sql
```

Note that if you have multiple files for the binary log, as in the example, you have to extract the queries with one process, as shown above.

Inspect the file with the queries to determine which position or date corresponds to the point-in-time wanted. Once determined, pipe it to the server. Assuming the point is `11-12-25 01:00:00`:

```
$ mysqlbinlog /path/to/datadir/mysql-bin.000003 /path/to/datadir/mysql-bin.000004 \
    --start-position=57 --stop-datetime="11-12-25 01:00:00" | mysql -u root -p
```

and the database will be rolled forward up to that Point-In-Time.

Making Backups in Replication Environments

There are options specific to back up from a replication replica.

The `--slave-info` Option

This option is useful when backing up a replication replica server. It prints the binary log position and name of the source server. It also writes this information to the `xtrabackup_slave_info` file as a `CHANGE MASTER` statement.

This option is useful for setting up a new replica for this source. You can start a replica server with this backup and issue the statement saved in the `xtrabackup_slave_info` file. More details of this procedure can be found in *How to setup a replica for replication in 6 simple steps with Percona XtraBackup*.

The `--safe-slave-backup` Option

In order to assure a consistent replication state, this option stops the replication SQL thread and waits to start backing up until `Slave_open_temp_tables` in `SHOW STATUS` is zero. If there are no open temporary tables, the backup will take place, otherwise the SQL thread will be started and stopped until there are no open temporary tables. The backup will fail if `Slave_open_temp_tables` does not become zero after *`--safe-slave-backup-timeout`* seconds (defaults to 300 seconds). The replication SQL thread will be restarted when the backup finishes.

Using this option is always recommended when taking backups from a replica server.

> **Warning:** Make sure your replica is a true replica of the source before using it as a source for backup. A good tool to validate a replica is pt-table-checksum.

Store backup history on the server

Percona XtraBackup supports storing the backups history on the server. This feature was implemented in *Percona XtraBackup* 2.2. Storing backup history on the server was implemented to provide users with additional information about backups that are being taken. Backup history information will be stored in the *PERCONA_SCHEMA.XTRABACKUP_HISTORY* table.

To use this feature the following options are available:

- *`--history`* =<name> : This option enables the history feature and allows the user to specify a backup series name that will be placed within the history record.
- *`--incremental-history-name`* =<name> : This option allows an incremental backup to be made based on a specific history series by name. **`xtrabackup`** will search the history table looking for the most recent (highest `to_lsn`) backup in the series and take the `to_lsn` value to use as it's starting lsn. This is mutually exclusive with *`--incremental-history-uuid`*, *`--incremental-basedir`* and *`--incremental-lsn`* options. If no valid LSN can be found (no series by that name) **`xtrabackup`** will return with an error.
- *`--incremental-history-uuid`* =<uuid> : Allows an incremental backup to be made based on a specific history record identified by UUID. **`xtrabackup`** will search the history table looking for the record matching UUID and take the `to_lsn` value to use as it's starting LSN. This options is mutually exclusive with *`--incremental-basedir`*, *`--incremental-lsn`* and *`--incremental-history-name`* options. If no valid LSN can be found (no record by that UUID or missing `to_lsn`), **`xtrabackup`** will return with an error.

Note: Backup that's currently being performed will **NOT** exist in the xtrabackup_history table within the resulting backup set as the record will not be added to that table until after the backup has been taken.

If you want access to backup history outside of your backup set in the case of some catastrophic event, you will need to either perform a `mysqldump`, partial backup or `SELECT *` on the history table after **`xtrabackup`** completes and store the results with you backup set.

For the necessary privileges, see *Permissions and Privileges Needed*.

PERCONA_SCHEMA.XTRABACKUP_HISTORY table

This table contains the information about the previous server backups. Information about the backups will only be written if the backup was taken with *`--history`* option.

Column Name	Description
uuid	Unique backup id
name	User provided name of backup series. There may be multiple entries with the same name used to identify related backups in a series.
tool_name	Name of tool used to take backup
tool_command	Exact command line given to the tool with –password and –encryption_key obfuscated
tool_version	Version of tool used to take backup
ibbackup_version	Version of the xtrabackup binary used to take backup
server_version	Server version on which backup was taken
start_time	Time at the start of the backup
end_time	Time at the end of the backup
lock_time	Amount of time, in seconds, spent calling and holding locks for `FLUSH TABLES WITH READ LOCK`
binlog_pos	Binlog file and position at end of `FLUSH TABLES WITH READ LOCK`
innodb_from_lsn	LSN at beginning of backup which can be used to determine prior backups
innodb_to_lsn	LSN at end of backup which can be used as the starting lsn for the next incremental
partial	Is this a partial backup, if `N` that means that it's the full backup
incremental	Is this an incremental backup
format	Description of result format (`xbstream`)
compact	Is this a compact backup
compressed	Is this a compressed backup
encrypted	Is this an encrypted backup

Limitations

- *`--history`* option must be specified only on the command line and not within a configuration file in order to be effective.
- *`--incremental-history-name`* and *`--incremental-history-uuid`* options must be specified only on the command line and not within a configuration file in order to be effective.

12.1.3 Implementation

Implementation Details

This page contains notes on various internal aspects of the **xtrabackup** tool's operation.

File Permissions

xtrabackup opens the source data files in read-write mode, although it does not modify the files. This means that you must run **xtrabackup** as a user who has permission to write the data files. The reason for opening the files in read-write mode is that **xtrabackup** uses the embedded *InnoDB* libraries to open and read the files, and *InnoDB* opens them in read-write mode because it normally assumes it is going to write to them.

Tuning the OS Buffers

Because **xtrabackup** reads large amounts of data from the filesystem, it uses `posix_fadvise()` where possible, to instruct the operating system not to try to cache the blocks it reads from disk. Without this hint, the operating system would prefer to cache the blocks, assuming that `xtrabackup` is likely to need them again, which is not the case. Caching such large files can place pressure on the operating system's virtual memory and cause other processes, such as the database server, to be swapped out. The `xtrabackup` tool avoids this with the following hint on both the source and destination files:

```
posix_fadvise(file, 0, 0, POSIX_FADV_DONTNEED)
```

In addition, xtrabackup asks the operating system to perform more aggressive read-ahead optimizations on the source files:

```
posix_fadvise(file, 0, 0, POSIX_FADV_SEQUENTIAL)
```

Copying Data Files

When copying the data files to the target directory, **xtrabackup** reads and writes 1 MB of data at a time. This is not configurable. When copying the log file, **xtrabackup** reads and writes 512 bytes at a time. This is also not possible to configure, and matches InnoDB's behavior (workaround exists in *Percona Server for MySQL* because it has an option to tune `innodb_log_block_size` for *XtraDB*, and in that case *Percona XtraBackup* will match the tuning).

After reading from the files, `xtrabackup` iterates over the 1MB buffer a page at a time, and checks for page corruption on each page with InnoDB's `buf_page_is_corrupted()` function. If the page is corrupt, it re-reads and retries up to 10 times for each page. It skips this check on the doublewrite buffer.

`xtrabackup` Exit Codes

The **xtrabackup** binary exits with the traditional success value of 0 after a backup when no error occurs. If an error occurs during the backup, the exit value is 1.

In certain cases, the exit value can be something other than 0 or 1, due to the command-line option code included from the *MySQL* libraries. An unknown command-line option, for example, will cause an exit code of 255.

12.1.4 References

The `xtrabackup` Option Reference

This page documents all of the command-line options for the **xtrabackup** binary.

Modes of operation

You invoke **xtrabackup** in one of the following modes:

- *--backup* mode to make a backup in a target directory
- *--prepare* mode to restore data from a backup (created in *--backup* mode)
- *--copy-back* to copy data from a backup to the location that contained the original data; to move data instead of copying use the alternate *--move-back* mode.
- *--stats* mode to scan the specified data files and print out index statistics.

When you intend to run **xtrabackup** in any of these modes, use the following syntax:

```
$ xtrabackup [--defaults-file=#] --backup|--prepare|--copy-back|--stats [OPTIONS]
```

For example, the *--prepare* mode is applied as follows:

```
$ xtrabackup --prepare --target-dir=/data/backup/mysql/
```

For all modes, the default options are read from the **xtrabackup** and **mysqld** configuration groups from the following files in the given order:

1. `/etc/my.cnf`
2. `/etc/mysql/my.cnf`
3. `/usr/etc/my.cnf`
4. `~/.my.cnf`.

As the first parameter to **xtrabackup** (in place of the *--defaults-file*, you may supply one of the following:

- *--print-defaults* to have **xtrabackup** print the argument list and exit.
- *--no-defaults* to forbid reading options from any file but the login file.
- *--defaults-file* to read the default options from the given file.
- *--defaults-extra-file* to read the specified additional file after the global files have been read.
- *--defaults-group-suffix* to read the configuration groups with the given suffix. The effective group name is constructed by concatenating the default configuration groups (**xtrabackup** and **mysqld**) with the given suffix.
- *--login-path* to read the given path from the login file.

InnoDB Options

There is a large group of InnoDB options that are normally read from the `my.cnf` configuration file, so that **xtrabackup** boots up its embedded InnoDB in the same configuration as your current server. You normally do not need to specify them explicitly. These options have the same behavior in InnoDB and XtraDB. See *--innodb-miscellaneous* for more information.

Options

--apply-log-only
This option causes only the redo stage to be performed when preparing a backup. It is very important for incremental backups.

--backup
Make a backup and place it in *--target-dir*. See *Creating a backup*.

--backup-lock-timeout
The timeout in seconds for attempts to acquire metadata locks.

--backup-lock-retry-count
The number of attempts to acquire metadata locks.

--backup-locks
This option controls if backup locks should be used instead of `FLUSH TABLES WITH READ LOCK` on the backup stage. The option has no effect when backup locks are not supported by the server. This option is enabled by default, disable with *--no-backup-locks*.

--check-privileges
This option checks if *Percona XtraBackup* has all required privileges. If a missing privilege is required for the current operation, it will terminate and print out an error message. If a missing privilege is not required for the current operation, but may be necessary for some other XtraBackup operation, the process is not aborted and a warning is printed.

```
xtrabackup: Error: missing required privilege LOCK TABLES on *.*
xtrabackup: Warning: missing required privilege REPLICATION CLIENT on *.*
```

--close-files
Do not keep files opened. When **xtrabackup** opens tablespace it normally doesn't close its file handle in order to handle the DDL operations correctly. However, if the number of tablespaces is really huge and can not fit into any limit, there is an option to close file handles once they are no longer accessed. *Percona XtraBackup* can produce inconsistent backups with this option enabled. Use at your own risk.

--compress
This option tells **xtrabackup** to compress all output data, including the transaction log file and meta data files, using either the `quicklz` or `lz4` compression algorithm. `quicklz` is chosen by default.

When using `--compress=quicklz` or `--compress`, the resulting files have the qpress archive format, i.e. every `*.qp` file produced by **xtrabackup** is essentially a one-file qpress archive and can be extracted and uncompressed by the qpress file archiver.

`--compress=lz4` produces `*.lz4` files. You can extract the contents of these files by using a program such as `lz4`.

See also:

QuickLZ http://www.quicklz.com

LZ4 https://lz4.github.io/lz4/

--compress-chunk-size=#
Size of working buffer(s) for compression threads in bytes. The default value is 64K.

--compress-threads=#
This option specifies the number of worker threads used by **xtrabackup** for parallel data compression. This option defaults to 1. Parallel compression (*--compress-threads*) can be used together with parallel file copying (*--parallel*). For example, --parallel=4 --compress --compress-threads=2 will create 4 I/O threads that will read the data and pipe it to 2 compression threads.

--copy-back
Copy all the files in a previously made backup from the backup directory to their original locations. This option will not copy over existing files unless *--force-non-empty-directories* option is specified.

--core-file
Write core on fatal signals.

--databases=#
This option specifies a list of databases and tables that should be backed up. The option accepts the list of the form "databasename1[.table_name1] databasename2[.table_name2] . . .".

--databases-exclude=name
Excluding databases based on name, Operates the same way as *--databases*, but matched names are excluded from backup. Note that this option has a higher priority than *--databases*.

--databases-file=#
This option specifies the path to the file containing the list of databases and tables that should be backed up. The file can contain the list elements of the form databasename1[.table_name1], one element per line.

--datadir=DIRECTORY
The source directory for the backup. This should be the same as the datadir for your *MySQL* server, so it should be read from my.cnf if that exists; otherwise you must specify it on the command line.

When combined with the *--copy-back* or *--move-back* option, *--datadir* refers to the destination directory.

Once connected to the server, in order to perform a backup you will need READ and EXECUTE permissions at a filesystem level in the server's *datadir*.

--debug-sleep-before-unlock=#
This is a debug-only option used by the **xtrabackup** test suite.

--debug-sync=name
The debug sync point. This option is only used by the **xtrabackup** test suite.

--decompress
Decompresses all files with the .qp extension in a backup previously made with the *--compress* option. The *--parallel* option will allow multiple files to be decrypted simultaneously. In order to decompress, the qpress utility MUST be installed and accessible within the path. *Percona XtraBackup* does not automatically remove the compressed files. In order to clean up the backup directory users should use *--remove-original* option.

The *--decompress* option may be used with *xbstream* to decompress individual qpress files.

If you used the lz4 compression algorithm to compress the files (--compress=lz4), change the *--decompress* parameter accordingly: --decompress=lz4.

--decompress-threads=#
Force *xbstream* to use the specified number of threads for decompressing.

--decrypt=ENCRYPTION-ALGORITHM
Decrypts all files with the .xbcrypt extension in a backup previously made with *--encrypt* option. The *--parallel* option will allow multiple files to be decrypted simultaneously. *Percona XtraBackup*

doesn't automatically remove the encrypted files. In order to clean up the backup directory users should use *--remove-original* option.

--defaults-extra-file=[MY.CNF]
Read this file after the global files are read. Must be given as the first option on the command-line.

--defaults-file=[MY.CNF]
Only read default options from the given file. Must be given as the first option on the command-line. Must be a real file; it cannot be a symbolic link.

--defaults-group=GROUP-NAME
This option is to set the group which should be read from the configuration file. This is used by **xtrabackup** if you use the *--defaults-group* option. It is needed for `mysqld_multi` deployments.

--defaults-group-suffix=#
Also reads groups with concat(group, suffix).

--dump-innodb-buffer-pool
This option controls whether or not a new dump of buffer pool content should be done.

With `--dump-innodb-buffer-pool`, **xtrabackup** makes a request to the server to start the buffer pool dump (it takes some time to complete and is done in background) at the beginning of a backup provided the status variable `innodb_buffer_pool_dump_status` reports that the dump has been completed.

```
$ xtrabackup --backup --dump-innodb-buffer-pool --target-dir=/home/user/backup
```

By default, this option is set to *OFF*.

If `innodb_buffer_pool_dump_status` reports that there is running dump of buffer pool, **xtrabackup** waits for the dump to complete using the value of *--dump-innodb-buffer-pool-timeout*

The file `ib_buffer_pool` stores tablespace ID and page ID data used to warm up the buffer pool sooner.

See also:

***MySQL* Documentation: Saving and Restoring the Buffer Pool State** https://dev.mysql.com/doc/refman/5.7/en/innodb-preload-buffer-pool.html

--dump-innodb-buffer-pool-timeout
This option contains the number of seconds that **xtrabackup** should monitor the value of `innodb_buffer_pool_dump_status` to determine if buffer pool dump has completed.

This option is used in combination with *--dump-innodb-buffer-pool*. By default, it is set to *10* seconds.

--dump-innodb-buffer-pool-pct
This option contains the percentage of the most recently used buffer pool pages to dump.

This option is effective if *--dump-innodb-buffer-pool* option is set to *ON*. If this option contains a value, **xtrabackup** sets the *MySQL* system variable `innodb_buffer_pool_dump_pct`. As soon as the buffer pool dump completes or it is stopped (see *--dump-innodb-buffer-pool-timeout*), the value of the *MySQL* system variable is restored.

See also:

Changing the timeout for buffer pool dump *--dump-innodb-buffer-pool-timeout*

***MySQL* Documentation: innodb_buffer_pool_dump_pct system variable** https://dev.mysql.com/doc/refman/8.0/en/innodb-parameters.html#sysvar_innodb_buffer_pool_dump_pct

--encrypt=ENCRYPTION_ALGORITHM
This option instructs xtrabackup to encrypt backup copies of InnoDB data files using the algorithm specified in the ENCRYPTION_ALGORITHM. Currently supported algorithms are: `AES128`, `AES192` and `AES256`

--encrypt-key=ENCRYPTION_KEY
A proper length encryption key to use. It is not recommended to use this option where there is uncontrolled access to the machine as the command line and thus the key can be viewed as part of the process info.

--encrypt-key-file=ENCRYPTION_KEY_FILE
The name of a file where the raw key of the appropriate length can be read from. The file must be a simple binary (or text) file that contains exactly the key to be used.

It is passed directly to the xtrabackup child process. See the **xtrabackup** *documentation* for more details.

--encrypt-threads=#
This option specifies the number of worker threads that will be used for parallel encryption/decryption. See the **xtrabackup** *documentation* for more details.

--encrypt-chunk-size=#
This option specifies the size of the internal working buffer for each encryption thread, measured in bytes. It is passed directly to the xtrabackup child process. See the **xtrabackup** *documentation* for more details.

--export
Create files necessary for exporting tables. See *Restoring Individual Tables*.

--extra-lsndir=DIRECTORY
(for –backup): save an extra copy of the `xtrabackup_checkpoints` and `xtrabackup_info` files in this directory.

--force-non-empty-directories
When specified, it makes *--copy-back* and *--move-back* option transfer files to non-empty directories. No existing files will be overwritten. If files that need to be copied/moved from the backup directory already exist in the destination directory, it will still fail with an error.

--ftwrl-wait-timeout=SECONDS
This option specifies time in seconds that xtrabackup should wait for queries that would block `FLUSH TABLES WITH READ LOCK` before running it. If there are still such queries when the timeout expires, xtrabackup terminates with an error. Default is `0`, in which case it does not wait for queries to complete and starts `FLUSH TABLES WITH READ LOCK` immediately. Where supported **xtrabackup** will automatically use Backup Locks as a lightweight alternative to `FLUSH TABLES WITH READ LOCK` to copy non-InnoDB data to avoid blocking DML queries that modify InnoDB tables.

--ftwrl-wait-threshold=SECONDS
This option specifies the query run time threshold which is used by xtrabackup to detect long-running queries with a non-zero value of *--ftwrl-wait-timeout*. `FLUSH TABLES WITH READ LOCK` is not started until such long-running queries exist. This option has no effect if *--ftwrl-wait-timeout* is `0`. Default value is `60` seconds. Where supported xtrabackup will automatically use Backup Locks as a lightweight alternative to `FLUSH TABLES WITH READ LOCK` to copy non-InnoDB data to avoid blocking DML queries that modify InnoDB tables.

--ftwrl-wait-query-type=`all|update`
This option specifies which types of queries are allowed to complete before xtrabackup will issue the global lock. Default is `all`.

--galera-info
This option creates the `xtrabackup_galera_info` file which contains the local node state at the time of the backup. Option should be used when performing the backup of *Percona XtraDB Cluster*. It has no effect when backup locks are used to create the backup.

--generate-new-master-key
Generate a new master key when doing a copy-back.

--generate-transition-key
xtrabackup needs to access the same keyring file or vault server during *prepare* and *copy-back* but it should not depend on whether the server keys have been purged.

--generate-transition-key creates and adds to the keyring a transition key for **xtrabackup** to use if the master key used for encryption is not found because it has been rotated and purged.

--get-server-public-key
Get the server public key

See also:

MySQL Documentation: The –get-server-public-key Option

https://dev.mysql.com/doc/refman/5.7/en/connection-options.html#option_general_get-server-public-key

--help
When run with this option or without any options **xtrabackup** displays information about how to run the program on the command line along with all supported options and variables with default values where appropriate.

--history=NAME
This option enables the tracking of backup history in the `PERCONA_SCHEMA.xtrabackup_history` table. An optional history series name may be specified that will be placed with the history record for the current backup being taken.

--host=HOST
This option accepts a string argument that specifies the host to use when connecting to the database server with TCP/IP. It is passed to the mysql child process without alteration. See **mysql --help** for details.

--incremental
This option tells **xtrabackup** to create an incremental backup. It is passed to the **xtrabackup** child process. When this option is specified, either *--incremental-lsn* or *--incremental-basedir* can also be given. If neither option is given, option *--incremental-basedir* is passed to **xtrabackup** by default, set to the first timestamped backup directory in the backup base directory.

See also:

More information about incremental backups See section *Incremental Backups*

--incremental-basedir=DIRECTORY
When creating an incremental backup, this is the directory containing the full backup that is the base dataset for the incremental backups.

--incremental-dir=DIRECTORY
When preparing an incremental backup, this is the directory where the incremental backup is combined with the full backup to make a new full backup.

--incremental-force-scan
When creating an incremental backup, force a full scan of the data pages in the instance being backuped even if the complete changed page bitmap data is available.

--incremental-history-name=name
This option specifies the name of the backup series stored in the `PERCONA_SCHEMA.xtrabackup_history` history record to base an incremental backup on. **xtrabackup** will search the history table looking for the most recent (highest `innodb_to_lsn`), successful backup in the series and take the to_lsn value to use as the starting `lsn` for the incremental backup. This will be mutually exclusive with *--incremental-history-uuid*, *--incremental-basedir* and *--incremental-lsn*. If no

valid lsn can be found (no series by that name, no successful backups by that name) **xtrabackup** will return with an error. It is used with the *--incremental* option.

--incremental-history-uuid=name
This option specifies the *UUID* of the specific history record stored in the PERCONA_SCHEMA.xtrabackup_history to base an incremental backup on. *--incremental-history-name*, *--incremental-basedir* and *--incremental-lsn*. If no valid lsn can be found (no success record with that *UUID*) **xtrabackup** will return with an error. It is used with the –incremental option.

--incremental-lsn=LSN
When creating an incremental backup, you can specify the log sequence number (*LSN*) instead of specifying *--incremental-basedir*. For databases created in 5.1 and later, specify the *LSN* as a single 64-bit integer. **ATTENTION**: If a wrong LSN value is specified (a user error which *Percona XtraBackup* is unable to detect), the backup will be unusable. Be careful!

--innodb[=name]
This option is ignored for MySQL option compatibility.

--innodb-miscellaneous
There is a large group of InnoDB options that are normally read from the my.cnf configuration file, so that **xtrabackup** boots up its embedded InnoDB in the same configuration as your current server. You normally do not need to specify these explicitly. These options have the same behavior in InnoDB and XtraDB:

- –innodb-adaptive-hash-index
- –innodb-additional-mem-pool-size
- –innodb-autoextend-increment
- –innodb-buffer-pool-size
- –innodb-buffer-pool-filename
- –innodb-checksum-algorithm
- –innodb-checksums
- –innodb-data-file-path
- –innodb-data-home-dir
- –innodb-directories
- –innodb-doublewrite-file
- –innodb-doublewrite
- –innodb-extra-undoslots
- –innodb-fast-checksum
- –innodb-file-io-threads
- –innodb-file-per-table
- –innodb-flush-log-at-trx-commit
- –innodb-flush-method
- –innodb-io-capacity
- –innodb-lock-wait-timeout
- –innodb-log-block-size
- –innodb-log-buffer-size
- –innodb-log-checksums
- –innodb-log-files-in-group
- –innodb-log-file-size
- –innodb-log-group-home-dir
- –innodb-max-dirty-pages-pct
- –innodb-open-files
- –innodb-page-size
- –innodb-read-io-threads
- –innodb-redo-log-encrypt
- –innodb-undo-directory
- –innodb-undo-log-encrypt

- –innodb-undo-tablespaces`
- –innodb-use-native-aio
- –innodb-write-io-threads

--keyring-file-data=FILENAME
The path to the keyring file. Combine this option with *--xtrabackup-plugin-dir*.

--kill-long-queries-timeout=SECONDS
This option specifies the number of seconds **xtrabackup** waits between starting FLUSH TABLES WITH READ LOCK and killing those queries that block it. Default is 0 seconds, which means **xtrabackup** will not attempt to kill any queries. In order to use this option xtrabackup user should have the PROCESS and SUPER privileges. Where supported, **xtrabackup** automatically uses Backup Locks as a lightweight alternative to FLUSH TABLES WITH READ LOCK to copy non-InnoDB data to avoid blocking DML queries that modify InnoDB tables.

--kill-long-query-type=all|select
This option specifies which types of queries should be killed to unblock the global lock. Default is "select".

--lock-ddl
Issue LOCK TABLES FOR BACKUP if it is supported by server (otherwise use LOCK INSTANCE FOR BACKUP) at the beginning of the backup to block all DDL operations.

Note: Prior to *Percona XtraBackup* 8.0.22-15.0, using a *safe-slave-backup* stops the SQL replica thread after the InnoDB tables and before the non-InnoDB tables are backed up.

As of *Percona XtraBackup* 8.0.22-15.0, using a *safe-slave-backup* option stops the SQL replica thread before copying the InnoDB files.

--lock-ddl-per-table
Lock DDL for each table before xtrabackup starts to copy it and until the backup is completed.

Note: As of *Percona XtraBackup* 8.0.15, the *–lock-ddl-per-table* option is deprecated. Use the *–lock-ddl* option instead.

--lock-ddl-timeout
If LOCK TABLES FOR BACKUP or LOCK INSTANCE FOR BACKUP does not return within given timeout, abort the backup.

--log
This option is ignored for *MySQL*

--log-bin
The base name for the log sequence.

--log-bin-index=name
File that holds the names for binary log files.

--log-copy-interval=#
This option specifies the time interval between checks done by the log copying thread in milliseconds (default is 1 second).

--login-path
Read the given path from the login file.

--move-back
Move all the files in a previously made backup from the backup directory to their original locations. As this option removes backup files, it must be used with caution.

--no-backup-locks
Explicity disables the *--backup-locks* option which is enabled by default.

--no-defaults
The default options are only read from the login file.

--no-lock
Use this option to disable table lock with `FLUSH TABLES WITH READ LOCK`. Use it only if ALL your tables are InnoDB and you **DO NOT CARE** about the binary log position of the backup. This option shouldn't be used if there are any `DDL` statements being executed or if any updates are happening on non-InnoDB tables (this includes the system MyISAM tables in the *mysql* database), otherwise it could lead to an inconsistent backup. Where supported **xtrabackup** will automatically use Backup Locks as a lightweight alternative to `FLUSH TABLES WITH READ LOCK` to copy non-InnoDB data to avoid blocking DML queries that modify InnoDB tables. If you are considering to use this because your backups are failing to acquire the lock, this could be because of incoming replication events are preventing the lock from succeeding. Please try using *--safe-slave-backup* to momentarily stop the replication replica thread, this may help the backup to succeed and you do not need to use this option.

--no-server-version-check
Implemented in *Percona XtraBackup* 8.0.21.

The `--no-server-version-check` option disables the server version check.

The default behavior runs a check that compares the source system version to the *Percona XtraBackup* version. If the source system version is higher than the XtraBackup version, the backup is aborted with a message.

Adding the option overrides this check, and the backup proceeds, but there may be issues with the backup.

See *Server Version and Backup Version Comparison* for more information.

--no-version-check
This option disables the version check. If you do not pass this option, the automatic version check is enabled implicitly when **xtrabackup** runs in the `--backup` mode. To disable the version check, you should pass explicitly the `--no-version-check` option when invoking **xtrabackup**.

When the automatic version check is enabled, **xtrabackup** performs a version check against the server on the backup stage after creating a server connection. **xtrabackup** sends the following information to the server:

- MySQL flavour and version
- Operating system name
- Percona Toolkit version
- Perl version

Each piece of information has a unique identifier. This is a MD5 hash value that Percona Toolkit uses to obtain statistics about how it is used. This is a random UUID; no client information is either collected or stored.

--open-files-limit=#
The maximum number of file descriptors to reserve with setrlimit().

--parallel=#
This option specifies the number of threads to use to copy multiple data files concurrently when creating a backup. The default value is 1 (i.e., no concurrent transfer). In *Percona XtraBackup* 2.3.10 and newer, this option can be used with the *--copy-back* option to copy the user data files in parallel (redo logs and system tablespaces are copied in the main thread).

--password=PASSWORD
This option specifies the password to use when connecting to the database. It accepts a string argument. See **mysql --help** for details.

--plugin-load
List of plugins to load.

--port=PORT
This option accepts a string argument that specifies the port to use when connecting to the database server with TCP/IP. It is passed to the **mysql** child process without alteration. See **mysql --help** for details.

--prepare
Makes **xtrabackup** perform a recovery on a backup created with *--backup*, so that it is ready to use. See *preparing a backup*.

--print-defaults
Print the program argument list and exit. Must be given as the first option on the command-line.

--print-param
Makes **xtrabackup** print out parameters that can be used for copying the data files back to their original locations to restore them.

--read-buffer-size
Set the datafile read buffer size, given value is scaled up to page size. Default is 10Mb.

--rebuild-indexes
Rebuilds indexes in a compact backup. This option only has effect when the *--prepare* and *--rebuild-threads* options are provided.

--rebuild-threads=#
Uses the given number of threads to rebuild indexes in a compact backup. This option only has effect with the *--prepare* and *--rebuild-indexes* options.

--remove-original
Implemented in *Percona XtraBackup* 2.4.6, this option when specified will remove `.qp`, `.xbcrypt` and `.qp.xbcrypt` files after decryption and decompression.

--rocksdb-datadir
RocksDB data directory

--rocksdb-wal-dir
RocksDB WAL directory.

--rocksdb-checkpoint-max-age
The checkpoint cannot be older than this number of seconds when the backup completes.

--rocksdb-checkpoint-max-count
Complete the backup even if the checkpoint age requirement has not been met after this number of checkpoints.

--rollback-prepared-trx
Force rollback prepared InnoDB transactions.

--rsync
Uses the **rsync** utility to optimize local file transfers. When this option is specified, **xtrabackup** uses **rsync** to copy all non-InnoDB files instead of spawning a separate **cp** for each file, which can be much faster for servers with a large number of databases or tables. This option cannot be used together with *--stream*.

--safe-slave-backup
When specified, xtrabackup will stop the replica SQL thread just before running `FLUSH TABLES WITH READ LOCK` and wait to start backup until `Slave_open_temp_tables` in `SHOW STATUS` is zero. If there are no open temporary tables, the backup will take place, otherwise the SQL thread will be started and stopped until there are no open temporary tables. The backup will fail if `Slave_open_temp_tables` does not become zero after *--safe-slave-backup-timeout* seconds. The replication SQL thread will be restarted when the backup finishes. This option is implemented in order to deal with replicating temporary tables and isn't neccessary with Row-Based-Replication.

--safe-slave-backup-timeout=SECONDS
How many seconds *--safe-slave-backup* should wait for `Slave_open_temp_tables` to become zero. Defaults to 300 seconds.

--secure-auth
Refuse client connecting to server if it uses old (pre-4.1.1) protocol. (Enabled by default; use –skip-secure-auth to disable.)

--server-id=#
The server instance being backed up.

--server-public-key-path
The file path to the server public RSA key in the PEM format.

See also:

***MySQL* Documentation: The –server-public-key-path Option** https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#option_general_server-public-key-path

--skip-tables-compatibility-check
See *--tables-compatibility-check*.

--slave-info
This option is useful when backing up a replication replica server. It prints the binary log position of the source server. It also writes the binary log coordinates to the `xtrabackup_slave_info` file as a `CHANGE MASTER` command. A new replica for this source can be set up by starting a replica server on this backup and issuing a `CHANGE MASTER` command with the binary log position saved in the `xtrabackup_slave_info` file.

--socket
This option accepts a string argument that specifies the socket to use when connecting to the local database server with a UNIX domain socket. It is passed to the mysql child process without alteration. See **mysql --help** for details.

--ssl
Enable secure connection. More information can be found in –ssl MySQL server documentation.

--ssl-ca
Path of the file which contains list of trusted SSL CAs. More information can be found in –ssl-ca MySQL server documentation.

--ssl-capath
Directory path that contains trusted SSL CA certificates in PEM format. More information can be found in –ssl-capath MySQL server documentation.

--ssl-cert
Path of the file which contains X509 certificate in PEM format. More information can be found in –ssl-cert MySQL server documentation.

--ssl-cipher
List of permitted ciphers to use for connection encryption. More information can be found in –ssl-cipher MySQL server documentation.

--ssl-crl
Path of the file that contains certificate revocation lists. More information can be found in –ssl-crl MySQL server documentation.

--ssl-crlpath
Path of directory that contains certificate revocation list files. More information can be found in –ssl-crlpath MySQL server documentation.

--ssl-fips-mode
SSL FIPS mode (applies only for OpenSSL); permitted values are: *OFF*, *ON*, *STRICT*.

--ssl-key
Path of file that contains X509 key in PEM format. More information can be found in –ssl-key MySQL server documentation.

--ssl-mode
Security state of connection to server. More information can be found in –ssl-mode MySQL server documentation.

--ssl-verify-server-cert
Verify server certificate Common Name value against host name used when connecting to server. More information can be found in –ssl-verify-server-cert MySQL server documentation.

--stats
Causes **xtrabackup** to scan the specified data files and print out index statistics.

--stream=FORMAT
Stream all backup files to the standard output in the specified format. Currently, this option only supports the *xbstream* format.

--strict
If this option is specified, **xtrabackup** fails with an error when invalid parameters are passed.

--tables=name
A regular expression against which the full tablename, in `databasename.tablename` format, is matched. If the name matches, the table is backed up. See *partial backups*.

--tables-compatibility-check
Enables the engine compatibility warning. The default value is ON. To disable the engine compatibility warning use *--skip-tables-compatibility-check*.

--tables-exclude=name
Filtering by regexp for table names. Operates the same way as *--tables*, but matched names are excluded from backup. Note that this option has a higher priority than *--tables*.

--tables-file=name
A file containing one table name per line, in databasename.tablename format. The backup will be limited to the specified tables.

--target-dir=DIRECTORY
This option specifies the destination directory for the backup. If the directory does not exist, **xtrabackup** creates it. If the directory does exist and is empty, **xtrabackup** will succeed. **xtrabackup** will not overwrite existing files, however; it will fail with operating system error 17, `file exists`.

If this option is a relative path, it is interpreted as being relative to the current working directory from which **xtrabackup** is executed.

In order to perform a backup, you need `READ`, `WRITE`, and `EXECUTE` permissions at a filesystem level for the directory that you supply as the value of *--target-dir*.

--innodb-temp-tablespaces-dir=DIRECTORY
Directory where temp tablespace files live, this path can be absolute.

--throttle=#
This option limits the number of chunks copied per second. The chunk size is *10 MB*. To limit the bandwidth to *10 MB/s*, set the option to *1*: *–throttle=1*.

See also:

More information about how to throttle a backup *Throttling Backups*

--tls-ciphersuites
TLS v1.3 cipher to use.

--tls-version
TLS version to use, permitted values are: *TLSv1*, *TLSv1.1*, *TLSv1.2*, *TLSv1.3*.

--tmpdir=name
Specify the directory that will be used to store temporary files during the backup

--transition-key=name
This option is used to enable processing the backup without accessing the keyring vault server. In this case, **xtrabackup** derives the AES encryption key from the specified passphrase and uses it to encrypt tablespace keys of tablespaces being backed up.

If *--transition-key* does not have any value, **xtrabackup** will ask for it. The same passphrase should be specified for the *--prepare* command.

--use-memory
This option affects how much memory is allocated for preparing a backup with *--prepare*, or analyzing statistics with *--stats*. Its purpose is similar to *innodb_buffer_pool_size*. It does not do the same thing as the similarly named option in Oracle's InnoDB Hot Backup tool. The default value is 100MB, and if you have enough available memory, 1GB to 2GB is a good recommended value. Multiples are supported providing the unit (e.g. 1MB, 1M, 1GB, 1G).

--user=USERNAME
This option specifies the MySQL username used when connecting to the server, if that's not the current user. The option accepts a string argument. See mysql –help for details.

-v
See *--version*

--version
This option prints **xtrabackup** version and exits.

--xtrabackup-plugin-dir=DIRNAME
The absolute path to the directory that contains the `keyring` plugin.

See also:

***Percona Server for MySQL* Documentation: keyring_vault plugin with Data at Rest Encryption**
https://www.percona.com/doc/percona-server/LATEST/management/data_at_rest_encryption.html#keyring-vault-plugin

***MySQL* Documentation: Using the keyring_file File-Based Plugin** https://dev.mysql.com/doc/refman/5.7/en/keyring-file-plugin.html

12.2 The xbstream binary

To support simultaneous compression and streaming, a new custom streaming format called xbstream was introduced to *Percona XtraBackup* in addition to the TAR format. That was required to overcome some limitations of traditional archive formats such as tar, cpio and others which did not allow streaming dynamically generated files, for example dynamically compressed files. Other advantages of xbstream over traditional streaming/archive format include ability to stream multiple files concurrently (so it is possible to use streaming in the xbstream format together with the –parallel option) and more compact data storage.

This utility has a tar-like interface:

- with the `-x` option it extracts files from the stream read from its standard input to the current directory unless specified otherwise with the `-c` option. Support for parallel extraction with the `--parallel` option has been implemented in *Percona XtraBackup* 2.4.7.
- with the `-c` option it streams files specified on the command line to its standard output.
- with the `--decrypt=ALGO` option specified xbstream will automatically decrypt encrypted files when extracting input stream. Supported values for this option are: `AES128`, `AES192`, and `AES256`. Either `--encrypt-key` or `--encrypt-key-file` options must be specified to provide encryption key, but not both. This option has been implemented in *Percona XtraBackup* 2.4.7.
- with the `--encrypt-threads` option you can specify the number of threads for parallel data encryption. The default value is `1`. This option has been implemented in *Percona XtraBackup* 2.4.7.
- the `--encrypt-key` option is used to specify the encryption key that will be used. It can't be used with `--encrypt-key-file` option because they are mutually exclusive. This option has been implemented in *Percona XtraBackup* 2.4.7.
- the `--encrypt-key-file` option is used to specify the file that contains the encryption key. It can't be used with `--encrypt-key` option. because they are mutually exclusive. This option has been implemented in *Percona XtraBackup* 2.4.7.

The utility also tries to minimize its impact on the OS page cache by using the appropriate `posix_fadvise()` calls when available.

When compression is enabled with **`xtrabackup`** all data is being compressed, including the transaction log file and meta data files, using the specified compression algorithm. The only currently supported algorithm is `quicklz`.

The resulting files have the qpress archive format, i.e., every `*.qp` file produced by **`xtrabackup`** is essentially a one-file qpress archive and can be extracted and uncompressed by the qpress file archiver. This means that there is no need to decompress entire backup to restore a single table as with `tar.gz`.

To decompress individual files, run *xbstream* with the *`--decompress`* option. You may control the number of threads used for decompressing by passing the *`--decompress-threads`* option.

Also, files can be decompressed using the **qpress** tool that can be downloaded from here. Qpress supports multi-threaded decompression.

12.3 The xbcrypt binary

To support encryption and decryption of the backups, a new tool `xbcrypt` was introduced to *Percona XtraBackup*.

This utility has been modeled after *The xbstream binary* to perform encryption and decryption outside of *Percona XtraBackup*. `xbcrypt` has following command line options:

`-d, --decrypt`
Decrypt data input to output.

`-i, --input``=name`
Optional input file. If not specified, input will be read from standard input.

`-o, --output``=name`
Optional output file. If not specified, output will be written to standard output.

`-a, --encrypt-algo``=name`
Encryption algorithm.

`-k, --encrypt-key``=name`
Encryption key.

-f, --encrypt-key-file=name
File which contains encryption key.

-s, --encrypt-chunk-size=#
Size of working buffer for encryption in bytes. The default value is 64K.

--encrypt-threads=#
This option specifies the number of worker threads that will be used for parallel encryption/decryption.

-v, --verbose
Display verbose status output.

12.4 The xbcloud Binary

The purpose of *xbcloud* is to download from the cloud and upload to the cloud the full or part of an *xbstream* archive. *xbcloud* will not overwrite the backup with the same name. *xbcloud* accepts input via a pipe from *xbstream* so that it can be invoked as a pipeline with **xtrabackup** to stream directly to the cloud without needing a local storage.

xbcloud stores each chunk as a separate object with a name `backup_name/database/table.ibd.NNNNNNNNNNNNNNNNNNNN`, where `NNN...` is a 0-padded serial number of chunk within file. Size of chunk produced by **xtrabackup** and *xbstream* changed to 10M.

xbcloud has three essential operations: *put*, *get*, and *delete*. With these operations, backups are created, stored, retrieved, restored, and deleted. *xbcloud* operations clearly map to similar operations within the AWS Amazon Amazon S3 API.

The *Exponential Backoff* feature was implemented in Percona XtraBackup 8.0.26-18. Suppose a chunk fails to upload or download. In that case, this feature adds an exponential backoff, or sleep, time and then retries the upload or download, which increases the chances of completing a backup or a restore operation.

12.4.1 Supported Cloud Storage Types

The following cloud storage types are supported:

- OpenStack Object Storage (Swift) - see *Using the xbcloud Binary with Swift*
- Amazon Simple Storage (S3) - see *Using xbcloud Binary with Amazon S3*
- Azure Cloud Storage - see xbcloud_azure
- Google Cloud Storage (gcs) - see *Using the xbcloud with Google Cloud Storage*
- MinIO - see *Using the xbcloud Binary with MinIO*

In addition to OpenStack Object Storage (Swift), which has been the only option for storing backups in a cloud storage until Percona XtraBackup 2.4.14, *xbcloud* supports Amazon S3, MinIO, and Google Cloud Storage. Other Amazon S3-compatible storages, such as Wasabi or Digital Ocean Spaces, are also supported.

See also:

OpenStack Object Storage ("Swift") https://wiki.openstack.org/wiki/Swift

Amazon Simple Storage Service https://aws.amazon.com/s3/

MinIO https://min.io/

Google Cloud Storage https://cloud.google.com/storage/

Wasabi https://wasabi.com/

Digital Ocean Spaces https://www.digitalocean.com/products/spaces/

12.4.2 Usage

The following sample command creates a full backup:

```
xtrabackup --backup --stream=xbstream --target-dir=/storage/backups/ --extra-lsndirk=/
↪storage/backups/| xbcloud \
put [options] full_backup
```

An incremental backup only includes the changes since the last backup. The last backup can be either a full or incremental backup.

The following sample command creates an incremental backup:

```
xtrabackup --backup --stream=xbstream --incremental-basedir=/storage/backups \
--target-dir=/storage/inc-backup | xbcloud  put [options] inc_backup
```

To prepare an incremental backup, you must first download the full backup with the following command:

```
xtrabackup get [options] full_backup | xbstream -xv -C /tmp/full-backup
```

You must prepare the full backup:

```
xtrabackup --prepare --apply-log-only --target-dir=/tmp/full-backup
```

After the full backup has been prepared, download the incremental backup:

```
xbcloud get [options] inc_backup | xbstream -xv -C /tmp/inc-backup
```

The downloaded backup is prepared by running the following command:

```
xtrabackup --prepare --target-dir=/tmp/full-backup --incremental-dir=/tmp/inc-backup
```

You do not need the full backup to restore only a specific database. You can specify only the tables to be restored:

```
xbcloud get [options] ibdata1 sakila/payment.ibd /tmp/partial/partial.xbs

xbstream -xv -C /tmp/partial < /tmp/partial/partial.xbs
```

12.4.3 Supplying parameters

Each storage type has mandatory parameters that you can supply on the command line, in a configuration file, and via environment variables.

Configuration files

The parameters the values of which do not change frequently can be stored in `my.cnf` or in a custom configuration file. The following example is a template of configuration options under the `[xbcloud]` group:

```
[xbcloud]
storage=s3
s3-endpoint=http://localhost:9000/
s3-access-key=minio
s3-secret-key=minio123
s3-bucket=backupsx
s3-bucket-lookup=path
s3-api-version=4
```

Note: If you explicitly use a parameter on the command line and in a configuration file, *xbcloud* uses the the value provided on the command line.

Environment variables

If you explicitly use a parameter on the command line, in a configuration file, and the corresponding environment variable contains a value, *xbcloud* uses the the value provided on the command line or in the configuration file.

Shortcuts

For all operations (put, get, and delete), you can use a shortcut to specify the storage type, bucket name, and backup name as one parameter instead of using three distinct parameters (–storage, –s3-bucket, and backup name per se).

Using a shortcut syntax to provide a storage type, bucket, and backup name

Use the following format: `storage-type://bucket-name/backup-name`

```
$ xbcloud get s3://operator-testing/bak22 ...
```

In this example, **s3** refers to a storage type, **operator-testing** is a bucket name, and **bak22** is the backup name. This shortcut expands as follows:

```
$ xbcloud get --storage=s3 --s3-bucket=operator-testing bak22 ...
```

You can supply the mandatory parameters not only on the command line. You may use configuration files and environment variables.

Additional parameters

xbcloud accepts additional parameters that you can use with any storage type. The `--md5` parameter computes the MD5 hash value of the backup chunks. The result is stored in files that following the `backup_name.md5` pattern.

```
$ xtrabackup --backup --stream=xbstream \
--parallel=8 2>backup.log | xbcloud put s3://operator-testing/bak22 \
--parallel=8 --md5 2>upload.log
```

You may use the `--header` parameter to pass an additional HTTP header with the server side encryption while specifying a customer key.

Example of using `--header` for AES256 encryption

```
$ xtrabackup --backup --stream=xbstream --parallel=4 | \
xbcloud put s3://operator-testing/bak-enc/ \
--header="X-Amz-Server-Side-Encryption-Customer-Algorithm: AES256" \
--header="X-Amz-Server-Side-Encryption-Customer-Key: CuStoMerKey=" \
--header="X-Amz-Server-Side-Encryption-Customer-Key-MD5: CuStoMerKeyMd5==" \
--parallel=8
```

The `--header` parameter is also useful to set the access control list (ACL) permissions: `--header="x-amz-acl: bucket-owner-full-control`

12.4.4 Incremental backups

First, you need to make the full backup on which the incremental one is going to be based:

```
xtrabackup --backup --stream=xbstream --extra-lsndir=/storage/backups/ \
--target-dir=/storage/backups/ | xbcloud put \
--storage=swift --swift-container=test_backup \
--swift-auth-version=2.0 --swift-user=admin \
--swift-tenant=admin --swift-password=xoxoxoxo \
--swift-auth-url=http://127.0.0.1:35357/ --parallel=10 \
full_backup
```

Then you can make the incremental backup:

```
$ xtrabackup --backup --incremental-basedir=/storage/backups \
--stream=xbstream --target-dir=/storage/inc_backup | xbcloud put \
--storage=swift --swift-container=test_backup \
--swift-auth-version=2.0 --swift-user=admin \
--swift-tenant=admin --swift-password=xoxoxoxo \
--swift-auth-url=http://127.0.0.1:35357/ --parallel=10 \
inc_backup
```

Preparing incremental backups

To prepare a backup you first need to download the full backup:

```
$ xbcloud get --swift-container=test_backup \
--swift-auth-version=2.0 --swift-user=admin \
--swift-tenant=admin --swift-password=xoxoxoxo \
--swift-auth-url=http://127.0.0.1:35357/ --parallel=10 \
full_backup | xbstream -xv -C /storage/downloaded_full
```

Once you download the full backup it should be prepared:

```
$ xtrabackup --prepare --apply-log-only --target-dir=/storage/downloaded_full
```

After the full backup has been prepared you can download the incremental backup:

```
$ xbcloud get --swift-container=test_backup \
--swift-auth-version=2.0 --swift-user=admin \
--swift-tenant=admin --swift-password=xoxoxoxo \
--swift-auth-url=http://127.0.0.1:35357/ --parallel=10 \
inc_backup | xbstream -xv -C /storage/downloaded_inc
```

Once the incremental backup has been downloaded you can prepare it by running:

```
$ xtrabackup --prepare --apply-log-only \
--target-dir=/storage/downloaded_full \
--incremental-dir=/storage/downloaded_inc

$ xtrabackup --prepare --target-dir=/storage/downloaded_full
```

Partial download of the cloud backup

If you do not want to download the entire backup to restore the specific database you can specify only the tables you want to restore:

```
$ xbcloud get --swift-container=test_backup
--swift-auth-version=2.0 --swift-user=admin \
--swift-tenant=admin --swift-password=xoxoxoxo \
--swift-auth-url=http://127.0.0.1:35357/ full_backup \
ibdata1 sakila/payment.ibd \
> /storage/partial/partial.xbs

$ xbstream -xv -C /storage/partial < /storage/partial/partial.xbs
```

Percona XtraBackup is a set of the following tools:

xtrabackup a compiled *C* binary that provides functionality to backup a whole *MySQL* database instance with *MyISAM*, *InnoDB*, and *XtraDB* tables.

xbcrypt utility used for encrypting and decrypting backup files.

xbstream utility that allows streaming and extracting files to/from the *xbstream* format.

xbcloud utility used for downloading and uploading full or part of *xbstream* archive from/to cloud.

After *Percona XtraBackup* 2.3 release, the recommend way to take the backup is by using the **`xtrabackup`** script. More information on script options can be found in *how to use xtrabackup*.

Part VI

Advanced Features

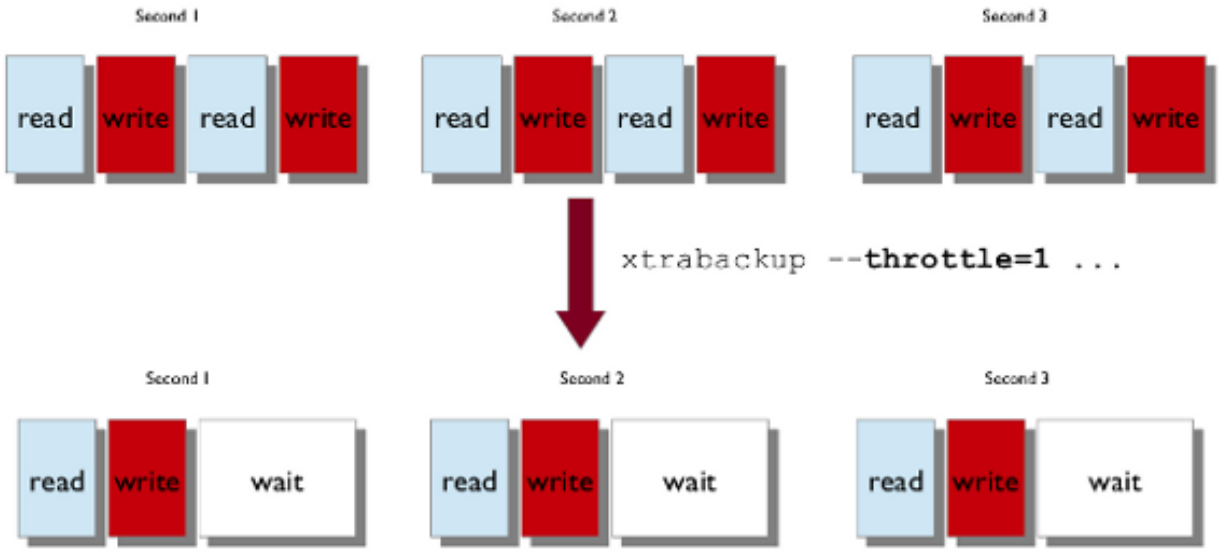
CHAPTER

THIRTEEN

THROTTLING BACKUPS

Although **xtrabackup** does not block your database's operation, any backup can add load to the system being backed up. On systems that do not have much spare I/O capacity, it might be helpful to throttle the rate at which **xtrabackup** reads and writes data. You can do this with the `--throttle` option. This option limits the number of chunks copied per second. The chunk +size is *10 MB*.

The image below shows how throttling works when `--throttle` is set to `1`.



When specified with the `--backup` option, this option limits the number of pairs of read-and-write operations per second that **xtrabackup** will perform. If you are creating an incremental backup, then the limit is the number of read I/O operations per second.

By default, there is no throttling, and **xtrabackup** reads and writes data as quickly as it can. If you set too strict of a limit on the IOPS, the backup might be so slow that it will never catch up with the transaction logs that InnoDB is writing, so the backup might never complete.

CHAPTER

FOURTEEN

ENCRYPTED INNODB TABLESPACE BACKUPS

InnoDB supports data encryption for InnoDB tables stored in file-per-table tablespaces. This feature provides an at-rest encryption for physical tablespace data files.

For an authenticated user or application to access an encrypted tablespace, InnoDB uses the master encryption key to decrypt the tablespace key. The master encryption key is stored in a keyring. **xtrabackup** supports two keyring plugins: `keyring_file`, and `keyring_vault`. These plugins are installed into the `plugin` directory.

Implemented in **xtrabackup** version 8.0.25, adds support for the `keyring_file` component, which is part of the component-based infrastructure MySQL includes to extend the server capabilities. The component is stored in the `plugin` directory.

See a comparison of keyring components and keyring plugins for more information.

Percona XtraBackup 8.0.27-19 adds support for the Key Management Interoperability Protocol (KMIP) which enables the communication between the key management system and encrypted database server.

This feature is *tech preview* quality.

14.1 Making a backup

14.1.1 Using `keyring_file` plugin

In order to backup and prepare a database containing encrypted InnoDB tablespaces, specify the path to a keyring file as the value of the `--keyring-file-data` option.

```
$ xtrabackup --backup --target-dir=/data/backup/ --user=root \
--keyring-file-data=/var/lib/mysql-keyring/keyring
```

After **xtrabackup** takes the backup, the following message confirms the action:

```
xtrabackup: Transaction log of lsn (5696709) to (5696718) was copied.
160401 10:25:51 completed OK!
```

Warning: **xtrabackup** does not copy the keyring file into the backup directory. To prepare the backup, you must copy the keyring file manually.

Preparing the Backup

To prepare the backup specify the keyring-file-data.

```
$ xtrabackup --prepare --target-dir=/data/backup \
--keyring-file-data=/var/lib/mysql-keyring/keyring
```

After **xtrabackup** takes the backup, the following message confirms the action:

```
InnoDB: Shutdown completed; log sequence number 5697064
160401 10:34:28 completed OK!
```

The backup is now prepared and can be restored with the *--copy-back* option. In case the keyring has been rotated, you must restore the keyring used when the backup was taken and prepared.

14.1.2 Using keyring_vault plugin

Keyring vault plugin settings are described here.

Creating Backup

The following command creates a backup in the /data/backup directory:

```
$ xtrabackup --backup --target-dir=/data/backup --user=root
```

After **xtrabackup** completes the action, the following confirmation message appears:

```
xtrabackup: Transaction log of lsn (5696709) to (5696718) was copied.
160401 10:25:51 completed OK!
```

Preparing the Backup

To prepare the backup, **xtrabackup** must access the keyring. **xtrabackup** does not communicate with the *MySQL* server or read the default my.cnf configuration file. Specify the keyring settings in the command line:

```
$ xtrabackup --prepare --target-dir=/data/backup \
--keyring-vault-config=/etc/vault.cnf
```

Note: Please look here for a description of keyring vault plugin settings.

After **xtrabackup** completes the action, the following confirmation message appears:

```
InnoDB: Shutdown completed; log sequence number 5697064
160401 10:34:28 completed OK!
```

The backup is now prepared and can be restored with the *--copy-back* option:

```
$ xtrabackup --copy-back --target-dir=/data/backup --datadir=/data/mysql
```

14.1.3 Using the `keyring_file` component

A component is not loaded with the `--early_plugin_load` option. The server uses a manifest to load the component and the component has its own configuration file. See component installation for more information.

An example of a manifest and a configuration file follows:

./bin/mysqld.my:

```
{
   "components": "file://component_keyring_file"
}
```

/lib/plugin/component_keyring_file.cnf:

```
{
   "path": "/var/lib/mysql-keyring/keyring_file", "read_only": false
}
```

For more information, see Keyring Component Installation and Using the keyring_file File-Based Keyring Plugin.

With the appropriate privilege, you can `SELECT` on the performance_schema.keyring_component_status table to view the attributes and status of the installed keyring component when in use.

The component has no special requirements for backing up a database that contains encrypted InnoDB tablespaces.

```
xtrabackup --backup --target-dir=/data/backup --user=root
```

After **xtrabackup** completes the action, the following confirmation message appears:

```
xtrabackup: Transaction log of lsn (5696709) to (5696718) was copied.
160401 10:25:51 completed OK!
```

Warning: **xtrabackup** does not copy the keyring file into the backup directory. To prepare the backup, you must copy the keyring file manually.

Preparing the Backup

xtrabackup reads the keyring_file component configuration from `xtrabackup_component_keyring_file.cnf`. You must specify the keyring_file data path if the `keyring-file-data` is not located in the attribute `PATH` from the xtrabackup_component_keyring_file.cnf.

The following is an example of adding the location for the keyring-file-data:

```
xtrabackup --prepare --target-dir=/data/backup \
--keyring-file-data=/var/lib/mysql-keyring/keyring
```

Note: **xtrabackup** attempts to read `xtrabackup_component_keyring_file.cnf`. You can assign another keyring file component configuration by passing the `--component-keyring-file-config` option.

After **xtrabackup** completes preparing the backup, the following confirmation message appears:

```
InnoDB: Shutdown completed; log sequence number 5697064
160401 10:34:28 completed OK!
```

The backup is prepared. To restore the backup use the `--copy-back` option. If the keyring has been rotated, you must restore the specific keyring used to take and prepare the backup.

14.1.4 Incremental Encrypted InnoDB tablespace backups with `keyring_file`

The process of taking incremental backups with InnoDB tablespace encryption is similar to taking the *Incremental Backups* with unencrypted tablespace.

Note: The `keyring-file` component should not used in production or for regulatory compliance.

Creating an Incremental Backup

To make an incremental backup, begin with a full backup. The **xtrabackup** binary writes a file called `xtrabackup_checkpoints` into the backup's target directory. This file contains a line showing the `to_lsn`, which is the database's *LSN* at the end of the backup. First you need to create a full backup with the following command:

```
$ xtrabackup --backup --target-dir=/data/backups/base \
--keyring-file-data=/var/lib/mysql-keyring/keyring
```

Warning: **xtrabackup** will not copy the keyring file into the backup directory. In order to prepare the backup, you must make a copy of the keyring file yourself. If you try to restore the backup after the keyring has been changed you'll see errors like `ERROR 3185 (HY000): Can't find master key from keyring, please check keyring plugin is loaded.` when trying to access an encrypted table.

If you look at the `xtrabackup_checkpoints` file, you should see contents similar to the following:

```
backup_type = full-backuped
from_lsn = 0
to_lsn = 7666625
last_lsn = 7666634
compact = 0
recover_binlog_info = 1
```

Now that you have a full backup, you can make an incremental backup based on it. Use a command such as the following:

```
$ xtrabackup --backup --target-dir=/data/backups/inc1 \
--incremental-basedir=/data/backups/base \
--keyring-file-data=/var/lib/mysql-keyring/keyring
```

Warning: ||xtrabackup| does not copy the keyring file into the backup directory. To prepare the backup, you must copy the keyring file manually.

If the keyring has not been rotated you can use the same as the one you've backed-up with the base backup. If the keyring has been rotated or you have upgraded the plugin to a component, you'll need to back up the keyring file, otherwise, you are unable to prepare the backup.

The `/data/backups/inc1/` directory should now contain delta files, such as `ibdata1.delta` and `test/table1.ibd.delta`. These represent the changes since the `LSN 7666625`. If you examine the `xtrabackup_checkpoints` file in this directory, you should see something similar to the following:

```
backup_type = incremental
from_lsn = 7666625
to_lsn = 8873920
last_lsn = 8873929
compact = 0
recover_binlog_info = 1
```

You can use this directory as the base for yet another incremental backup:

```
$ xtrabackup --backup --target-dir=/data/backups/inc2 \
--incremental-basedir=/data/backups/inc1 \
--keyring-file-data=/var/lib/mysql-keyring/keyring
```

Preparing Incremental Backups

The *--prepare* step for incremental backups is not the same as for normal backups. In normal backups, two types of operations are performed to make the database consistent: committed transactions are replayed from the log file against the data files, and uncommitted transactions are rolled back. You must skip the rollback of uncommitted transactions when preparing a backup, because transactions that were uncommitted at the time of your backup may be in progress, and it's likely that they will be committed in the next incremental backup. You should use the *--apply-log-only* option to prevent the rollback phase.

Warning: If you do not use the *--apply-log-only* option to prevent the rollback phase, then your incremental backups are useless. After transactions have been rolled back, further incremental backups cannot be applied.

Beginning with the full backup you created, you can prepare it and then apply the incremental differences to it. Recall that you have the following backups:

```
/data/backups/base
/data/backups/inc1
/data/backups/inc2
```

To prepare the base backup, you need to run *--prepare* as usual, but prevent the rollback phase:

```
$ xtrabackup --prepare --apply-log-only --target-dir=/data/backups/base \
--keyring-file-data=/var/lib/mysql-keyring/keyring
```

The output should end with some text such as the following:

```
InnoDB: Shutdown completed; log sequence number 7666643
InnoDB: Number of pools: 1
160401 12:31:11 completed OK!
```

To apply the first incremental backup to the full backup, you should use the following command:

```
$ xtrabackup --prepare --apply-log-only --target-dir=/data/backups/base \
--incremental-dir=/data/backups/inc1 \
--keyring-file-data=/var/lib/mysql-keyring/keyring
```

Warning: The backup should be prepared with the keyring file and type that was used when backup was being taken. This means that if the keyring has been rotated or you have upgraded from a plugin to a component between the base and incremental backup that you must use the keyring that was in use when the first incremental backup has been taken.

Preparing the second incremental backup is a similar process: apply the deltas to the (modified) base backup, and you will roll its data forward in time to the point of the second incremental backup:

```
$ xtrabackup --prepare --target-dir=/data/backups/base \
--incremental-dir=/data/backups/inc2 \
--keyring-file-data=/var/lib/mysql-keyring/keyring
```

Note: *--apply-log-only* should be used when merging all incrementals except the last one. That's why the previous line doesn't contain the *--apply-log-only* option. Even if the *--apply-log-only* was used on the last step, backup would still be consistent but in that case server would perform the rollback phase.

The backup is now prepared and can be restored with *--copy-back* option. In case the keyring has been rotated you'll need to restore the keyring which was used to take and prepare the backup.

14.2 Using Key Management Interoperability Protocol

This feature is *tech preview* quality.

Percona XtraBackup 8.0.27-19 adds support for the Key Management Interoperability Protocol (KMIP) which enables the communication between the key management system and encrypted database server.

Percona XtraBackup has no special requirements for backing up a database that contains encrypted InnoDB tablespaces.

Percona XtraBackup performs the following actions:

1. Connects to the MySQL server
2. Pulls the configuration
3. Connects to the KMIP server
4. Fetches the necessary keys from the KMIP server
5. Stores the KMIP server configuration settings in the `xtrabackup_component_keyring_kmip.cnf` file in the backup directory

When preparing the backup, Percona XtraBackup connects to the KMIP server with the settings from the `xtrabackup_component_keyring_kmip.cnf` file.

14.3 Restoring a Backup When Keyring Is not Available

While the described restore method works, this method requires access to the same keyring that the server is using. It may not be possible if the backup is prepared on a different server or at a much later time, when keys in the keyring are purged, or, in the case of a malfunction, when the keyring vault server is not available at all.

The `--transition-key=<passphrase>` option should be used to make it possible for **xtrabackup** to process the backup without access to the keyring vault server. In this case, **xtrabackup** derives the AES encryption key from the specified passphrase and will use it to encrypt tablespace keys of tablespaces that are being backed up.

Creating a Backup with a Passphrase

The following example illustrates how the backup can be created in this case:

```
$ xtrabackup --backup --user=root -p --target-dir=/data/backup \
--transition-key=MySecetKey
```

If `--transition-key` is specified without a value, **xtrabackup** will ask for it.

Note: **xtrabackup** scrapes `--transition-key` so that its value is not visible in the `ps` command output.

Preparing the Backup with a Passphrase

The same passphrase should be specified for the *prepare* command:

```
$ xtrabackup --prepare --target-dir=/data/backup
```

There are no `--keyring-vault...`,``–keyring-file...``, or `--component-keyring-file-config` options here, because **xtrabackup** does not talk to the keyring in this case.

Restoring the Backup with a Generated Key

When restoring a backup you will need to generate a new master key. Here is the example for `keyring_file` plugin or component:

```
$ xtrabackup --copy-back --target-dir=/data/backup --datadir=/data/mysql \
--transition-key=MySecetKey --generate-new-master-key \
--keyring-file-data=/var/lib/mysql-keyring/keyring
```

In case of `keyring_vault`, it will look like this:

```
$ xtrabackup --copy-back --target-dir=/data/backup --datadir=/data/mysql \
--transition-key=MySecetKey --generate-new-master-key \
--keyring-vault-config=/etc/vault.cnf
```

xtrabackup will generate a new master key, store it in the target keyring vault server and re-encrypt the tablespace keys using this key.

Making the Backup with a Stored Transition Key

Finally, there is an option to store a transition key in the keyring. In this case, **xtrabackup** will need to access the same keyring file or vault server during prepare and copy-back but does not depend on whether the server keys have been purged.

In this scenario, the three stages of the backup process look as follows.

- Backup

```
$ xtrabackup --backup --user=root -p --target-dir=/data/backup \
--generate-transition-key
```

- Prepare
  - `keyring_file` variant:

```
$ xtrabackup --prepare --target-dir=/data/backup \
--keyring-file-data=/var/lib/mysql-keyring/keyring
```

  - `keyring_vault` variant:

```
$ xtrabackup --prepare --target-dir=/data/backup \
--keyring-vault-config=/etc/vault.cnf
```

- Copy-back
  - `keyring_file` variant:

```
$ xtrabackup --copy-back --target-dir=/data/backup --datadir=/data/mysql \
--generate-new-master-key --keyring-file-data=/var/lib/mysql-keyring/keyring
```

  - `keyring_vault` variant:

```
$ xtrabackup --copy-back --target-dir=/data/backup --datadir=/data/mysql \
--generate-new-master-key --keyring-vault-config=/etc/vault.cnf
```

CHAPTER

FIFTEEN

`LOCK-DDL-PER-TABLE` OPTION IMPROVEMENTS

To block DDL statements on an instance, Percona Server implemented LOCK TABLES FOR BACKUP. *Percona XtraBackup* uses this lock for the duration of the backup. This lock does not affect DML statements.

Percona XtraBackup has also implemented `--lock-ddl-per-table`, which blocks DDL statements by using metadata locks (MDL).

The following procedures describe a simplified backup operation when using `--lock-ddl-per-table`:

1. Parse and copy all redo logs after the checkpoint mark
2. Fork a dedicated thread to continue following new redo log entries
3. List the tablespaces required to copy
4. Iterate through the list. The following steps occur with each listed tablespace:

    a. Query INFORMATION_SCHEMA.INNODB_TABLES to find which tables belong to the tablespace ID and take a MDL on the underlying table or tables in case there is a shared tablespace.

    b. Copy the tablespace `.ibd` files.

The backup process may encounter a redo log event, generated by the bulk load operations, which notifies backup tools that data file changes have been omitted from the redo log. This event is an `MLOG_INDEX_LOAD`. If this event is found by the redo follow thread, the backup continues and assumes the backup is safe because the MDL protects tablespaces already copied and the MLOG_INDEX_LOAD event is for a tablespace that is not copied.

These assumptions may not be correct and may lead to inconsistent backups.

15.1 `--lock-ddl-per-table` redesign

Implemented in *Percona XtraBackup* version 8.0.22-15.0, the `--lock-ddl-per-table` has been redesigned to minimize inconsistent backups. The following procedure reorders the steps:

- The MDL lock acquired at the beginning of the backup
- Scan the redo logs. An `MLOG_INDEX_LOAD` event may be recorded if a `CREATE INDEX` statement has occurred before the backup starts. At this time, the backup process is safe and can parse and accept the event.
- After the first scan has completed, the follow redo log thread is initiated. This thread stops the backup process if an `MLOG_INDEX_LOAD` event is found.
- Gather the tablespace list to copy
- Copy the `.ibd` files.

15.2 Other Improvements

The following improvements have been added:

- If the `.ibd` file belongs to a temporary table, the `SELECT` query is skipped.
- For a FullText Index, an MDL is acquired on the base table.
- A `SELECT` query that acquires an MDL does not retrieve any data.

Warning: The `lock-ddl-per-table` variable is deprecated in *Percona Server for MySQL* 8.0. Use `--lock-ddl` instead of this variable.

CHAPTER

SIXTEEN

INCREMENTAL BACKUP USING PAGE TRACKING

This feature is *tech preview* quality.

Percona XtraBackup 8.0.27 adds support for the page tracking functionality for incremental backup .

To create an incremental backup with page tracking, Percona XtraBackup uses the MySQL mysqlbackup component. This component provides a list of pages modified since the last backup, and Percona XtraBackup copies only those pages. This operation removes the need to scan all of the pages in the database. If the majority of pages have not been modified, the page tracking feature can improve the speed of incremental backups.

16.1 Install the component

To start using the page tracking functionality, do the following steps:

1. Install the mysqlbackup component and enable it on the server:

```
$ INSTALL COMPONENT "file://component_mysqlbackup";
```

2. Check whether the mysqlbackup component is installed succesfuly:

```
SELECT COUNT(1) FROM mysql.component WHERE component_urn='file://component_
↪mysqlbackup';
```

16.2 Using page tracking

You can enable the page tracking functionality for the full and incremental backups with the `--page-tracking` option.

The option has the following benefits:

- Resets page tracking to the start of the backup. This reset allows the next incremental backup to use page tracking.
- Allows the use of page tracking for an incremental backup if the page tracking data is available from the backup's start checkpoint LSN.

Note: Percona XtraBackup processes a list of all the tracked pages in memory. If Percona XtraBackup does not have enough available memory to process the list of all the tracked pages, Percona XtraBackup throws an error and exits. For example, if an incremental backup uses 200GB, Percona XtraBackup can additionally use about 100MB of memory to store the page tracking data.

An example of creating a full backup using the `--page-tracking` option.

```
xtrabackup --backup --target-dir=$FULL_BACK --page-tracking
```

An example of creating an incremental backup using the `--page-tracking` option.

```
xtrabackup --backup --target-dir=$INC_BACKUP  --incremental-basedir=$FULL_
↪BACKUP --page-tracking
```

After enabling the functionality, the next incremental backup finds changed pages using page tracking.

Note: For the very first full backup using page tracking, Percona XtraBackup may have a delay. The following is an example of the message you can receive:

```
xtrabackup: pagetracking: Sleeping for 1 second, waiting for checkpoint lsn 17852922 /
to reach to page tracking start lsn 21353759
```

You can avoid this delay by enabling page tracking before creating the first backup. Thus, you ensure that page tracking log sequence number (LSN) is more than the checkpoint LSN of the server.

16.3 Start page tracking manually

After the mysqlbackup component is loaded and active on the server, you can start page tracking manually with the following option:

```
$ SELECT mysqlbackup_page_track_set(true);
```

16.4 Check the LSN value

Check the LSN value starting from which changed pages are tracked with the following option:

```
$ SELECT mysqlbackup_page_track_get_start_lsn();
```

16.5 Stop page tracking

If you need to stop page tracking, use the following option:

```
$ SELECT mysqlbackup_page_track_set(false);
```

16.6 Purge page tracking data

When you start page tracking, it creates a file under the server's datadir to collect data about changed pages. This file grows until you stop the page tracking. If you stop the server and then restart it, page tracking creates a new file but also keeps the old one. The old file continues to grow until you stop the page tracking explicitly.

If you purge the page tracking data, you should create a full backup afterward. To purge the page tracking data, do the following steps:

```
$ SELECT mysqlbackup_page_track_set(false);
$ SELECT mysqlbackup_page_track_purge_up_to(9223372036854775807);
/* Specify the LSN up to which you want to purge page tracking data. /
9223372036854775807 is the highest possible LSN which purges all page␣
↪tracking files.*/
$ SELECT mysqlbackup_page_track_set(true);
```

16.7 Known issue

If the index is built in place using an exclusive algorithm and then is added to a table after the last LSN checkpoint, you may get a bad incremental backup using page tracking. Find more details in PS-8032 .

16.8 Uninstall the mysqlbackup component

If you need to uninstall the mysqlbackup component, use the following option:

```
$ UNINSTALL COMPONENT "file://component_mysqlbackup"
```

Part VII

Security

CHAPTER

SEVENTEEN

WORKING WITH SELINUX

Percona XtraBackup is installed as an unconfined process running in an undefined domain. SELinux allows unconfined processes almost all access and the processes only use Discretionary Access Control (DAC) rules.

You find the current state of the *Percona XtraBackup* file with the following command:

```
$ ls -Z /usr/bin | grep xtrabackup
-rwxr-xr-x. root root   system_u:object_r:bin_t:s0       xtrabackup
```

The SELinux context is the following:

- user (root)
- role (object_r)
- type (bin_t)
- level (s0)

The unconfined domain supports the network-facing services, which are protected by SELinux. These domains are not exposed. In this configuration, SELinux protects against remote intrusions but local intrusions, which require local access, are not confined.

Percona XtraBackup works locally. The service is not network-facing and cannot be exploited externally. The service interacts only with the local user, who provides the parameters. *Percona XtraBackup* requires access to the `target-dir` location.

17.1 Confine XtraBackup

You can modify your security configuration to confine *Percona XtraBackup*. The first question is where to store the backup files. The service requires read and write access to the selected location.

You can use either of the following methods:

- Allow *Percona XtraBackup* to write to any location. The user provides any path to the `target-dir` parameter.
- Allow *Percona XtraBackup* to write to a specific location, such as /backups or the user's home directory.

The first option opens the entire system to read and write. Select the second option to harden your security.

17.2 Install SELinux tools

To work with policies, you must install the SELinux tools. To find which package provides the `semanage` command and install the package. The following is an example on CentOS 7.

```
$ yum provides *bin/semanage
...
policycoreutils-python-2.5-34.el7.x86_64 : SELinux policy core python␣
↪utilities
...
$ sudo yum install -y policycoreutils-python
```

The following is an example on CentOS 8:

```
$ yum provides *bin/semanage
...
policycoreutils-python-utils-2.8-16.1.el8.noarch : SELinux policy core␣
↪python utilities
...
$ sudo yum install -y policycoreutils-python-utils
```

17.3 Create a policy

Use a modular approach to create an SELinux policy. Create a policy module to manage XtraBackup. You must create a `.te` file for type enforcement, and an optional `.fc` file for the file contexts.

Use *ps -efZ | grep xtrabackup* to verify the service is not confined by SELinux.

Create the `xtrabackup.fc` file and add content. This file defines the security contexts.

```
/usr/bin/xtrabackup    -- gen_context(system_u:object_r:xtrabackup_exec_t,s0)
/usr/bin/xbcrypt    -- gen_context(system_u:object_r:xtrabackup_exec_t,s0)
/usr/bin/xbstream    -- gen_context(system_u:object_r:xtrabackup_exec_t,s0)
/usr/bin/xbcloud    -- gen_context(system_u:object_r:xtrabackup_exec_t,s0)
/backups(/.*)?       system_u:object_r:xtrabackup_data_t:s0
```

Note: If you are using the `/backups` directory you must have the last line. If you are storing the backups in the user's home directory, you can omit this line.

Download the `xtrabackup.te` file from the following location:

https://github.com/percona/percona-xtrabackup/tree/8.0/packaging/percona/selinx

Note: In the file, the sections in bold should be modified for your system. The fc file can also be downloaded from the same location.

Complile the policy module:

```
$ make -f /usr/share/selinux/devel/Makefile xtrabackup.pp
```

Install the module:

```
$ semodule -i xtrabackup.pp
```

Tag the PXB binaries with the proper SELinux tags, such as `xtrabackup_exec_t`.

```
$ restorecon -v /usr/bin/*
```

If you store your backups at `/backups`, restore the tag in that location:

```
$ restorecon -v /backups
```

Note: Remember to add the standard Linux DAC permissions for this directory.

Perform the backup in the standard way.

CHAPTER

EIGHTEEN

WORKING WITH APPARMOR

The Linux Security Module implements mandatory access controls (MAC) with AppArmor. Debian and Ubuntu systems install AppArmor by default. AppArmor uses profiles which define which files and permissions are needed for application.

Percona XtraBackup does not have a profile and is not confined by AppArmor.

For a list of common AppArmor commands, see Percona Server for MySQL - AppArmor

18.1 Develop a profile

Download the profile from:

https://github.com/percona/percona-xtrabackup/tree/8.0/packaging/percona/apparmor/apparmor.d

The following profile sections should be updated with your system information, such as location of the backup destination directory.

```
# enable storing backups only in /backups directory
# /backups/** rwk,

# enable storing backups anywhere in caller user home directory
/@{HOME}/** rwk,


# enable storing backups only in /backups directory
# /backups/** rwk,

# enable storing backups anywhere in caller user home directory
/@{HOME}/** rwk,
}

# enable storing backups only in /backups directory
# /backups/** rwk,

# enable storing backups anywhere in caller user home directory
/@{HOME}/** rwk,
}
```

Move the updated file:

```
$ sudo mv usr.sbin.xtrabackup /etc/apparmor.d/
```

Install the profile with the following command:

```
$ sudo apparmor_parser -r -T -W /etc/apparmor.d/usr.sbin.xtrabackup
```

Run the backup as usual.

No additional AppArmor-related actions are required to restore a backup.

Part VIII

xbcloud Binary

CHAPTER

NINETEEN

USING THE XBCLOUD BINARY WITH SWIFT

19.1 Creating a full backup with Swift

The following example shows how to make a full backup and upload it to Swift.

```
$ xtrabackup --backup --stream=xbstream --extra-lsndir=/tmp --target-dir=/tmp | \
xbcloud put --storage=swift \
--swift-container=test \
--swift-user=test:tester \
--swift-auth-url=http://192.168.8.80:8080/ \
--swift-key=testing \
--parallel=10 \
full_backup
```

The following OpenStack environment variables are also recognized and mapped automatically to the corresponding **swift** parameters (`--storage=swift`):

- OS_AUTH_URL
- OS_TENANT_NAME
- OS_TENANT_ID
- OS_USERNAME
- OS_PASSWORD
- OS_USER_DOMAIN
- OS_USER_DOMAIN_ID
- OS_PROJECT_DOMAIN
- OS_PROJECT_DOMAIN_ID
- OS_REGION_NAME
- OS_STORAGE_URL
- OS_CACERT

19.2 Restoring with Swift

```
$ xbcloud get [options] <name> [<list-of-files>] | xbstream -x
```

The following example shows how to fetch and restore the backup from Swift:

```
$ xbcloud get --storage=swift \
--swift-container=test \
--swift-user=test:tester \
--swift-auth-url=http://192.168.8.80:8080/ \
--swift-key=testing \
full_backup | xbstream -xv -C /tmp/downloaded_full

$ xbcloud delete --storage=swift --swift-user=xtrabackup \
--swift-password=xtrabackup123! --swift-auth-version=3 \
--swift-auth-url=http://openstack.ci.percona.com:5000/ \
--swift-container=mybackup1 --swift-domain=Default
```

19.3 Command-line options

xbcloud has the following command line options:

--storage=[swift|s3|google]
: Cloud storage option. *xbcloud* supports Swift, MinIO, and AWS S3. The default value is `swift`.

--swift-auth-url
: URL of Swift cluster.

--swift-storage-url
: xbcloud will try to get object-store URL for given region (if any specified) from the keystone response. One can override that URL by passing –swift-storage-url=URL argument.

--swift-user
: Swift username (X-Auth-User, specific to Swift)

--swift-key
: Swift key/password (X-Auth-Key, specific to Swift)

--swift-container
: Container to backup into (specific to Swift)

--parallel=N
: Maximum number of concurrent upload/download requests. Default is `1`.

--cacert
: Path to the file with CA certificates

--insecure
: Do not verify servers certificate

19.3.1 Swift authentication options

Swift specification describes several authentication options. *xbcloud* can authenticate against keystone with API version 2 and 3.

--swift-auth-version
Specifies the swift authentication version. Possible values are: `1.0` - TempAuth, `2.0` - Keystone v2.0, and `3` - Keystone v3. Default value is `1.0`.

For v2 additional options are:

--swift-tenant
Swift tenant name.

--swift-tenant-id
Swift tenant ID.

--swift-region
Swift endpoint region.

--swift-password
Swift password for the user.

For v3 additional options are:

--swift-user-id
Swift user ID.

--swift-project
Swift project name.

--swift-project-id
Swift project ID.

--swift-domain
Swift domain name.

--swift-domain-id
Swift domain ID.

CHAPTER

TWENTY

USING XBCLOUD BINARY WITH AMAZON S3

20.1 Creating a full backup with Amazon S3

```
$ xtrabackup --backup --stream=xbstream --extra-lsndir=/tmp --target-dir=/tmp | \
xbcloud put --storage=s3 \
--s3-endpoint='s3.amazonaws.com' \
--s3-access-key='YOUR-ACCESSKEYID' \
--s3-secret-key='YOUR-SECRETACCESSKEY' \
--s3-bucket='mysql_backups'
--parallel=10 \
$(date -I)-full_backup
```

The following options are available when using Amazon S3:

Option	Details
–s3-access-key	Use to supply the AWS access key ID
–s3-secret-key	Use to supply the AWS secret access key
–s3-bucket	Use supply the AWS bucket name
–s3-region	Use to specify the AWS region. The default value is **us-east-1**
–s3-api-version = <AUTO|2|4>	Select the signing algorithm. The default value is AUTO. In this case, *xbcloud* will probe.
–s3-bucket-lookup = <AUTO|PATH|DNS>	Specify whether to use **bucket.endpoint.com** or *endpoint.com/bucket** style requests. The default value is AUTO. In this case, *xbcloud* will probe.
–s3-storage-class=<name>	Specify the S3 storage class. The default storage class depends on the provider. The name options are the following: • STANDARD • STANDARD_IA • GLACIER **Note:** If you use the GLACIER storage class, the object must be restored to S3 before restoring the backup. Also supports using custom S3 implementations such as MinIO or CephRadosGW.

20.2 Environment variables

The following environment variables are recognized. xbcloud maps them automatically to corresponding parameters applicable to the selected storage.

- AWS_ACCESS_KEY_ID (or ACCESS_KEY_ID)
- AWS_SECRET_ACCESS_KEY (or SECRET_ACCESS_KEY)
- AWS_DEFAULT_REGION (or DEFAULT_REGION)
- AWS_ENDPOINT (or ENDPOINT)
- AWS_CA_BUNDLE

20.3 Restoring with S3

```
$ xbcloud get s3://operator-testing/bak22 \
--s3-endpoint=https://storage.googleapis.com/ \
--parallel=10 2>download.log | xbstream -x -C restore --parallel=8
```

CHAPTER

TWENTYONE

USING THE XBCLOUD BINARY WITH MINIO

21.1 Creating a full backup with MinIO

```
$ xtrabackup --backup --stream=xbstream --extra-lsndir=/tmp --target-dir=/tmp | \
xbcloud put --storage=s3 \
--s3-endpoint='play.minio.io:9000' \
--s3-access-key='YOUR-ACCESSKEYID' \
--s3-secret-key='YOUR-SECRETACCESSKEY' \
--s3-bucket='mysql_backups'
--parallel=10 \
$(date -I)-full_backup
```

CHAPTER

TWENTYTWO

USING THE XBCLOUD WITH GOOGLE CLOUD STORAGE

22.1 Creating a full backup with Google Cloud Storage

The support for Google Cloud Storage is implemented using the interoperability mode. This mode was especially designed to interact with cloud services compatible with Amazon S3.

See also:

Cloud Storage Interoperability https://cloud.google.com/storage/docs/interoperability

```
$ xtrabackup --backup --stream=xbstream --extra-lsndir=/tmp --target-dir=/tmp | \
xbcloud put --storage=google \
--google-endpoint=`storage.googleapis.com` \
--google-access-key='YOUR-ACCESSKEYID' \
--google-secret-key='YOUR-SECRETACCESSKEY' \
--google-bucket='mysql_backups'
--parallel=10 \
$(date -I)-full_backup
```

The following options are available when using Google Cloud Storage:

- –google-access-key = <ACCESS KEY ID>
- –google-secret-key = <SECRET ACCESS KEY>
- –google-bucket = <BUCKET NAME>
- –google-storage-class=name

Note: The Google storage class name options are the following:

- STANDARD
- NEARLINE
- COLDLINE
- ARCHIVE

See also:

Google storage classes The default Google storage class depends on the storage class of the bucket

---

CHAPTER

TWENTYTHREE

EXPONENTIAL BACKOFF

This feature was implemented in *Percona XtraBackup 8.0.26-18.0* in the xbcloud binary.

Exponential backoff increases the chances for the completion of a backup or a restore operation. For example, a chunk upload or download may fail if you have an unstable network connection or other network issues. This feature adds an exponential backoff, or sleep, time and then retries the upload or download.

When a chunk upload or download operation fails, xbcloud checks the reason for the failure. This failure can be a CURL error or an HTTP error, or a client-specific error. If the error is listed in the *Retriable errors* list, xbcloud pauses for a calculated time before retrying the operation until that time reaches the `--max-backoff` value.

The operation is retried until the `--max-retries` value is reached. If the chunk operation fails on the last retry, xbcloud aborts the process.

The default values are the following:

- –max-backoff = 300000 (5 minutes)
- –max-retries = 10

You can adjust the number of retries by adding the `--max-retries` parameter and adjust the maximum length of time between retries by adding the `--max-backoff` parameter to an xbcloud command.

Since xbcloud does multiple asynchronous requests in parallel, a calculated value, measured in milliseconds, is used for `max-backoff`. This algorithm calculates how many milliseconds to sleep before the next retry. A number generated is based on the combining the power of two (2), the number of retries already attempted and adds a random number between 1 and 1000. This number avoids network congestion if multiple chunks have the same backoff value. If the default values are used, the final retry attempt to be approximately 17 minutes after the first try. The number is no longer calculated when the milliseconds reach the `--max-backoff` setting. At that point, the retries continue by using the `--max-backoff` setting until the `max-retries` parameter is reached.

23.1 Retriable errors

We retry for the following CURL operations:

- CURLE_GOT_NOTHING
- CURLE_OPERATION_TIMEOUT
- CURLE_RECV_ERROR
- CURLE_SEND_ERROR
- CURLE_SEND_FAIL_REWIND
- CURLE_PARTIAL_FILE
- CURLE_SSL_CONNECT_ERROR

We retry for the following HTTP operation status codes:

- 503
- 500
- 504
- 408

Each cloud provider may return a different CURL error or an HTTP error, depending on the issue. Add new errors by setting the following variables `--curl-retriable-errors` or `--http-retriable-errors` on the command line or in `my.cnf` or in a custom configuration file under the [xbcloud] section.

The error handling is enhanced when using the `--verbose` output. This output specifies which error caused xbcloud to fail and what parameter a user must add to retry on this error.

The following is an example of a verbose output:

```
210701 14:34:23 /work/pxb/ins/8.0/bin/xbcloud: Operation failed. Error: Server␣
↪returned nothing (no headers, no data)
210701 14:34:23 /work/pxb/ins/8.0/bin/xbcloud: Curl error (52) Server returned␣
↪nothing (no headers, no data) is not configured as retriable. You can allow it by␣
↪adding --curl-retriable-errors=52 parameter
```

23.2 Example

The following example adjusts the maximum number of retries and the maximum time between retries.

```
xbcloud [options] --max-retries=5 --max-backoff=10000
```

The following text is an example of the exponential backoff used with the command:

```
210702 10:07:05 /work/pxb/ins/8.0/bin/xbcloud: Operation failed. Error: Server␣
↪returned nothing (no headers, no data)
210702 10:07:05 /work/pxb/ins/8.0/bin/xbcloud: Sleeping for 2384 ms before retrying␣
↪backup3/xtrabackup_logfile.00000000000000000006 [1]
. . .
210702 10:07:23 /work/pxb/ins/8.0/bin/xbcloud: Operation failed. Error: Server␣
↪returned nothing (no headers, no data)
210702 10:07:23 /work/pxb/ins/8.0/bin/xbcloud: Sleeping for 4387 ms before retrying␣
↪backup3/xtrabackup_logfile.00000000000000000006 [2]
. . .
210702 10:07:52 /work/pxb/ins/8.0/bin/xbcloud: Operation failed. Error: Failed␣
↪sending data to the peer
210702 10:07:52 /work/pxb/ins/8.0/bin/xbcloud: Sleeping for 8691 ms before retrying␣
↪backup3/xtrabackup_logfile.00000000000000000006 [3]
. . .
210702 10:08:47 /work/pxb/ins/8.0/bin/xbcloud: Operation failed. Error: Failed␣
↪sending data to the peer
210702 10:08:47 /work/pxb/ins/8.0/bin/xbcloud: Sleeping for 10000 ms before retrying␣
↪backup3/xtrabackup_logfile.00000000000000000006 [4]
. . .
210702 10:10:12 /work/pxb/ins/8.0/bin/xbcloud: successfully uploaded chunk: backup3/
↪xtrabackup_logfile.00000000000000000006, size: 8388660
```

The following list details the example output:

[1.] Chunk `xtrabackup_logfile.00000000000000000006` fails to upload _ the first time and slept for 2384 milliseconds.

[2.] The same chunk fails for the second time and the time is increased to 4387 milliseconds.

[3.] The same chunk fails for the third time and the time is increased to 8691 milliseconds.

[4.] The same chunk fails for the fourth time. The `max-backoff` parameter has been reached. All retries sleep the same amount of time after reaching the parameter.

[5.] The same chunk is successfully uploaded.

CHAPTER

TWENTYFOUR

USING THE XBCLOUD BINARY WITH MICROSOFT AZURE CLOUD STORAGE

This feature is *technical preview* quality.

Implemented in Percona XtraBackup 8.0.27-19, the **xbcloud** binary adds support for the Microsoft Azure Cloud Storage using the REST API.

24.1 Options

The following are the options, environment variables, and descriptions for uploading a backup to Azure using the REST API. The environment variables are recognized by **xbcloud**, which maps them automatically to the corresponding parameters:

Option name	Environment variables	Description
`--azure-storage-account=name`	AZURE_STORAGE_ACCOUNT	An Azure storage account is a unique namespace to access and store your Azure data objects.
`--azure-container-name=name`	AZURE_CONTAINER_NAME	A container name is a valid DNS name that conforms to the Azure naming rules
`--azure-access-key=name`	AZURE_ACCESS_KEY	A generated key that can be used to authorize access to data in your account using the Shared Key authorization.
`--azure-endpoint=name`	AZURE_ENDPOINT	The endpoint allows clients to securely access data
`--azure-tier-class=name`	AZURE_STORAGE_CLASS	Cloud tier can decrease the local storage required while maintaining the performance. When enabled, this feature has the following categories: • Hot - Frequently accessed or modified data • Cool - Infrequently accessed or modified data • Archive - Rarely accessed or modified data

Test your Azure applications with the Azurite open-source emulator. For testing purposes, the **xbcloud** binary adds

the `--azure-development-storage` option that uses the default `access_key` and `storage account` of azurite and `testcontainer` for the container. You can overwrite these options, if needed.

24.2 Usage

All of the available options for **xbcloud**, such as parallel, max-retries, and others, can be used. For more information, *The xbcloud Binary*.

24.3 Examples

An example of an **xbcloud** backup.

```
xtrabackup --backup --stream=xbstream --target-dir= $TARGET_DIR | xbcloud put backup_
↪name --azure-storage-account=pxbtesting --azure-access-key=$AZURE_KEY --azure-
↪container-name=test --storage=azure
```

An example of restoring a backup from **xbcloud**.

```
xbcloud get backup_name  --azure-storage-account=pxbtesting --azure-access-key=$AZURE_
↪KEY --azure-container-name=test --storage=azure --parallel=10 2>download.log |␣
↪xbstream -x -C restore
```

An example of deleting a backup from **xbcloud**.

```
xbcloud delete backup_name --azure-storage-account=pxbtesting --azure-access-key=
↪$AZURE_KEY --azure-container-name=test --storage=azure
```

An example of using a shortcut restore.

```
xbcloud get azure://operator-testing/bak22 ...
```

Part IX

Tutorials, Recipes, How-tos

CHAPTER

TWENTYFIVE

HOW-TOS AND RECIPES

25.1 Recipes for `xtrabackup`

25.1.1 Making a Full Backup

Backup the InnoDB data and log files located in `/var/lib/mysql/` to `/data/backups/mysql/` (destination). Then, prepare the backup files to be ready to restore or use (make the data files consistent).

Making a backup

```
$ xtrabackup --backup --target-dir=/data/backup/mysql/
```

Preparing the backup twice

```
$ xtrabackup --prepare --target-dir=/data/backup/mysql/
$ xtrabackup --prepare --target-dir=/data/backup/mysql/
```

Success Criteria

- The exit status of xtrabackup is 0.
- In the second *`--prepare`* step, you should see InnoDB print messages similar to `Log file ./ib_logfile0 did not exist: new to be created`, followed by a line indicating the log file was created (creating new logs is the purpose of the second preparation).

Note: You might want to set the *`--use-memory`* option to a value close to the size of your buffer pool, if you are on a dedicated server that has enough free memory. More details *here*.

A more detailed explanation is *here*.

25.1.2 Making an Incremental Backup

Backup all InnoDB data and log files - located in `/var/lib/mysql/` - **once**, then make two daily incremental backups in `/data/backups/mysql/` (destination). Finally, prepare the backup files to be ready to restore or use.

Create one full backup

Making an incremental backup requires a full backup as a base:

```
xtrabackup --backup --target-dir=/data/backups/mysql/
```

It is important that you **do not run** the *--prepare* command yet.

Create two incremental backups

Suppose the full backup is on Monday, and you will create an incremental one on Tuesday:

```
xtrabackup --backup --target-dir=/data/backups/inc/tue/ \
      --incremental-basedir=/data/backups/mysql/
```

and the same policy is applied on Wednesday:

```
xtrabackup --backup --target-dir=/data/backups/inc/wed/ \
       --incremental-basedir=/data/backups/inc/tue/
```

Prepare the base backup

Prepare the base backup (Monday's backup):

```
xtrabackup --prepare --apply-log-only --target-dir=/data/backups/mysql/
```

Roll forward the base data to the first increment

Roll Monday's data forward to the state on Tuesday:

```
xtrabackup --prepare --apply-log-only --target-dir=/data/backups/mysql/ \
   --incremental-dir=/data/backups/inc/tue/
```

Roll forward again to the second increment

Roll forward again to the state on Wednesday:

```
xtrabackup --prepare --apply-log-only --target-dir=/data/backups/mysql/ \
   --incremental-dir=/data/backups/inc/wed/
```

Prepare the whole backup to be ready to use

Create the new logs by preparing it:

```
xtrabackup --prepare --target-dir=/data/backups/mysql/
```

Notes

- You might want to set the *--use-memory* to speed up the process if you are on a dedicated server that has enough free memory. More details *here*.
- A more detailed explanation is *here*.

25.1.3 Make a Streaming Backup

Stream mode sends the backup to `STDOUT` in the `xbstream` format instead of copying it to the directory named by the first argument. You can pipe the output to a local file, or, across the network, to another server.

To extract the resulting `xbstream` file, you **must** use the `xbstream` utility: `xbstream -x < backup.xbstream`.

Examples of Using `xbstream`

Task	Command
Stream the backup into an archive named `backup.xbstream`	`xtrabackup --backup --stream=xbstream --target-dir=./ > backup.xbstream`
Stream the backup into a *compressed* archive named `backup.xbstream`	`xtrabackup --backup --stream=xbstream --compress --target-dir=./ > backup.xbstream`
Encrypt the backup	$ xtrabackup –backup –stream=xbstream ./ > backup.xbstream gzip - | openssl des3 -salt -k “password” > backup.xbstream.gz.des3
Unpack the backup to the current directory	`xbstream -x < backup.xbstream`
Send the backup compressed directly to another host and unpack it	`xtrabackup --backup --compress --stream=xbstream --target-dir=./ | ssh user@otherhost "xbstream -x"`

Send the backup to another server using `netcat`.	On the destination host:
	`$ nc -l 9999 | cat - > /data/backups/backup.xbstream`
	On the source host:
	`$ xtrabackup --backup --stream=xbstream ./ | nc desthost 9999`

Send the backup to another server using a one-liner:	$ ssh user@desthost “(nc -l 9999 > /data/backups/backup.xbstream &)” && xtrabackup –backup –stream=xbstream ./ | nc desthost 9999
Throttle the throughput to 10MB/sec using the `pipe viewer` tool[1]	$ xtrabackup –backup –stream=xbstream ./ | pv -q -L10m ssh user@desthost “cat - > /data/backups/backup.xbstream”

Checksumming the backup during the streaming:	On the destination host:
	`$ nc -l 9999 | tee >(sha1sum > destination_checksum) > /data/backups/backup.xbstream`
	On the source host:
	`$ xtrabackup --backup --stream=xbstream ./ | tee >(sha1sum > source_checksum) | nc desthost 9999`
	Compare the checksums on the source host:
	`$ cat source_checksum`
	`65e4f916a49c1f216e0887ce54cf59bf3934dbad -`
	Compare the checksums on the destination host:
	`$ cat destination_checksum`
	`65e4f916a49c1f216e0887ce54cf59bf3934dbad -`

Parallel compression with parallel copying backup	`xtrabackup --backup --compress --compress-threads=8 --stream=xbstream --parallel=4 --target-dir=./ > backup.xbstream`

See also:

More information about streaming and compressing backups Section *Streaming Backups*

[1] Install from the official site or from the distribution package (`apt install pv`)

25.1.4 Making a Compressed Backup

In order to make a compressed backup, use the *--compress* option along with the *--backup* and *--target-dir* options. By default *--compress* uses the `quicklz` compression algorithm.

```
$ xtrabackup --backup --compress --target-dir=/data/backup
```

You can also use the `lz4` compression algorithm by setting :option:`--compress` to `lz4`.

```
$ xtrabackup --backup --compress=lz4 --target-dir=/data/backup
```

If you want to speed up the compression you can use the parallel compression, which can be enabled with *--compress-threads* option. The following example uses four threads for compression.

```
$ xtrabackup --backup --compress --compress-threads=4 --target-dir=/data/backup
```

Output should look like this

```
...

[01] Compressing ./imdb/comp_cast_type.ibd to /data/backup/imdb/comp_cast_type.ibd.qp
[01]        ...done
...
130801 11:50:24  xtrabackup: completed OK
```

Preparing the backup

Before you can prepare the backup you'll need to uncompress all the files with qpress (which is available from Percona Software repositories). You can use the following one-liner to uncompress all the files:

```
$ for bf in `find . -iname "*\.qp"`; do qpress -d $bf $(dirname $bf) && rm $bf; done
```

If you used the `lz4` compression algorithm change this script to search for `*.lz4` files:

```
$ for bf in `find . -iname "*\.lz4"`; do lz4 -d $bf $(dirname $bf) && rm $bf; done
```

You can decompress the backup by using the *--decompress* option:

```
$ xtrabackup --decompress --target-dir=/data/backup/
```

See also:

What is required to use the *–decompress* option effectively? *--decompress*

When the files are uncompressed you can prepare the backup with the *--apply-log-only* option:

```
$ xtrabackup --apply-log-only --target-dir=/data/backup/
```

You should check for a confirmation message:

```
130802 02:51:02  xtrabackup: completed OK!
```

Now the files in `/data/backup/` is ready to be used by the server.

Note: *Percona XtraBackup* doesn't automatically remove the compressed files. In order to clean up the backup directory users should remove the `*.qp` files.

Restoring the backup

Once the backup has been prepared you can use the *--copy-back* to restore the backup.

```
$ xtrabackup --copy-back --target-dir=/data/backup/
```

This will copy the prepared data back to its original location as defined by the `datadir` variable in your *my.cnf*.

After the confirmation message, you should check the file permissions after copying the data back.

```
130802 02:58:44  xtrabackup: completed OK!
```

You may need to adjust the file permissions. The following example demonstrates how to do it recursively by using **chown**:

```
$ chown -R mysql:mysql /var/lib/mysql
```

Now, your *data directory* contains the restored data. You are ready to start the server.

25.1.5 Backing Up and Restoring Individual Partitions

Percona XtraBackup lets you backup individual partitions because partitions are regular tables with specially formatted names. The only requirement for this feature is having the *innodb_file_per_table* option enabled in the server.

There is one caveat about using this kind of backup: you can not copy back the prepared backup. Restoring partial backups should be done by importing the tables.

Creating the backup

There are three ways of specifying which part of the whole data will be backed up: regular expressions (–tables), enumerating the tables in a file (–tables) or providing a list of databases (–databases).

The regular expression provided to this option will be matched against the fully qualified database name and table name, in the form of `database-name.table-name`.

If the partition 0 is not backed up, Percona XtraBackup cannot generate a .cfg file. MySQL 8.0 stores the table metadata in partition 0.

For example, this operation takes a back up of the partition `p4` from the table `name` located in the database `imdb`:

```
xtrabackup --tables=^imdb[.]name#p#p4 --backup
```

If partition 0 is not backed up, the following errors may occur:

```
xtrabackup: export option not specified
xtrabackup: error: cannot find dictionary record of table imdb/name#p#p4
```

Note that this option is passed to `xtrabackup --tables` and is matched against each table of each database, the directories of each database will be created even if they are empty.

Preparing the backup

For preparing partial backups, the procedure is analogous to restoring individual tables apply the logs and use *xtrabackup –export*:

```
xtrabackup --apply-log --export /mnt/backup/2012-08-28_10-29-09
```

You may see warnings in the output about tables that do not exist. This is because *InnoDB*-based engines stores its data dictionary inside the tablespace files. **xtrabackup** removes the missing tables (those that haven't been selected in the partial backup) from the data dictionary in order to avoid future warnings or errors.

Restoring from the backups

Restoring should be done by importing the tables in the partial backup to the server.

First step is to create new table in which data will be restored.

```
CREATE TABLE `name_p4` (
`id` int(11) NOT NULL AUTO_INCREMENT,
`name` text NOT NULL,
`imdb_index` varchar(12) DEFAULT NULL,
`imdb_id` int(11) DEFAULT NULL,
`name_pcode_cf` varchar(5) DEFAULT NULL,
`name_pcode_nf` varchar(5) DEFAULT NULL,
`surname_pcode` varchar(5) DEFAULT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2812744 DEFAULT CHARSET=utf8
```

Note: Generate a .cfg metadata file by running `FLUSH TABLES ... FOR EXPORT`. The command can only be run on a table, not on the individual table partitions. The file is located in the table schema directory and is used for schema verification when importing the tablespace.

To restore the partition from the backup, the tablespace must be discarded for that table:

```
ALTER TABLE name_p4 DISCARD TABLESPACE;
```

The next step is to copy the *.exp* and *ibd* files from the backup to the MySQL data directory:

```
cp /mnt/backup/2012-08-28_10-29-09/imdb/name#p#p4.exp /var/lib/mysql/imdb/name_p4.exp
cp /mnt/backup/2012-08-28_10-29-09/imdb/name#P#p4.ibd /var/lib/mysql/imdb/name_p4.ibd
```

Note: Make sure that the copied files can be accessed by the user running MySQL.

The last step is to import the tablespace:

```
ALTER TABLE name_p4 IMPORT TABLESPACE;
```

25.2 How-Tos

25.2.1 How to setup a replica for replication in 6 simple steps with Percona XtraBackup

Data is, by far, the most valuable part of a system. Having a backup done systematically and available for a rapid recovery in case of failure is admittedly essential to a system. However, it is not common practice because of its costs, infrastructure needed or even the boredom associated to the task. Percona XtraBackup is designed to solve this problem.

You can have almost real-time backups in 6 simple steps by setting up a replication environment with Percona XtraBackup.

All the things you will need

Setting up a replica for replication with *Percona XtraBackup* is a straightforward procedure. In order to keep it simple, here is a list of the things you need to follow the steps without hassles:

`Source` A system with a *MySQL*-based server installed, configured and running. This system will be called `Source`, as it is where your data is stored and the one to be replicated. We will assume the following about this system:

- the *MySQL* server is able to communicate with others by the standard TCP/IP port;
- the *SSH* server is installed and configured;
- you have a user account in the system with the appropriate permissions;
- you have a MySQL's user account with appropriate privileges.
- server has binlogs enabled and server-id set up to 1.

`Replica` Another system, with a *MySQL*-based server installed on it. We will refer to this machine as `Replica` and we will assume the same things we did about `Source`, except that the server-id on `Replica` is 2.

`Percona XtraBackup` The backup tool we will use. It should be installed in both computers for convenience.

Note: It is not recommended to mix MySQL variants (Percona Server, MySQL) in your replication setup. This may produce incorrect `xtrabackup_slave_info` file when adding a new replica.

STEP 1: Make a backup on the `Source` and prepare it

At the `Source`, issue the following to a shell:

```
$ xtrabackup --backup --user=yourDBuser --password=MaGiCdB1 --target-dir=/path/to/
↪backupdir
```

After this is finished you should get:

```
xtrabackup: completed OK!
```

This will make a copy of your *MySQL* data dir to the `/path/to/backupdir` directory. You have told *Percona XtraBackup* to connect to the database server using your database user and password, and do a hot backup of all your data in it (all *MyISAM*, *InnoDB* tables and indexes in them).

In order for snapshot to be consistent you need to prepare the data on the source:

```
$ xtrabackup --user=yourDBuser --password=MaGiCdB1 \
            --prepare --target-dir=/path/to/backupdir
```

You need to select path where your snapshot has been taken. If everything is ok you should get the same OK message. Now the transaction logs are applied to the data files, and new ones are created: your data files are ready to be used by the MySQL server.

Percona XtraBackup knows where your data is by reading your *my.cnf*. If you have your configuration file in a non-standard place, you should use the flag `--defaults-file` =/location/of/my.cnf.

If you want to skip writing the user name and password every time you want to access *MySQL*, you can set it up in `.mylogin.cnf` as follows:

```
mysql_config_editor set --login-path=client --host=localhost --user=root --password
```

For more information, see *MySQL Configuration Utility <https://dev.mysql.com/doc/refman/8.0/en/mysql-config-editor.html>*.

This is will give you root access to MySQL.

STEP 2: Copy backed up data to the Replica

On the Source, use rsync or scp to copy the data from the Source to the Replica. If you are syncing the data directly to replica's data directory, we recommend that you stop the mysqld there.

```
$ rsync -avpP -e ssh /path/to/backupdir Replica:/path/to/mysql/
```

After data has been copied, you can back up the original or previously installed *MySQL datadir* (**NOTE**: Make sure mysqld is shut down before you move the contents of its datadir, or move the snapshot into its datadir.). Run the following commands on the Replica:

```
$ mv /path/to/mysql/datadir /path/to/mysql/datadir_bak
```

and move the snapshot from the `Source` in its place:

```
$ xtrabackup --move-back --target-dir=/path/to/mysql/backupdir
```

After you copy data over, make sure the Replica *MySQL* has the proper permissions to access them.

```
$ chown mysql:mysql /path/to/mysql/datadir
```

If the ibdata and iblog files are located in directories outside of the datadir, you must put these files in their proper place after the logs have been applied.

STEP 3: Configure the Source's MySQL server

On the source, run the following command to add the appropriate grant. This grant allows the replica to be able to connect to source:

```
> GRANT REPLICATION SLAVE ON *.*  TO 'repl'@'$replicaip'
IDENTIFIED BY '$replicapass';
```

Also make sure that firewall rules are correct and that the `Replica` can connect to the `Source`. Run the following command on the Replica to test that you can run the mysql client on `Replica`, connect to the `Source`, and authenticate.

```
$ mysql --host=Source --user=repl --password=$replicapass
```

Verify the privileges.

```
mysql> SHOW GRANTS;
```

STEP 4: Configure the Replica's MySQL server

Copy the *my.cnf* file from the `Source` to the `Replica`:

```
$ scp user@Source:/etc/mysql/my.cnf /etc/mysql/my.cnf
```

and change the following options in /etc/mysql/my.cnf:

```
server-id=2
```

and start/restart *mysqld* on the `Replica`.

In case you're using init script on Debian-based system to start mysqld, be sure that the password for `debian-sys-maint` user has been updated and it's the same as that user's password on the `Source`. Password can be seen and updated in `/etc/mysql/debian.cnf`.

STEP 5: Start the replication

On the `Replica`, review the content of the file `xtrabackup_binlog_info`, it will be something like:

```
 $ cat /var/lib/mysql/xtrabackup_binlog_info
Source-bin.000001     481
```

If you are using version 8.0.23 or later, on the `Replica`, execute the CHANGE_REPLICATION_SOURCE_TO and the appropriate options on a MySQL console. `CHANGE_MASTER_TO` is deprecated as of that release. Use the user name and password you created in STEP 3.

If you are using a version before 8.0.23, on the `Replica`, execute the `CHANGE MASTER` statement on a MySQL console and use the username and password you've set up in STEP 3:

```
CHANGE REPLICATION SOURCE TO
    SOURCE_HOST='$sourceip',
    SOURCE_USER='repl',
    SOURCE_PASSWORD='$replicapass',
    SOURCE_LOG_FILE='Source-bin.000001',
    SOURCE_LOG_POS=481;
```

Start the replica:

```
START REPLICA;
```

If you are using version 8.0.22 or later, use `START REPLICA` instead of `START SLAVE`. `START SLAVE` is deprecated as of that release. If you are using a version before 8.0.22, use `START SLAVE`.

STEP 6: Check

On the Replica, check that everything went OK with:

```
SHOW REPLICA STATUS\G
```

The result shows the status:

```

```

... Slave_IO_Running: Yes Slave_SQL_Running: Yes ... Seconds_Behind_Master: 13 ...

Both IO and SQL threads need to be running. The Seconds_Behind_Master means the SQL currently being executed has a current_timestamp of 13 seconds ago. It is an estimation of the lag between the Source and the Replica. Note that at the beginning, a high value could be shown because the Replica has to "catch up" with the Source.

Adding more replicas to the Source

You can use this procedure with slight variation to add new replicas to a source. We will use *Percona XtraBackup* to clone an already configured replica. We will continue using the previous scenario for convenience but we will add a NewReplica to the plot.

At the Replica, do a full backup:

```
$ xtrabackup --user=yourDBuser --password=MaGiCiGaM \
   --backup --slave-info --target-dir=/path/to/backupdir
```

By using the *--slave-info* *Percona XtraBackup* creates additional file called xtrabackup_slave_info.

Apply the logs:

```
$ xtrabackup --prepare --use-memory=2G --target-dir=/path/to/backupdir/
```

Copy the directory from the Replica to the NewReplica (**NOTE**: Make sure mysqld is shut down on the NewReplica before you copy the contents the snapshot into its *datadir*.):

```
rsync -avprP -e ssh /path/to/backupdir NewReplica:/path/to/mysql/datadir
```

For example, to set up a new user, user2, you add an additional grant on the Source:

```
> GRANT REPLICATION SLAVE ON *.*  TO 'user2'@'$newreplicaip'
 IDENTIFIED BY '$replicapass';
```

On the NewReplica, copy the configuration file from the Replica:

```
$ scp user@Replica:/etc/mysql/my.cnf /etc/mysql/my.cnf
```

Make sure you change the server-id variable in /etc/mysql/my.cnf to 3 and disable the replication on start:

```
skip-slave-start
server-id=3
```

After setting server_id, start **mysqld**.

Fetch the master_log_file and master_log_pos from the file xtrabackup_slave_info, execute the statement for setting up the source and the log file for the *NewReplica*:

```
> CHANGE MASTER TO
    MASTER_HOST='$Sourceip',
    MASTER_USER='repl',
    MASTER_PASSWORD='$replicapass',
    MASTER_LOG_FILE='Source-bin.000001',
    MASTER_LOG_POS=481;
```

If you are using version 8.0.23 or later, use CHANGE_REPLICATION_SOURCE_TO and the appropriate options. `CHANGE_MASTER_TO` is deprecated as of that version. In versions before 8.0.23, use `CHANGE MASTER TO`.

and start the replica:

```
> START SLAVE;
```

If you are using version 8.0.22 or later, use `START REPLICA` instead of `START SLAVE`. `START SLAVE` is deprecated as of that release. If you are using a version before 8.0.22 use `START SLAVE`.

If both IO and SQL threads are running when you check the `NewReplica`, server is replicating the `Source`.

25.2.2 Verifying Backups with replication and pt-checksum

One way to verify if the backup is consistent is by setting up the replication and running pt-table-checksum. This can be used to verify any type of backups, but before setting up replication, backup should be prepared and be able to run (this means that incremental backups should be merged to full backups, encrypted backups decrypted etc.).

Setting up the replication

How to setup a replica for replication in 6 simple steps with Percona XtraBackup guide provides a detailed instructions on how to take the backup and set up the replication.

For checking the backup consistency you can use either the original server where the backup was taken, or another test server created by using a different backup method (such as cold backup, mysqldump or LVM snapshots) as the source server in the replication setup.

Using pt-table-checksum

This tool is part of the *Percona Toolkit*. It performs an online replication consistency check by executing checksum queries on the source, which produces different results on replicas that are inconsistent with the source.

After you confirmed that replication has been set up successfully, you can install or download *pt-table-checksum*. This example shows downloading the latest version of *pt-table-checksum*:

```
$ wget percona.com/get/pt-table-checksum
```

Note: In order for pt-table-checksum to work correctly `libdbd-mysql-perl` will need to be installed on *Debian/Ubuntu* systems or `perl-DBD-MySQL` on *RHEL/CentOS*. If you installed the *percona-toolkit* package from the Percona repositories package manager should install those libraries automatically.

After this command has been run, *pt-table-checksum* will be downloaded to your current working directory.

Running the *pt-table-checksum* on the source will create `percona` database with the `checksums` table which will be replicated to the replicas as well. Example of the *pt-table-checksum* will look like this:

```
$ ./pt-table-checksum
    TS ERRORS  DIFFS     ROWS  CHUNKS SKIPPED    TIME TABLE
    04-30T11:31:50      0      0   633135       8       0   5.400 exampledb.aka_name
04-30T11:31:52      0      0   290859       1       0   2.692 exampledb.aka_title
Checksumming exampledb.user_info:  16% 02:27 remain
Checksumming exampledb.user_info:  34% 01:58 remain
Checksumming exampledb.user_info:  50% 01:29 remain
Checksumming exampledb.user_info:  68% 00:56 remain
Checksumming exampledb.user_info:  86% 00:24 remain
04-30T11:34:38      0      0 22187768     126       0 165.216 exampledb.user_info
04-30T11:38:09      0      0        0       1       0   0.033 mysql.time_zone_name
04-30T11:38:09      0      0        0       1       0   0.052 mysql.time_zone_
↪transition
04-30T11:38:09      0      0        0       1       0   0.054 mysql.time_zone_
↪transition_type
04-30T11:38:09      0      0        8       1       0   0.064 mysql.user
```

If all the values in the `DIFFS` column are 0 that means that backup is consistent with the current setup.

In case backup wasn't consistent *pt-table-checksum* should spot the difference and point to the table that doesn't match. Following example shows adding new user on the backed up replica in order to simulate the inconsistent backup:

```
mysql> grant usage on exampledb.* to exampledb@localhost identified by
↪'thisisnewpassword';
```

If we run the *pt-table-checksum* now difference should be spotted

```
$ ./pt-table-checksum
TS ERRORS  DIFFS     ROWS  CHUNKS SKIPPED    TIME TABLE
04-30T11:31:50      0      0   633135       8       0   5.400 exampledb.aka_name
04-30T11:31:52      0      0   290859       1       0   2.692 exampledb.aka_title
Checksumming exampledb.user_info:  16% 02:27 remain
Checksumming exampledb.user_info:  34% 01:58 remain
Checksumming exampledb.user_info:  50% 01:29 remain
Checksumming exampledb.user_info:  68% 00:56 remain
Checksumming exampledb.user_info:  86% 00:24 remain
04-30T11:34:38      0      0 22187768     126       0 165.216 exampledb.user_info
04-30T11:38:09      0      0        0       1       0   0.033 mysql.time_zone_name
04-30T11:38:09      0      0        0       1       0   0.052 mysql.time_zone_
↪transition
04-30T11:38:09      0      0        0       1       0   0.054 mysql.time_zone_
↪transition_type
04-30T11:38:09      1      0        8       1       0   0.064 mysql.user
```

This output shows that source and the replica aren't in consistent state and that the difference is in the `mysql.user` table.

More information on different options that pt-table-checksum provides can be found in the *pt-table-checksum* documentation.

25.2.3 How to create a new (or repair a broken) GTID-based Replica

MySQL 5.6 introduced the Global Transaction ID (GTID) support in replication. *Percona XtraBackup* automatically stores the `GTID` value in the `xtrabackup_binlog_info` when doing the backup of *MySQL* and *Percona Server for MySQL* 5.7 with the `GTID` mode enabled. This information can be used to create a new (or repair a broken) `GTID`-based replica.

STEP 1: Take a backup from any server on the replication environment, source or replica

The following command takes a backup and saves it in the `/data/backups/$TIMESTAMP` folder:

```
$ xtrabackup --backup --target-dirs=/data/backups/
```

In the destination folder, there will be a file with the name `xtrabackup_binlog_info`. This file contains both binary log coordinates and the `GTID` information.

```
$ cat xtrabackup_binlog_info
mysql-bin.000002    1232        c777888a-b6df-11e2-a604-080027635ef5:1-4
```

That information is also printed by xtrabackup after taking the backup:

```
xtrabackup: MySQL binlog position: filename 'mysql-bin.000002', position 1232, GTID␣
↪of the last change 'c777888a-b6df-11e2-a604-080027635ef5:1-4'
```

STEP 2: Prepare the backup

The backup will be prepared with the following command on the Source:

```
$ xtrabackup --prepare --target-dir=/data/backup
```

You need to select the path where your snapshot has been taken, for example `/data/backups/2013-05-07_08-33-33`. If everything is ok you should get the same OK message. Now, the transaction logs are applied to the data files, and new ones are created: your data files are ready to be used by the MySQL server.

STEP 3: Move the backup to the destination server

Use **`rsync`** or **`scp`** to copy the data to the destination server. If you are synchronizing the data directly to the already running replica's data directory it is advised to stop the *MySQL* server there.

```
$ rsync -avprP -e ssh /path/to/backupdir/$TIMESTAMP NewSlave:/path/to/mysql/
```

After you copy the data over, make sure *MySQL* has proper permissions to access them.

```
$ chown mysql:mysql /path/to/mysql/datadir
```

STEP 4: Configure and start replication

Set the gtid_purged variable to the `GTID` from `xtrabackup_binlog_info`. Then, update the information about the source node and, finally, start the replica. Run the following commands on the replica if you are using a version before 8.0.22:

```
# Using the mysql shell
 > SET SESSION wsrep_on = 0;
 > RESET MASTER;
 > SET SESSION wsrep_on = 1;
 > SET GLOBAL gtid_purged='<gtid_string_found_in_xtrabackup_binlog_info>';
 > CHANGE MASTER TO
             MASTER_HOST="$masterip",
             MASTER_USER="repl",
             MASTER_PASSWORD="$slavepass",
             MASTER_AUTO_POSITION = 1;
 > START SLAVE;
```

If you are using version 8.0.22 or later, use `START REPLICA` instead of `START SLAVE`. `START SLAVE` is deprecated as of that release. If you are using version 8.0.21 or earlier, use `START SLAVE`.

If you are using a version 8.0.23 or later, run the following commands:

```
# Using the mysql shell
 > SET SESSION wsrep_on = 0;
 > RESET MASTER;
 > SET SESSION wsrep_on = 1;
 > SET GLOBAL gtid_purged='<gtid_string_found_in_xtrabackup_binlog_info>';
 > CHANGE REPLICATION SOURCE TO
             SOURCE_HOST="$masterip",
             SOURCE_USER="repl",
             SOURCE_PASSWORD="$slavepass",
             SOURCE_AUTO_POSITION = 1;
 > START REPLICA;
```

If you are using version 8.0.23 or later, use CHANGE_REPLICATION_SOURCE_TO and the appropriate options. `CHANGE_MASTER_TO` is deprecated as of that release.

Note: The example above is applicable to Percona XtraDB Cluster. The `wsrep_on` variable is set to *0* before resetting the source (`RESET MASTER`). The reason is that Percona XtraDB Cluster will not allow resetting the source if `wsrep_on=1`.

STEP 5: Check the replication status

The following command returns the replica status:

```
SHOW REPLICA STATUS\G
[..]
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
[...]
Retrieved_Gtid_Set: c777888a-b6df-11e2-a604-080027635ef5:5
Executed_Gtid_Set: c777888a-b6df-11e2-a604-080027635ef5:1-5
```

Note: The command SHOW SLAVE STATUS is deprecated. Use SHOW REPLICA STATUS.

We can see that the replica has retrieved a new transaction with number 5, so transactions from 1 to 5 are already on this slave.

We have created a new replica in our `GTID` based replication environment.

25.3 Auxiliary Guides

25.3.1 Enabling the server to communicate via TCP/IP

Most of the Linux distributions do not enable by default to accept TCP/IP connections from outside in their MySQL or Percona Server packages.

You can check it with `netstat` on a shell:

```
$ netstat -lnp | grep mysql
tcp         0        0 0.0.0.0:3306 0.0.0.0:* LISTEN 2480/mysqld
unix 2 [ ACC ] STREAM LISTENING 8101 2480/mysqld /tmp/mysql.sock
```

You should check two things:

- there is a line starting with `tcp` (the server is indeed accepting TCP connections) and
- the first address (`0.0.0.0:3306` in this example) is different than `127.0.0.1:3306` (the bind address is not localhost's).

In the first case, the first place to look is the `my.cnf` file. If you find the option `skip-networking`, comment it out or just delete it. Also check that *if* the variable `bind_address` is set, then it shouldn't be set to localhost's but to the host's IP. Then restart the MySQL server and check it again with `netstat`. If the changes you did had no effect, then you should look at your distribution's startup scripts (like `rc.mysqld`). You should comment out flags like `--skip-networking` and/or change the `bind-address`.

After you get the server listening to remote TCP connections properly, the last thing to do is checking that the port (3306 by default) is indeed open. Check your firewall configurations (`iptables -L`) and that you are allowing remote hosts on that port (in `/etc/hosts.allow`).

And we're done! We have a MySQL server running which is able to communicate with the world through TCP/IP.

25.3.2 Privileges and Permissions for Users

We will be referring to *permissions* to the ability of a user to access and perform changes on the relevant parts of the host's filesystem, starting/stopping services and installing software.

By *privileges* we refer to the abilities of a database user to perform different kinds of actions on the database server.

At a system level

There are many ways for checking the permission on a file or directory. For example, `ls -ls /path/to/file` or `stat /path/to/file | grep Access` will do the job:

```
$ stat /etc/mysql | grep Access
Access: (0755/drwxr-xr-x)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2011-05-12 21:19:07.129850437 -0300
$ ls -ld /etc/mysql/my.cnf
-rw-r--r-- 1 root root 4703 Apr  5 06:26 /etc/mysql/my.cnf
```

As in this example, `my.cnf` is owned by `root` and not writable for anyone else. Assuming that you do not have `root` 's password, you can check what permissions you have on this types of files with `sudo -l`:

```
$ sudo -l
Password:
You may run the following commands on this host:
(root) /usr/bin/
(root) NOPASSWD: /etc/init.d/mysqld
(root) NOPASSWD: /bin/vi /etc/mysql/my.cnf
(root) NOPASSWD: /usr/local/bin/top
(root) NOPASSWD: /usr/bin/ls
(root) /bin/tail
```

Being able to execute with `sudo` scripts in `/etc/init.d/`, `/etc/rc.d/` or `/sbin/service` is the ability to start and stop services.

Also, If you can execute the package manager of your distribution, you can install or remove software with it. If not, having `rwx` permission over a directory will let you do a local installation of software by compiling it there. This is a typical situation in many hosting companies' services.

There are other ways for managing permissions, such as using *PolicyKit*, *Extended ACLs* or *SELinux*, which may be preventing or allowing your access. You should check them in that case.

At a database server level

To query the privileges that your database user has been granted, at a console of the server execute:

```
mysql> SHOW GRANTS;
```

or for a particular user with:

```
mysql> SHOW GRANTS FOR 'db-user'@'host';
```

It will display the privileges using the same format as for the GRANT statement.

Note that privileges may vary across versions of the server. To list the exact list of privileges that your server support (and a brief description of them) execute:

```
mysql> SHOW PRIVILEGES;
```

25.3.3 Installing and configuring a SSH server

Many Linux distributions only install the ssh client by default. If you don't have the ssh server installed already, the easiest way of doing it is by using your distribution's packaging system:

```
   ubuntu$ sudo apt install openssh-server
archlinux$ sudo pacman -S openssh
```

You may need to take a look at your distribution's documentation or search for a tutorial on the internet to configure it if you haven't done it before.

Some links of them are:

- Debian - http://wiki.debian.org/SSH
- Ubuntu - https://help.ubuntu.com/10.04/serverguide/C/openssh-server.html
- Archlinux - https://wiki.archlinux.org/index.php/SSH
- Fedora - http://docs.fedoraproject.org/en-US/Fedora/12/html/Deployment_Guide/s1-openssh-server-config.html
- CentOS - http://www.centos.org/docs/5/html/Deployment_Guide-en-US/s1-openssh-server-config.html
- RHEL - http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Deployment_Guide/ch-OpenSSH.html

25.4 Assumptions in this section

Most of the times, the context will make the recipe or tutorial understandable. To assure that, a list of the assumptions, names and "things" that will appear in this section is given. At the beginning of each recipe or tutorial they will be specified in order to make it quicker and more practical.

`HOST`

A system with a *MySQL*-based server installed, configured and running. We will assume the following about this system:

- the MySQL server is able to *communicate with others by the standard TCP/IP port*;
- a SSH server is installed and configured - see *here* if it is not;
- you have an user account in the system with the appropriate *permissions* and
- you have a MySQL's user account with appropriate *Connection and Privileges Needed*.

`USER` An user account in the system with shell access and appropriate permissions for the task. A guide for checking them is *here*.

`DB-USER` An user account in the database server with appropriate privileges for the task. A guide for checking them is *here*.

- *Recipes for xtrabackup*
- *How-Tos*
- *Auxiliary Guides*

Part X

Release notes

CHAPTER

TWENTYSIX

PERCONA XTRABACKUP 8.0 RELEASE NOTES

26.1 *Percona XtraBackup* 8.0.27-19

Date February 2, 2022

Installation Installing Percona XtraBackup

Percona XtraBackup enables MySQL backups without blocking user queries, making it ideal for companies with large data sets and mission-critical applications that cannot tolerate long periods of downtime. Offered free as an open source solution, it drives down backup costs while providing unique features for MySQL backups.

26.1.1 Release Highlights

The following list contains a number of the bug fixes for *MySQL* 8.0.27, provided by Oracle, and included in Percona Server for MySQL:

- The `default_authentication_plugin` is deprecated. Support for this plugin may be removed in future versions. Use the `authentication_policy` variable.
- The `binary` operator is deprecated. Support for this operator may be removed in future versions. Use `CAST(... AS BINARY)`.
- When a parent table updated or deleted a row, this operation initiated a cascading `SET NULL` operation on the child table. On the child table, a virtual column value could be set to NULL instead of the value derived from the base column value.

Find the full list of bug fixes and changes in the MySQL 8.0.27 Release Notes.

26.1.2 New Features

- PXB-1883: Added support for Microsoft Azure Cloud Storage in the xbcloud binary. (Thanks to Ivan for reporting this issue)

26.1.3 Improvements

- PXB-2434: Use MySQL page tracking for incremental backup

26.1.4 Bugs Fixed

- PXB-1741: PS startup displays errors for databases excluded during partial backup
- PXB-2614: The `xbstream` binary was enhanced to support sparse files on the XFS file system.
- PXB-2608: Upgrade the Vault API to V2 (Thanks to Benedito Marques Magalhaes for reporting this issue)
- PXB-2543: Fix that adds `IB_EXPORT_CFG_VERSION_V6` when exporting a `.cfg` file for a table (Thanks to Peng Gao for reporting this issue)
- PXB-2465: Fix for querying the `ps.log_status` when group replication channels are items on that list. XtraBackup skips the name if it matches either `group_replication_applier``or ``group_replication_recovery`.(Thanks to Galamb Gergő for reporting this issue)
- PXB-2625: The default version of CURL in Debian Buster failed to identify a TLS HTTP2 connection as closed and attempted to reuse the connection. (Thanks to Johan Andersson for reporting this issue)

26.2 *Percona XtraBackup* 8.0.26-18.0

Date September 2, 2021

Installation Installing Percona XtraBackup

Percona XtraBackup enables MySQL backups without blocking user queries, making it ideal for companies with large data sets and mission-critical applications that cannot tolerate long periods of downtime. Offered free as an open source solution, it drives down backup costs while providing unique features for MySQL backups.

26.2.1 Improvements

- PXB-2477: The xbcloud should retry on error and utilize incremental backoff (Thanks to Baptiste Mille-Mathias for reporting this issue)
- PXB-2580: With the xbcloud binary, a chunk-upload on an SSL connect error to S3 was not retried (Thanks to Tim Vaillancourt for providing the patch)
- PXB-2317: Remove the obsolete LOCKLESS binary log functionality since the `performance_schema.log_status` table is now used to get the log information on all the storages without locking them

26.2.2 Bugs Fixed

- PXB-2101: The Prepare step fails due to duplicate Serialized Dictionary Information (SDI) (Thanks to Fungo Wang for reporting this issue)
- PXB-1504: The FIND_GCRYPT macro is broken (Thanks to Maxim Bublis for reporting this issue)
- PXB-1961: The Prepare step of a full backup fails for encrypted tables with a generic error
- PXB-2585: XtraBackup fails to backup archive RocksDB Write-Ahead Log (WAL) files

26.3 *Percona XtraBackup* 8.0.25-17.0

Date June 3, 2021

Installation Installing Percona XtraBackup

Percona XtraBackup enables MySQL backups without blocking user queries, making it ideal for companies with large data sets and mission-critical applications that cannot tolerate long periods of downtime. Offered free as an open source solution, it drives down backup costs while providing unique features for MySQL backups.

26.3.1 Bugs Fixed

- PXB-2442 : Backup cannot be decompressed using the AppArmor profile
- PXB-2444 : The xbcloud binary fails to upload backup with enforcing SELinux mode
- PXB-2443 : Version check fails with AppArmor profile
- PXB-2445 : Initializing the libgcrypt in xbcloud
- PXB-2455 : XtraBackup prepare fails if the checkpoint LSN is greater than the last LSN
- PXB-2473 : SELinux errors in audit logs
- PXB-1462 : Long gtid_executed breaks —history functionality
- PXB-2369 : Issues with “Installing” page in XtraBackup documentation
- PXB-2457 : Incorrect binlog names if binlog name contains periods
- PXB-2106 : Copy-back creates wrong binlog.index

26.4 *Percona XtraBackup* 8.0.23-16.0

Date March 22, 2021

Installation Installing Percona XtraBackup

Percona XtraBackup enables MySQL backups without blocking user queries, making it ideal for companies with large data sets and mission-critical applications that cannot tolerate long periods of downtime. Offered free as an open source solution, it drives down backup costs while providing unique features for MySQL backups.

This release fixes the security vulnerability CVE-2020-29488

26.4.1 Improvements

- PXB-2280: Provide SELinux and AppArmor default policies
- PXB-1979: Enable –lock-ddl by default to prevent corruption of the backup

26.4.2 Bugs Fixed

- PXB-2274: Correct restore processing when there are DML statements running during backup stage by writing the last_checkpoint and LSN from ps.log_status instead of the redo log (Thanks to user Li Biao for reporting this issue)
- PXB-2430: Correct build failure by removing the dependency of no-server-version-check with version-check (Thanks to user agarner for reporting this issue)
- PXB-2415: Add build dependencies to correct Debian/Ubuntu packages in docker (Thanks to user Matt Cole for reporting this issue)
- PXB-2429: Correct Backup failure for encrypted tablespace by skipping the encryption reset operation during the redo scan phase
- PXB-2418: Remove PROTOBUF_LITE_LIBRARY - it is not used and is not needed
- PXB-2395: Update versions for xbstream and xbcrypt
- PXB-2394: Correct spellings in xbcloud help
- PXB-2357: Correct hang in backup with redo log archive by using block number instead of checksum (checksum of redo file and archive file can be different)
- PXB-2180: Correct incremental prepare failure with logical redo by skipping the apply of logical redos (MLOG_TABLE_DYNAMIC_META) during the incremental prepare (except the last prepare).

26.5 *Percona XtraBackup* 8.0.22-15.0

Date December 14, 2020

Installation Installing Percona XtraBackup

Percona XtraBackup enables MySQL backups without blocking user queries, making it ideal for companies with large data sets and mission-critical applications that cannot tolerate long periods of downtime. Offered free as an open source solution, it drives down backup costs while providing unique features for MySQL backups.

Note: The naming structure for the releases has changed. See the aligning Percona XtraBackup versions with Percona Server for MySQL blog post for more information.

26.5.1 New Features

- PXB-2112: xbcloud: support storage_class option with –storage=s3 (Thanks to user rluisr for reporting this issue)
- PXB-2242: Prevent back up if Source system (DB) version is higher then current PXB version

26.5.2 Improvements

- PXB-2254: Redesign –lock-ddl-per-table

26.5.3 Bugs Fixed

- PXB-793: Fix syntax error when executing –lock-ddl-per-table queries
- PXB-953: Improve stdout for the end of usage of –lock-ddl-per-table
- PXB-787: Modify scan to provide correct status when finding corrupted tablespace
- PXB-2314: Handle Disk format changes introduced in MySQL 8.0.22 release
- PXB-2279: Modify to look for prefixes in xtrabackup_tablespaces* without extensions (Thanks to user mrmainnet for reporting this issue)
- PXB-2336: Modify to check to see if first block is an all-zero block and process accordingly
- PXB-2275: Modify backup processing to add validations if an encrypted table is created
- PXB-2272: Modify regexp to consider all #sql as temporary tables
- PXB-2257: Modify –lock-ddl-per-table to properly close database connection

26.6 *Percona XtraBackup* 8.0.14

Date August 31, 2020

Installation Installing Percona XtraBackup

Percona XtraBackup enables MySQL backups without blocking user queries, making it ideal for companies with large data sets and mission-critical applications that cannot tolerate long periods of downtime. Offered free as an open source solution, it drives down backup costs while providing unique features for MySQL backups.

Percona XtraBackup 8.0.14 supports backup and restore processing for all versions of MySQL and has been tested with the latest MySQL 8.0.21.

26.6.1 Improvements

- PXB-1605: Document how to use Percona Xtrabackup with Docker
- PXB-2252: Introduce debug option to print redo log records scanned and applied

26.6.2 Bugs Fixed

- PXB-2215: Modify Import tablespace process to correctly calculate pack_length
- PXB-2114: Document when table is ignored by full backup it is not automatically ignored for incremental backup
- PXB-2255: Modify processing to Error out if undo truncation during backup
- PXB-2249: Verify perl binary exists before completing version check
- PXB-2243: Correct processing when undo truncation happens between full backup and incremental
- PXB-2238: Provide *binary tarball* with shared libs and glibc suffix & minimal tarballs

- PXB-2202: Modify Xbcloud to display an error when xtrabackup fails to create a backup

26.7 *Percona XtraBackup* 8.0.13

Date June 17, 2020

Installation Installing Percona XtraBackup

Percona XtraBackup enables MySQL backups without blocking user queries, making it ideal for companies with large data sets and mission-critical applications that cannot tolerate long periods of downtime. Offered free as an open source solution, it drives down backup costs while providing unique features for MySQL backups.

Percona XtraBackup 8.0.13 supports backup and restore processing for all versions of MySQL and has been tested with the latest MySQL 8.0.20.

26.7.1 Bugs Fixed

- PXB-2165: Modify xbcloud to store backups using s3 access key parameters if AWS access key env variables are set
- PXB-2164: Modify xbcloud to return the error when the backup doesn't exist in s3 bucket
- PXB-2127: Modify xbcloud to upload backups with empty database to min.io storage
- PXB-2198: Modify xbcloud delete to return the error when the backup doesn't exist in s3 bucket
- PXB-2155: Correct corruption when redo logs are encrypted using xtrabackup –backup –compress=lz4
- PXB-2166: Modify xbcloud to check for bucket existence and return 'ok' status when domain name can be resolved

26.8 *Percona XtraBackup* 8.0.12

Date May 27, 2020

Installation Installing Percona XtraBackup

Percona XtraBackup enables MySQL backups without blocking user queries, making it ideal for companies with large data sets and mission-critical applications that cannot tolerate long periods of downtime. Offered free as an open source solution, it drives down backup costs while providing unique features for MySQL backups.

Percona XtraBackup 8.0.12 now supports backup and restore processing for all versions of MySQL; previous versions of Percona XtraBackup will not work with MySQL 8.0.20 and higher.

26.8.1 Bugs Fixed

- PXB-2133: Correct prtype stored for DECIMAL columns in .cfg file by –export
- PXB-2146: Amazon S3 session token support added to xbcloud
- PXB-2179: Modify 'xtrabackup prepare' to shut down keyring plugin when run with –generate-transition-key
- PXB-2157: Modify 'xtrabackup –prepare' to generate cfg files for partition tables when –export is used

26.9 Percona XtraBackup 8.0.11

Date April 13, 2020

Installation *Installing Percona XtraBackup 8.0*

Downloads are available from our download site and from *apt* and *yum* repositories.

Percona XtraBackup enables MySQL backups without blocking user queries, making it ideal for companies with large data sets and mission-critical applications that cannot tolerate long periods of downtime. Offered free as an open source solution, it drives down backup costs while providing unique features for *MySQL* backups.

All Percona software is open-source and free.

This release fixes security vulnerability *CVE-2020-10997*.

26.10 Percona XtraBackup 8.0.10

Date March 16, 2020

Installation *Installing Percona XtraBackup 8.0*

Downloads are available from our download site and from *apt* and *yum* repositories.

Percona XtraBackup enables MySQL backups without blocking user queries, making it ideal for companies with large data sets and mission-critical applications that cannot tolerate long periods of downtime. Offered free as an open source solution, it drives down backup costs while providing unique features for *MySQL* backups.

All Percona software is open-source and free.

26.10.1 Bugs Fixed

- PXB-1982: The *history* table showed a wrong value for `lock_time`.
- PXB-2099: `copy-back` did not move undo files to the *data directory*.
- PXB-2107: If the *undo tablespace* was created in a directory which is a symbolic link to another directory then the backup failed at backup stage.
- PXB-2118: Undo log file was renamed incorrectly to undo tablespace name after restore

26.11 Percona XtraBackup 8.0.9

Percona is glad to announce the release of Percona XtraBackup 8.0.9 on December 16, 2019. Downloads are available from our download site and from *apt* and *yum* repositories.

Percona XtraBackup enables MySQL backups without blocking user queries, making it ideal for companies with large data sets and mission-critical applications that cannot tolerate long periods of downtime. Offered free as an open source solution, it drives down backup costs while providing unique features for *MySQL* backups.

All Percona software is open-source and free.

26.11.1 Bugs Fixed

- Sometime between December 3rd and December 10th, a change was introduced in AWS (Amazon Web Services) that caused an incompatibility with our Percona XtraBackup `xbcloud` utility. Bug fixed PXB-1978.

26.12 Percona XtraBackup 8.0.8

Percona is glad to announce the release of Percona XtraBackup 8.0.8 on November 21, 2019. Downloads are available from our download site and from *apt* and *yum* repositories.

Percona XtraBackup enables MySQL backups without blocking user queries, making it ideal for companies with large data sets and mission-critical applications that cannot tolerate long periods of downtime. Offered free as an open source solution, it drives down backup costs while providing unique features for *MySQL* backups.

All Percona software is open-source and free.

26.12.1 New Features and Improvements

- Support log archiving feature in PXB 8.0. More information in PXB-1912 and in the Redo Log section of MySQL documentation
- For the MyRocks storage engine, support the creation of renewable checkpoints (controlled via `--rocksdb-checkpoint-max-age` and `--rocksdb-checkpoint-max-count`) to minimize the amount of binary logs to apply after the backup was completed. Using renewable checkpoints, Percona XtraBackup only copies the SST files that were created after the previous checkpoint. More information in PXB-1915.
- Two options (*--backup-lock-timeout* and *--backup-lock-retry-count*) were added to enable the configuring of the timeout for acquiring metadata locks in `FLUSH TABLES WITH READ LOCK`, `LOCK TABLE FOR BACKUP`, and `LOCK BINLOG FOR BACKUP` statements. More information in PXB-1914

26.12.2 Bugs Fixed

- An encrypted table could not be restored when `ADD INDEX` or `DROP INDEX` commands had been run on the table. Bug fixed PXB-1905
- In some cases `xtrabackup --prepare` could fail to decrypt a table but reported that the operation completed ok. Bug fixed PXB-1936
- `xtrabackup --move-back` did not complete successfully when the encrypted binlog file. Bug fixed PXB-1937.
- Percona XtraBackup could crash during the prepare stage when making an incremental backup when a multi valued index was being added or dropped for JSON data. Bug fixed PXB-1913.

Other bugs fixed: PXB-1928, PXB-1938, PXB-1951, PXB-1953, PXB-1954.

26.13 Percona XtraBackup 8.0.7

Percona is glad to announce the release of Percona XtraBackup 8.0.7 on August 7, 2019. Downloads are available from our download site and from *apt* and *yum* repositories.

Percona XtraBackup enables MySQL backups without blocking user queries, making it ideal for companies with large data sets and mission-critical applications that cannot tolerate long periods of downtime. Offered free as an open source solution, it drives down backup costs while providing unique features for *MySQL* backups.

In release 8.0.7, Percona XtraBackup enables making backups of databases that contain the encrypted system tablespace. Encrypted *mysql* tablespace is now also supported.

Percona XtraBackup 8.0.7 implements the support of the `lz4` compression algorithm so that you could make compressed backups using *lz4* (*–compress=lz4*) in addition to the default *quicklz* method.

All Percona software is open-source and free.

26.13.1 New Features and Improvements

- Add support of the system tablespace encryption. More information in PXB-1649.
- Implemented the support of the *lz4* compression algorithm. More information in PXB-1857.

26.13.2 Bugs Fixed

- When the *encrypted tablespaces* feature was enabled, encrypted and compressed tables were not usable on the joiner node (Percona XtraDB Cluster) via SST (State Snapshot Transfer) with the *xtrabackup-v2* method. Bug fixed PXB-1867.
- `xbcloud` did not update date related fields of the HTTP header when retrying a request. Bug fixed PXB-1874.
- `xbcloud` did not retry to send the request after receiving the HTTP 408 error (request timeout). Bug fixed PXB-1875.
- `xtrabackup` did not accept decimal fractions as values of the `innodb_max_dirty_pages_pct` option. Bug fixed PXB-1807.
- If the user tried to merge an already prepared incremental backup, a misleading error was produced without informing that incremental backups may not be used twice. Bug fixed PXB-1862.

Other bugs fixed: PXB-1493, PXB-1557, PXB-1887, PXB-1870, PXB-1879, PXB-1901.

26.14 Percona XtraBackup 8.0.6

Percona is glad to announce the release of Percona XtraBackup 8.0.6 on May 9, 2019. Downloads are available from our download site and from *apt* and *yum* repositories.

Percona XtraBackup enables MySQL backups without blocking user queries, making it ideal for companies with large data sets and mission-critical applications that cannot tolerate long periods of downtime. Offered free as an open source solution, it drives down backup costs while providing unique features for *MySQL* backups.

In version 8.0.6, Percona XtraBackup introduces the support of the MyRocks storage engine with Percona Server for MySQL version 8.0.15-6 or higher.

Percona XtraBackup 8.0.6 enables saving backups to an Amazon S3, MinIO, and Google Cloud Storage (using interoperability mode) when using xbcloud. The following example demonstrates how to use an Amazon S3 storage to make a full backup:

```
  $ xtrabackup --backup --stream=xbstream --extra-lsndir=/tmp --target-dir=/tmp | \
  xbcloud put --storage=s3 \
  --s3-endpoint='s3.amazonaws.com' \
  --s3-access-key='YOUR-ACCESSKEYID' \
  --s3-secret-key='YOUR-SECRETACCESSKEY' \
  --s3-bucket='mysql_backups'
  --parallel=10 \
  ${date -I}-full_backup

All |percona| software is open-source and free
```

26.14.1 New Features

- The MyRocks storage engine is now supported with Percona XtraBackup. More information in PXB-1754.
- Amazon S3 is now supported in xbcloud. More information in PXB-1813.

26.14.2 Bugs Fixed

- Percona XtraBackup could fail to restore the undo tablespace created during or before incremental backup. Bug fixed PXB-1780.
- A backup could fail if `log_bin_index` was defined in `my.cnf`. Bug fixed PXB-1801.
- When the row format was changed during the backup, xtrabackup could crash during the incremental prepare stage. Bug fixed PXB-1824.
- During the `prepare` phase, *Percona XtraBackup* could freeze and never finish execution. Bug fixed PXB-1819.
- Percona XtraBackup could crash during the `prepare` stage when making a backup of a host running MySQL Server v8.0.16. Bug fixed PXB-1839.

Other bugs fixed: PXB-1809, PXB-1810, PXB-1832, PXB-1837.

26.15 Percona XtraBackup 8.0.5

Percona is glad to announce the release of Percona XtraBackup 8.0.5 on March 4, 2019. Downloads are available from our download site and from *apt* and *yum* repositories.

Percona XtraBackup enables MySQL backups without blocking user queries, making it ideal for companies with large data sets and mission-critical applications that cannot tolerate long periods of downtime. Offered free as an open source solution, it drives down backup costs while providing unique features for *MySQL* backups.

Percona XtraBackup 8.0.5 introduces the support of undo tablespaces created using the new syntax (`CREATE UNDO TABLESPACE`) available since MySQL 8.0.14. Percona XtraBackup also supports the binary log encryption introduced in *MySQL* 8.0.14.

Two new options were added to *xbstream*. Use the *`--decompress`* option with *xbstream* to decompress individual qpress files. With the *`--decompress-threads`* option, specify the number of threads to apply when decompressing. Thanks to Rauli Ikonen for this contribution.

This release of Percona XtraBackup is a **General Availability** release ready for use in a production environment.

All Percona software is open-source and free.

Please note the following about this release:

- The deprecated innobackupex has been removed. Use the xtrabackup command to back up your instances: `$ xtrabackup --backup --target-dir=/data/backup`
- When migrating from earlier database server versions, backup and restore and using Percona XtraBackup 2.4 and then use `mysql_upgrade` from *MySQL* 8.0.x
- If using `yum` or `apt` repositories to install *Percona XtraBackup* 8.0.5, ensure that you have enabled the new *tools* repository. You can do this with the **`percona-release enable tools release`** command and then install the *percona-xtrabackup-80* package.

26.15.1 New Features

- PXB-1548: Percona XtraBackup enables updating the `ib_buffer_pool` file with the latest pages present in the buffer pool using the *`--dump-innodb-buffer-pool`* option. Thanks to Marcelo Altmann for contribution.
- PXB-1768: Added support for undo tablespaces created with the new *MySQL* 8.0.14 syntax.
- PXB-1781: Added support for binary log encryption introduced in *MySQL* 8.0.14.
- PXB-1797: For *xbstream*, two new options were added. The *`--decompress`* option enables *xbstream* to decompress individual qpress files. The *`--decompress-threads`* option controls the number of threads to apply when decompressing. Thanks to Rauli Ikonen for this contribution.

26.15.2 Bugs Fixed

- Using `--lock-ddl-per-table` caused the server to scan all records of partitioned tables which could lead to the "out of memory" error. Bugs fixed PXB-1691 and PXB-1698.
- When Percona XtraBackup was started run with the *`--slave-info`*, incorrect coordinates were written to the xtrabackup_slave_info file.. Bug fixed PXB-1737
- Percona XtraBackup could crash at the prepare stage when making an incremental backup if the variable `innodb-rollback-segments` was changed after starting the *MySQL* Server. Bug fixed PXB-1785.
- The full backup could fail when *Percona Server for MySQL* was started with the `--innodb-encrypt-tables` parameter. Bug fixed PXB-1793.

Other bugs fixed: PXB-1632, PXB-1715, PXB-1770, PXB-1771, PXB-1773.

26.16 Percona XtraBackup 8.0.4

Percona is glad to announce the release of Percona XtraBackup 8.0.4 on December 10, 2018. Downloads are available from our download site and from *apt* and *yum* repositories.

Percona XtraBackup enables MySQL backups without blocking user queries, making it ideal for companies with large data sets and mission-critical applications that cannot tolerate long periods of downtime. Offered free as an open source solution, it drives down backup costs while providing unique features for *MySQL* backups.

This release of Percona Xtrabackup is a **General Availability** release ready for use in a production environment.

Please note the following about this release:

- The deprecated innobackupex has been removed. Use the xtrabackup command to back up your instances: `$ xtrabackup --backup --target-dir=/data/backup`
- When migrating from earlier database server versions, backup and restore and using Percona XtraBackup 2.4 and then use `mysql_upgrade` from *MySQL* 8.0.x
- If using `yum` or `apt` repositories to install *Percona XtraBackup* 8.0.4, ensure that you have enabled the new *tools* repository. You can do this with the **`percona-release enable tools release`** command and then install the *percona-xtrabackup-80* package.

All Percona software is open-source and free. We are grateful to the community for the invaluable contributions to *Percona XtraBackup*. We would especially like to highlight the input of *Alexey Kopytov* who has been actively offering improvements and submitting bug reports for *Percona XtraBackup*.

26.16.1 New Features

- *Percona XtraBackup* 8.0.4 is based on *MySQL* 8.0.13 and fully supports Percona Server for *MySQL* 8.0 series and *MySQL* 8.0 series.

26.16.2 Bugs Fixed

- PXB-1699: *xtrabackup –prepare* could fail on backups of MySQL 8.0.13 databases
- PXB-1704: *xtrabackup –prepare* could hang while performing insert buffer merge
- PXB-1668: When the *–throttle* option was used, the applied value was different from the one specified by the user (off by one error)
- PXB-1679: PXB could crash when `ALTER TABLE ... TRUNCATE PARTITION` command was run during a backup without locking DDL

26.17 *Percona XtraBackup* 8.0-3-rc1

Percona is glad to announce the release of *Percona XtraBackup* 8.0-3-rc1 on October 31 2018. Downloads are available from our download site and from *apt* and *yum* repositories.

This is a **Release Candidate** quality release and it is not intended for production. If you want a high quality, Generally Available release, use the current Stable version (the most recent stable version at the time of writing is 2.4.12 in the 2.4 series).

Things to Note

- `innobackupex` was previously deprecated and has been removed
- Due to the new MySQL redo log and data dictionary formats the Percona XtraBackup 8.0.x versions will only be compatible with MySQL 8.0.x and the upcoming Percona Server for MySQL 8.0.x
- For experimental migrations from earlier database server versions, you will need to backup and restore and using XtraBackup 2.4 and then use `mysql_upgrade` from MySQL 8.0.x

26.17.1 Improvements

- PXB-1655: The `--lock-ddl` option is supported when backing up MySQL 8

26.17.2 Bugs Fixed

- PXB-1678: Incremental backup prepare with the `--apply-log-only` option could roll back uncommitted transactions
- PXB-1672: The MTS slave without GTID could be backed up when the `--safe-slave-backup` option was applied.

Part XI

References

CHAPTER

TWENTYSEVEN

FREQUENTLY ASKED QUESTIONS

27.1 Does *Percona XtraBackup* 8.0 support making backups of databases in versions prior to 8.0?

Percona XtraBackup 8.0 does not support making backups of databases created in versions prior to 8.0 of *MySQL*, *Percona Server for MySQL* or *Percona XtraDB Cluster*. As the changes that *MySQL* 8.0 introduced in *data dictionaries*, *redo log* and *undo log* are incompatible with previous versions, it is currently impossible for *Percona XtraBackup* 8.0 to also support versions prior to 8.0.

Due to changes in MySQL 8.0.20 released by Oracle at the end of April 2020, *Percona XtraBackup* 8.0, up to version 8.0.11, is not compatible with MySQL version 8.0.20 or higher, or Percona products that are based on it: Percona Server for MySQL and Percona XtraDB Cluster.

For more information, see Percona XtraBackup 8.x and MySQL 8.0.20

27.2 Why will `innobackupex` not run in *Percona XtraBackup* 8.0?

`innobackupex` has been removed from *Percona XtraBackup* '8.0' in favor of **`xtrabackup`**.

27.3 What's the difference between `innobackupex` and `xtrabackup`?

See *Why will innobackupex not run in Percona XtraBackup 8.0?*

27.4 Are you aware of any web-based backup management tools (commercial or not) built around *Percona XtraBackup*?

Zmanda Recovery Manager is a commercial tool that uses *Percona XtraBackup* for Non-Blocking Backups:

> *"ZRM provides support for non-blocking backups of MySQL using |Percona XtraBackup|. ZRM with |Percona XtraBackup| provides resource utilization management by providing throttling based on the number of IO operations per second. |Percona XtraBackup| based backups also allow for table level recovery even though the backup was done at the database level (needs the recovery database server to be |Percona Server| with XtraDB)."*

27.5 `xtrabackup` binary fails with a floating point exception

In most of the cases this is due to not having install the required libraries (and version) by **`xtrabackup`**. Installing the *GCC* suite with the supporting libraries and recompiling **`xtrabackup`** will solve the issue. See *Compiling and Installing from Source Code* for instructions on the procedure.

27.6 How xtrabackup handles the ibdata/ib_log files on restore if they aren't in mysql datadir?

In case the `ibdata` and `ib_log` files are located in different directories outside of the datadir, you will have to put them in their proper place after the logs have been applied.

27.7 Backup fails with Error 24: 'Too many open files'

This usually happens when database being backed up contains large amount of files and *Percona XtraBackup* can't open all of them to create a successful backup. In order to avoid this error the operating system should be configured appropriately so that *Percona XtraBackup* can open all its files. On Linux, this can be done with the `ulimit` command for specific backup session or by editing the `/etc/security/limits.conf` to change it globally (**NOTE**: the maximum possible value that can be set up is `1048576` which is a hard-coded constant in the Linux kernel).

27.8 How to deal with skipping of redo logs for DDL operations?

To prevent creating corrupted backups when running DDL operations, Percona XtraBackup aborts if it detects that redo logging is disabled. In this case, the following error is printed:

```
[FATAL] InnoDB: An optimized (without redo logging) DDL operation has been performed.␣
↪All modified pages may not have been flushed to the disk yet.
Percona XtraBackup will not be able to take a consistent backup. Retry the backup␣
↪operation.
```

Note: Redo logging is disabled during a sorted index build

To avoid this error, Percona XtraBackup can use metadata locks on tables while they are copied:

- To block all DDL operations, use the `--lock-ddl` option that issues `LOCK TABLES FOR BACKUP`.
- If `LOCK TABLES FOR BACKUP` is not supported, you can block DDL for each table before XtraBackup starts to copy it and until the backup is completed using the `--lock-ddl-per-table` option.

  Note: As of *Percona XtraBackup* 8.0.15, the *–lock-ddl-per-table* option is deprecated. Use the *–lock-ddl* option.

See also:

lock-ddl-per-table Option Improvements

CHAPTER

TWENTYEIGHT

GLOSSARY

UUID Universally Unique IDentifier which uniquely identifies the state and the sequence of changes node undergoes. 128-bit UUID is a classic DCE UUID Version 1 (based on current time and MAC address). Although in theory this UUID could be generated based on the real MAC-address, in the Galera it is always (without exception) based on the generated pseudo-random addresses ("locally administered" bit in the node address (in the UUID structure) is always equal to unity).

Complete structure of the 128-bit UUID field and explanation for its generation are as follows:

From	To	Length	Content
0	31	32	Bits 0-31 of Coordinated Universal Time (UTC) as a count of 100-nanosecond intervals since 00:00:00.00, 15 October 1582, encoded as big-endian 32-bit number.
32	47	16	Bits 32-47 of UTC as a count of 100-nanosecond intervals since 00:00:00.00, 15 October 1582, encoded as big-endian 16-bit number.
48	59	12	Bits 48-59 of UTC as a count of 100-nanosecond intervals since 00:00:00.00, 15 October 1582, encoded as big-endian 16-bit number.
60	63	4	UUID version number: always equal to 1 (DCE UUID).
64	69	6	most-significants bits of random number, which generated from the server process PID and Coordinated Universal Time (UTC) as a count of 100-nanosecond intervals since 00:00:00.00, 15 October 1582.
70	71	2	UID variant: always equal to binary 10 (DCE variant).
72	79	8	8 least-significant bits of random number, which generated from the server process PID and Coordinated Universal Time (UTC) as a count of 100-nanosecond intervals since 00:00:00.00, 15 October 1582.
80	80	1	Random bit ("unique node identifier").
81	81	1	Always equal to the one ("locally administered MAC address").
82	127	46	Random bits ("unique node identifier"): readed from the `/dev/urandom` or (if `/dev/urandom` is unavailable) generated based on the server process PID, current time and bits of the default "zero node identifier" (entropy data).

LSN Each InnoDB page (usually 16kb in size) contains a log sequence number, or LSN. The LSN is the system version number for the entire database. Each page's LSN shows how recently it was changed.

innodb_file_per_table By default, InnoDB creates tables and indexes in a file-per-tablespace. If the `innodb_file_per_table` variable is disabled, you can enable the variable in your configuration file:

```
[mysqld]
innodb_file_per_table
```

or start the server with `--innodb_file_per_table`.

innodb_expand_import This feature of *Percona Server for MySQL* implements the ability to import arbitrary *.ibd* files exported using the *Percona XtraBackup* `--export` option.

See the the full documentation for more information.

innodb_data_home_dir The directory (relative to *datadir*) where the database server stores the files in a shared tablespace setup. This option does not affect the location of *innodb_file_per_table*. For example:

```
[mysqld]
innodb_data_home_dir = ./
```

innodb_data_file_path Specifies the names, sizes and location of shared tablespace files:

```
[mysqld]
innodb_data_file_path=ibdata1:50M;ibdata2:50M:autoextend
```

innodb_log_group_home_dir Specifies the location of the *InnoDB* log files:

```
[mysqld]
innodb_log_group_home=/var/lib/mysql
```

innodb_buffer_pool_size The size in bytes of the memory buffer to cache data and indexes of *InnoDB*'s tables. This aims to reduce disk access to provide better performance. By default:

```
[mysqld]
innodb_buffer_pool_size=8MB
```

InnoDB Storage engine which provides ACID-compliant transactions and foreign key support, among others improvements over *MyISAM*. It is the default engine for *MySQL* as of the 8.0 series.

MyISAM Previous default storage engine for *MySQL* for versions prior to 5.5. It doesn't fully support transactions but in some scenarios may be faster than *InnoDB*. Each table is stored on disk in 3 files: *.frm*, *.MYD*, *.MYI*.

XtraDB *Percona XtraDB* is an enhanced version of the InnoDB storage engine, designed to better scale on modern hardware, and including a variety of other features useful in high performance environments. It is fully backwards compatible, and so can be used as a drop-in replacement for standard InnoDB. More information here.

my.cnf This file refers to the database server's main configuration file. Most Linux distributions place it as `/etc/mysql/my.cnf` or `/etc/my.cnf`, but the location and name depends on the particular installation. Note that this is not the only way of configuring the server, some systems does not have one even and rely on the command options to start the server and its defaults values.

datadir The directory in which the database server stores its databases. Most Linux distribution use `/var/lib/mysql` by default.

xbcrypt To support encryption and decryption of the backups, a new tool xbcrypt was introduced to *Percona XtraBackup*. This utility has been modeled after The xbstream binary to perform encryption and decryption outside of *Percona XtraBackup*.

xbstream

To support simultaneous compression and streaming, *Percona XtraBackup* uses the xbstream format. For more information see *--stream*

ibdata Default prefix for tablespace files, e.g. `ibdata1` is a 10MB auto-extensible file that *MySQL* creates for the shared tablespace by default.

.frm For each table, the server will create a file with the `.frm` extension containing the table definition (for all storage engines).

.ibd On a multiple tablespace setup (*innodb_file_per_table* enabled), *MySQL* will store each newly created table on a file with a `.ibd` extension.

.MYD Each *MyISAM* table has `.MYD` (MYData) file which contains the data on it.

.MYI Each *MyISAM* table has `.MYI` (MYIndex) file which contains the table's indexes.

.exp Files with the `.exp` extension are created by *Percona XtraBackup* per each *InnoDB* tablespace when the `--export` option is used on prepare. These files can be used to import those tablespaces on *Percona Server for MySQL* 5.5 or lower versions, see *restoring individual tables*".

.MRG Each table using the **`MERGE`** storage engine, besides of a *.frm* file, will have *.MRG* file containing the names of the *MyISAM* tables associated with it.

.TRG File containing the Triggers associated to a table, e.g. *:file: `mytable.TRG*. With the *.TRN* file, they represent all the Trigger definitions.

.TRN File containing the Triggers' Names associated to a table, e.g. *:file: `mytable.TRN*. With the *.TRG* file, they represent all the Trigger definitions.

.CSM Each table with the **`CSV Storage Engine`** has `.CSM` file which contains the metadata of it.

.CSV Each table with the **`CSV Storage`** engine has `.CSV` file which contains the data of it (which is a standard Comma Separated Value file).

.opt *MySQL* stores options of a database (like charset) in a file with a `.opt` extension in the database directory.

.par Each partitioned table has .par file which contains metadata about the partitions.

CHAPTER

TWENTYNINE

INDEX OF FILES CREATED BY PERCONA XTRABACKUP

- Information related to the backup and the server
  - **backup-my.cnf** This file contains information to start the mini instance of InnoDB during the *--prepare*. This is **NOT** a backup of original `my.cnf`. The InnoDB configuration is read from the file `backup-my.cnf` created by **xtrabackup** when the backup was made. *--prepare* uses InnoDB configuration from `backup-my.cnf` by default, or from *--defaults-file*, if specified. InnoDB configuration in this context means server variables that affect data format, i.e. `innodb_page_size` option, `innodb_log_block_size`, etc. Location-related variables, like `innodb_log_group_home_dir` or `innodb_data_file_path` are always ignored by *--prepare*, so preparing a backup always works with data files from the backup directory, rather than any external ones.
  - **xtrabackup_checkpoints** The type of the backup (e.g. full or incremental), its state (e.g. prepared) and the *LSN* range contained in it. This information is used for incremental backups. Example of the `xtrabackup_checkpoints` after taking a full backup:

    ```
    backup_type = full-backuped
    from_lsn = 0
    to_lsn = 15188961605
    last_lsn = 15188961605
    ```

    Example of the `xtrabackup_checkpoints` after taking an incremental backup:

    ```
    backup_type = incremental
    from_lsn = 15188961605
    to_lsn = 15189350111
    last_lsn = 15189350111
    ```

  - **xtrabackup_binlog_info**

    The binary log file used by the server and its position at the moment of the backup. A result of the following que

    ```
    SELECT server_uuid, local, replication, storage_engines FROM␣
    ↪performance_schema.log_status;
    ```

  - **xtrabackup_binary** The **xtrabackup** binary used in the process.
  - **xtrabackup_logfile** Contains data needed for running the: *--prepare*. The bigger this file is the *--prepare* process will take longer to finish.
  - **<table_name>.delta.meta** This file is going to be created when performing the incremental backup. It contains the per-table delta metadata: page size, size of compressed page (if the value is 0 it means the tablespace isn't compressed) and space id. Example of this file could looks like this:

```
page_size = 16384
zip_size = 0
space_id = 0
```

- Information related to the replication environment (if using the *--slave-info* option):
  - **xtrabackup_slave_info** The `CHANGE MASTER` statement needed for setting up a replica.
- Information related to the *Galera* and *Percona XtraDB Cluster* (if using the *--galera-info* option):
  - **xtrabackup_galera_info** Contains the values of `wsrep_local_state_uuid` and `wsrep_last_committed` status variables

CHAPTER

THIRTY

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 LLC and/or its affiliates. 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.

CHAPTER

THIRTYONE

VERSION CHECKING

Some Percona software contains "version checking" functionality which is a feature that enables Percona software users to be notified of available software updates to improve your environment security and performance. Alongside this, the version check functionality also provides Percona with information relating to which software versions you are running, coupled with the environment confirmation which the software is running within. This helps enable Percona to focus our development effort accordingly based on trends within our customer community.

The purpose of this document is to articulate the information that is collected, as well as to provide guidance on how to disable this functionality if desired.

31.1 Usage

Version Check was implemented in Percona Toolkit 2.1.4, and was enabled by default in version 2.2.1. Currently it is supported as a `--[no]version-check` option by a number of tools in Percona Toolkit, Percona XtraBackup, and PMM (Percona Monitoring and Management).

When launched with Version Check enabled, the tool that supports this feature connects to a Percona's *version check service* via a secure HTTPS channel. It compares the locally installed version for possible updates, and also checks versions of the following software:

- Operating System
- Percona Monitoring and Management (PMM)
- MySQL
- Perl
- MySQL driver for Perl (DBD::mysql)
- Percona Toolkit

Then it checks for and warns about versions with known problems if they are identified as running in the environment.

Each version check request is logged by the server. Stored information consists of the checked system unique ID followed by the software name and version. The ID is generated either at installation or when the *version checking* query is submitted for the first time.

Note: Prior to version 3.0.7 of Percona Toolkit, the system ID was calculated as an MD5 hash of a hostname, and starting from Percona Toolkit 3.0.7 it is generated as an MD5 hash of a random number. Percona XtraBackup continues to use hostname-based MD5 hash.

As a result, the content of the sent query is as follows:

```
85624f3fb5d2af8816178ea1493ed41a;DBD::mysql;4.044
c2b6d625ef3409164cbf8af4985c48d3;MySQL;MySQL Community Server (GPL) 5.7.22-log
85624f3fb5d2af8816178ea1493ed41a;OS;Manjaro Linux
85624f3fb5d2af8816178ea1493ed41a;Percona::Toolkit;3.0.11-dev
85624f3fb5d2af8816178ea1493ed41a;Perl;5.26.2
```

31.2 Disabling Version Check

Although the *version checking* feature does not collect any personal information, you might prefer to disable this feature, either one time or permanently. To disable it one time, use `--no-version-check` option when invoking the tool from a Percona product which supports it. Here is a simple example which shows running pt-diskstats tool from the Percona Toolkit with *version checking* turned off:

```
pt-diskstats --no-version-check
```

Disabling *version checking* permanently can be done by placing `no-version-check` option into the configuration file of a Percona product (see correspondent documentation for exact file name and syntax). For example, in case of Percona Toolkit this can be done in a global configuration file `/etc/percona-toolkit/percona-toolkit.conf`:

```
# Disable Version Check for all tools:
no-version-check
```

In case of Percona XtraBackup this can be done in its configuration file in a similar way:

```
[xtrabackup]
no-version-check
```

31.3 Frequently Asked Questions

31.3.1 Why is this functionality enabled by default?

We believe having this functionality enabled improves security and performance of environments running Percona Software and it is good choice for majority of the users.

31.3.2 Why not rely on Operating System's built in functionality for software updates?

In many environments the Operating Systems repositories may not carry the latest version of software and newer versions of software often installed manually, so not being covered by operating system wide check for updates.

31.3.3 Why do you send more information than just the version of software being run as a part of version check service?

Compatibility problems can be caused by versions of various components in the environment, for example problematic versions of Perl, DBD or MySQL could cause operational problems with Percona Toolkit.

Part XII

Indices and tables

- genindex
- search

INDEX

Symbols

C

D

I

L

M

U

X