Contents

Mark as Complete

Justin Ellingwood Subscribe Share

How To Configure MySQL Backups with Percona XtraBackup on Ubuntu 16.04

Updated August 30, 2017 21.3k BACKUPS MYSQL UBUNTU UBUNTU 16.04

Introduction

Databases often store some of the most valuable information in your infrastructure. Because of this, it is important to have reliable backups to guard against data loss in the event of an accident or hardware failure.

The Percona XtraBackup backup tools provide a method of performing "hot" backups of MySQL data while the system is running. They do this by copying the data files at the filesystem level and then performing a crash recovery to achieve consistency within the dataset.

In this guide, we will create a system to automate backups of MySQL data on an Ubuntu 16.04 server. We will use cron and the Percona tools in a group of scripts to create regular, secure backups that we can use for recovery in case of problems.

Prerequisites

To complete this guide, you will need an Ubuntu 16.04 server with a non-root `sudo` user configured for administrative tasks. You can follow our "Initial Server Setup with Ubuntu 16.04" guide to set up a user with these privileges on your server.

Once you have a `sudo` user available, you will need to install MySQL. Either of these guides can be used, depending on which package you would like to use. The first guide is appropriate if you want to stick with the official Ubuntu repositories, while the second guide is better suited if you require more up-to-date features:

- How To Install MySQL on Ubuntu 16.04 (uses the default package in the Ubuntu repositories)
- How To Install the Latest MySQL on Ubuntu 16.04 (uses updated packages provided by the MySQL project)

Once MySQL is installed, log into your server as your `sudo` user to continue.

Installing the Percona Xtrabackup Tools

The first thing we need to do is install the actual Percona backup utilities. The project maintains its own repositories that we can add to our MySQL server to gain access to the packages.

To start, go to the Percona release page for Ubuntu to find the latest `.deb` packages for installing the repository. Since we are on Ubuntu 16.04, which is codenamed "Xenial Xerus", we should choose the "xenial" package. Right-click the corresponding link and copy the address.

Note: You can double-check the release codename of your server at any time by typing:

```
$ lsb_release -c
```

```
Output
```

```
Codename:   xenial
```

Once you have copied the link, move to the `/tmp` directory and then download the repository configuration package with `curl`:

```
$ cd /tmp
$ curl -LO https://repo.percona.com/apt/percona-release_0.1-4.xenial_all.deb
```

Next, use `dpkg` to install the downloaded package, which will configure the Percona `apt` repository on the system:

```
$ sudo dpkg -i percona*
```

With the new repository configured, we'll update the local package index to pull down information about the newly available packages. We will then install the XtraBackup tools and the `qpress` compression utility from the repository:

```
$ sudo apt-get update
$ sudo apt-get install percona-xtrabackup-24 qpress
```

Among other bundled utilities, the `xtrabackup`, `xbstream`, and `qpress` commands will now be available. Our scripts will use each of these to perform backups and restore data.

Configuring a MySQL Backup User and Adding Test Data

To begin, start up an interactive MySQL session with the MySQL root user:

```
$ mysql -u root -p
```

You will be prompted for the administrative password you selected during the MySQL installation. Once you've entered the password, you will be dropped into a MySQL session.

Create a MySQL User with Appropriate Privileges

The first thing we need to do is create a new MySQL user configured to handle backup tasks. We will only give this user the privileges it needs to copy the data safely while the system is running.

To be explicit about the account's purpose, we will call the new user `backup`. We will be placing the user's credentials in a secure file, so feel free to choose a complex password:

```
mysql> CREATE USER 'backup'@'localhost' IDENTIFIED BY 'password';
```

Next we need to grant the new `backup` user the permissions it needs to perform all backup actions on the database system. Grant the required privileges and apply them to the current session by typing:

```
mysql> GRANT RELOAD, LOCK TABLES, REPLICATION CLIENT, CREATE TABLESPACE, PROCESS, SUPER, CREATE
mysql> FLUSH PRIVILEGES;
```

Our MySQL backup user is configured and has the access it requires.

Create Test Data for the Backups

Next, we will create some test data. Run the following commands to create a `playground` database with an `equipment` table. We will start by inserting a single record representing a blue slide:

```
mysql> CREATE DATABASE playground;
mysql> CREATE TABLE playground.equipment ( id INT NOT NULL AUTO_INCREMENT, type VARCHAR(50), qu
mysql> INSERT INTO playground.equipment (type, quant, color) VALUES ("slide", 2, "blue");
```

Later in this guide, we will use and alter this data to test our ability to create full and incremental backups.

Before we end our MySQL session, we will check the value of the `datadir` variable. We will need to know this value to ensure that our system-level `backup` user has access to the MySQL data files.

Display the value of the `datadir` variable by typing:

```
mysql> SELECT @@datadir;
```

```
Output
+-----------------+
| @@datadir       |
+-----------------+
| /var/lib/mysql/ |
+-----------------+
1 row in set (0.01 sec)
```

Take a note of the location you find.

This is all that we need to do within MySQL at the moment. Exit to the shell by typing:

```
mysql> exit
```

Next, we can take a look at some system-level configuration.

Configuring a Systems Backup User and Assigning Permissions

Now that we have a MySQL user to perform backups, we will ensure that a corresponding Linux user exists with similar limited privileges.

On Ubuntu 16.04, a `backup` user and corresponding `backup` group is already available. Confirm this by checking the `/etc/passwd` and `/etc/group` files with the following command:

```
$ grep backup /etc/passwd /etc/group
```

```
Output
/etc/passwd:backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
/etc/group:backup:x:34:
```

The first line from the `/etc/passwd` file describes the `backup` user, while the second line from the `/etc/group` file defines the `backup` group.

The `/var/lib/mysql` directory where the MySQL data is kept is owned by the `mysql` user and group. We can add the `backup` user to the `mysql` group to safely allow access to the database files and directories. We should also add our `sudo` user to the `backup` group so that we can access the files we will back up.

Type the following commands to add the `backup` user to the `mysql` group and your `sudo` user to the `backup` group:

```
$ sudo usermod -aG mysql backup
$ sudo usermod -aG backup ${USER}
```

If we check the `/etc/group` files again, you will see that your current user is added to the `backup` group and that the `backup` user is added to the `mysql` group:

```
$ grep backup /etc/group
```

```
Output
backup:x:34:sammy
mysql:x:116:backup
```

The new group isn't available in our current session automatically. To re-evaluate the groups available to our `sudo` user, either log out and log back in, or type:

```
$ exec su - ${USER}
```

You will be prompted for your `sudo` user's password to continue. Confirm that your current session now has access to the `backup` group by checking our user's groups again:

```
$ id -nG
```

```
Output
sammy sudo backup
```

Our `sudo` user will now be able to take advantage of its membership in the `backup` group.

Next, we need to make the `/var/lib/mysql` directory and its subdirectories accessible to the `mysql` group by adding group execute permissions. Otherwise, the `backup` user will be unable to enter those directories, even though it is a member of the `mysql` group.

Note: If the value of `datadir` was not `/var/lib/mysql` when you checked inside of MySQL earlier, substitute the directory you discovered in the commands that follow.

To give the `mysql` group access to the MySQL data directories, type:

```
$ sudo find /var/lib/mysql -type d -exec chmod 750 {} \;
```

Our `backup` user now has the access it needs to the MySQL directory.

Creating the Backup Assets

Now that MySQL and system backup users are available, we can begin to set up the configuration files, encryption keys, and other assets that we need to successfully create and secure our backups.

Create a MySQL Configuration File with the Backup Parameters

Begin by creating a minimal MySQL configuration file that the backup script will use. This will contain the MySQL credentials for the MySQL user.

Open a file at `/etc/mysql/backup.cnf` in your text editor:

```
$ sudo nano /etc/mysql/backup.cnf
```

Inside, start a `[client]` section and set the MySQL backup user and password user you defined within MySQL:

/etc/mysql/backup.cnf

```
[client]
user=backup
password=password
```

Save and close the file when you are finished.

Give ownership of the file to the `backup` user and then restrict the permissions so that no other users can access the file:

```
$ sudo chown backup /etc/mysql/backup.cnf
$ sudo chmod 600 /etc/mysql/backup.cnf
```

The backup user will be able to access this file to get the proper credentials but other users will be restricted.

Create a Backup Root Directory

Next, create a directory for the backup content. We will use `/backups/mysql` as the base directory for our backups:

```
$ sudo mkdir -p /backups/mysql
```

Next, assign ownership of the `/backups/mysql` directory to the `backup` user and group ownership to the `mysql` group:

```
$ sudo chown backup:mysql /backups/mysql
```

The `backup` user should now be able to write backup data to this location.

Create an Encryption Key to Secure the Backup Files

Because backups contain all of the data from the database system itself, it is important to secure them properly. The `xtrabackup` utility has the ability to encrypt each file as it is backed up and archived. We just need to provide it with an encryption key.

We can create an encryption key within the backup root directory with the `openssl` command:

```
$ printf '%s' "$(openssl rand -base64 24)" | sudo tee /backups/mysql/encryption_key && echo
```

It is very important to restrict access to this file as well. Again, assign ownership to the `backup` user and deny access to all other users:

```
$ sudo chown backup:backup /backups/mysql/encryption_key
$ sudo chmod 600 /backups/mysql/encryption_key
```

This key will be used during the backup process and any time you need to restore from a backup.

Creating the Backup and Restore Scripts

We now have everything we need to perform secure backups of the running MySQL instance.

In order to make our backup and restore steps repeatable, we will script the entire process. We will create the following scripts:

- `backup-mysql.sh`: This script backs up the MySQL databases, encrypting and compressing the files in the process. It creates full and incremental backups and automatically organizes content by day. By default, the script maintains 3 days worth of backups.

- `extract-mysql.sh`: This script decompresses and decrypts the backup files to create directories with the backed up content.
- `prepare-mysql.sh`: This script "prepares" the back up directories by processing the files and applying logs. Any incremental backups are applied to the full backup. Once the prepare script finishes, the files are ready to be moved back to the data directory.

You can view the scripts in the repository for this tutorial on GitHub at any time. If you do not want to copy and paste the contents below, you can download them directly from GitHub by typing:

```
$ cd /tmp
$ curl -LO https://raw.githubusercontent.com/do-community/ubuntu-1604-mysql-backup/master/backu
$ curl -LO https://raw.githubusercontent.com/do-community/ubuntu-1604-mysql-backup/master/extra
$ curl -LO https://raw.githubusercontent.com/do-community/ubuntu-1604-mysql-backup/master/prepa
```

Be sure to inspect the scripts after downloading to make sure they were retrieved successfully and that you approve of the actions they will perform. If you are satisfied, mark the scripts as executable and then move them into the `/usr/local/bin` directory by typing:

```
$ chmod +x /tmp/{backup,extract,prepare}-mysql.sh
$ sudo mv /tmp/{backup,extract,prepare}-mysql.sh /usr/local/bin
```

Next, we will set up each of these scripts and discuss them in more detail.

Create the backup-mysql.sh Script

If you didn't download the `backup-mysql.sh` script from GitHub, create a new file in the `/usr/local/bin` directory called `backup-mysql.sh`:

```
$ sudo nano /usr/local/bin/backup-mysql.sh
```

Copy and paste the script contents into the file:

/usr/local/bin/backup-mysql.sh

```
#!/bin/bash

export LC_ALL=C

days_of_backups=3  # Must be less than 7
backup_owner="backup"
parent_dir="/backups/mysql"
defaults_file="/etc/mysql/backup.cnf"
todays_dir="${parent_dir}/$(date +%a)"
log_file="${todays_dir}/backup-progress.log"
encryption_key_file="${parent_dir}/encryption_key"
now="$(date +%m-%d-%Y_%H-%M-%S)"
processors="$(nproc --all)"

# Use this to echo to standard error
error () {
    printf "%s: %s\n" "$(basename "${BASH_SOURCE}")" "${1}" >&2
    exit 1
}

trap 'error "An unexpected error occurred."' ERR

sanity_check () {
    # Check user running the script
    if [ "$(id --user --name)" != "$backup_owner" ]; then
        error "Script can only be run as the \"$backup_owner\" user"
    fi
```

Mark as Complete

```
    # Check whether the encryption key file is available
    if [ ! -r "${encryption_key_file}" ]; then
        error "Cannot read encryption key at ${encryption_key_file}"
    fi
}

set_options () {
    # List the xtrabackup arguments
    xtrabackup_args=(
        "--defaults-file=${defaults_file}"
        "--backup"
        "--extra-lsndir=${todays_dir}"
        "--compress"
        "--stream=xbstream"
        "--encrypt=AES256"
        "--encrypt-key-file=${encryption_key_file}"
        "--parallel=${processors}"
        "--compress-threads=${processors}"
        "--encrypt-threads=${processors}"
        "--slave-info"
    )

    backup_type="full"

    # Add option to read LSN (log sequence number) if a full backup has been
    # taken today.
    if grep -q -s "to_lsn" "${todays_dir}/xtrabackup_checkpoints"; then
        backup_type="incremental"
        lsn=$(awk '/to_lsn/ {print $3;}' "${todays_dir}/xtrabackup_checkpoints")
        xtrabackup_args+=( "--incremental-lsn=${lsn}" )
    fi
}

rotate_old () {
    # Remove the oldest backup in rotation
    day_dir_to_remove="${parent_dir}/$(date --date="${days_of_backups} days ago" +%a)"

    if [ -d "${day_dir_to_remove}" ]; then
        rm -rf "${day_dir_to_remove}"
    fi
}

take_backup () {
    # Make sure today's backup directory is available and take the actual backup
    mkdir -p "${todays_dir}"
    find "${todays_dir}" -type f -name "*.incomplete" -delete
    xtrabackup "${xtrabackup_args[@]}" --target-dir="${todays_dir}" > "${todays_dir}/${backup_t

    mv "${todays_dir}/${backup_type}-${now}.xbstream.incomplete" "${todays_dir}/${backup_type}-
}

sanity_check && set_options && rotate_old && take_backup

# Check success and print message
if tail -1 "${log_file}" | grep -q "completed OK"; then
    printf "Backup successful!\n"
    printf "Backup created at %s/%s-%s.xbstream\n" "${todays_dir}" "${backup_type}" "${now}"
else
    error "Backup failure! Check ${log_file} for more information"
fi
```

The script has the following functionality:

- Creates an encrypted, compressed full backup the first time it is run each day.

- Generates encrypted, compressed incremental backups based on the daily full backup when called again on the same day.
- Maintains backups organized by day. By default, three days of backups are kept. This can be changed by adjusting the `days_of_backups` parameter within the script.

When the script is run, a daily directory is created where timestamped files representing individual backups will be written. The first timestamped file will be a full backup, prefixed by `full-`. Subsequent backups for the day will be incremental backups, indicated by an `incremental-` prefix, representing the changes since the last full or incremental backup.

Backups will generate a file called `backup-progress.log` in the daily directory with the output from the most recent backup operation. A file called `xtrabackup_checkpoints` containing the most recent backup metadata will be created there as well. This file is needed to produce future incremental backups, so it is important not to remove it. A file called `xtrabackup_info`, which contains additional metadata, is also produced but the script does not reference this file.

When you are finished, save and close the file.

Next, if you haven't already done so, make the script executable by typing:

```
$ sudo chmod +x /usr/local/bin/backup-mysql.sh
```

We now have a single command available that will initiate MySQL backups.

Create the extract-mysql.sh Script

Next, we will create the `extract-mysql.sh` script. This will be used to extract the MySQL data directory structure from individual backup files.

If you did not download the script from the repository, create and open a file called `extract-mysql.sh` in the `/usr/local/bin` directory:

```
$ sudo nano /usr/local/bin/extract-mysql.sh
```

Inside, paste the following script:

/usr/local/bin/extract-mysql.sh

```
#!/bin/bash

export LC_ALL=C

backup_owner="backup"
encryption_key_file="/backups/mysql/encryption_key"
log_file="extract-progress.log"
number_of_args="${#}"
processors="$(nproc --all)"

# Use this to echo to standard error
error () {
    printf "%s: %s\n" "$(basename "${BASH_SOURCE}")" "${1}" >&2
    exit 1
}

trap 'error "An unexpected error occurred.  Try checking the \"${log_file}\" file for more info

sanity_check () {
    # Check user running the script
    if [ "${USER}" != "${backup_owner}" ]; then
        error "Script can only be run as the \"${backup_owner}\" user"
    fi
```

Mark as Complete

```
    # Check whether the qpress binary is installed
    if ! command -v qpress >/dev/null 2>&1; then
        error "Could not find the \"qpress\" command.  Please install it and try again."
    fi

    # Check whether any arguments were passed
    if [ "${number_of_args}" -lt 1 ]; then
        error "Script requires at least one \".xbstream\" file as an argument."
    fi

    # Check whether the encryption key file is available
    if [ ! -r "${encryption_key_file}" ]; then
        error "Cannot read encryption key at ${encryption_key_file}"
    fi
}

do_extraction () {
    for file in "${@}"; do
        base_filename="$(basename "${file%.xbstream}")"
        restore_dir="./restore/${base_filename}"

        printf "\n\nExtracting file %s\n\n" "${file}"

        # Extract the directory structure from the backup file
        mkdir --verbose -p "${restore_dir}"
        xbstream -x -C "${restore_dir}" < "${file}"

        xtrabackup_args=(
            "--parallel=${processors}"
            "--decrypt=AES256"
            "--encrypt-key-file=${encryption_key_file}"
            "--decompress"
        )

        xtrabackup "${xtrabackup_args[@]}" --target-dir="${restore_dir}"
        find "${restore_dir}" -name "*.xbcrypt" -exec rm {} \;
        find "${restore_dir}" -name "*.qp" -exec rm {} \;

        printf "\n\nFinished work on %s\n\n" "${file}"

    done > "${log_file}" 2>&1
}

sanity_check && do_extraction "$@"

ok_count="$(grep -c 'completed OK' "${log_file}")"

# Check the number of reported completions.  For each file, there is an
# informational "completed OK".  If the processing was successful, an
# additional "completed OK" is printed. Together, this means there should be 2
# notices per backup file if the process was successful.
if (( $ok_count !=  $# )); then
    error "It looks like something went wrong. Please check the \"${log_file}\" file for additi
else
    printf "Extraction complete! Backup directories have been extracted to the \"restore\" dire
fi
```

Unlike the `backup-mysql.sh` script, which is designed to be automated, this script is designed to be used intentionally when you plan to restore from a backup. Because of this, the script expects you to pass in the `.xbstream` files that you wish to extract.

The script creates a `restore` directory within the current directory and then creates individual directories within for each of the backups passed in as arguments. It will process the provided `.xbstream` files by extracting directory structure from the archive, decrypting the individual files within, and then decompressing the decrypted files.

After this process has completed, the `restore` directory should contain directories for each of the provided backups. This allows you to inspect the directories, examine the contents of the backups, and decide which backups you wish to prepare and restore.

Save and close the file when you are finished. Afterward, ensure that the script is executable by typing:

```
$ sudo chmod +x /usr/local/bin/extract-mysql.sh
```

This script will allow us to expand individual backup files into the directory structure needed to restore.

Create the prepare-mysql.sh Script

Finally, download or create the `prepare-mysql.sh` script within the `/usr/local/bin` directory. This script will apply the logs to each backup to create a consistent database snapshot. It will apply any incremental backups to the full backup to incorporate the later changes.

Create the script file in your text editor if you did not download it earlier:

```
$ sudo nano /usr/local/bin/prepare-mysql.sh
```

Inside, paste the following contents:

/usr/local/bin/prepare-mysql.sh

```
#!/bin/bash

export LC_ALL=C

shopt -s nullglob
incremental_dirs=( ./incremental-*/ )
full_dirs=( ./full-*/ )
shopt -u nullglob

backup_owner="backup"
log_file="prepare-progress.log"
full_backup_dir="${full_dirs[0]}"

# Use this to echo to standard error
error() {
    printf "%s: %s\n" "$(basename "${BASH_SOURCE}")" "${1}" >&2
    exit 1
}

trap 'error "An unexpected error occurred.  Try checking the \"${log_file}\" file for more info

sanity_check () {
    # Check user running the script
    if [ "${USER}" != "${backup_owner}" ]; then
        error "Script can only be run as the \"${backup_owner}\" user."
    fi

    # Check whether a single full backup directory are available
    if (( ${#full_dirs[@]} != 1 )); then
        error "Exactly one full backup directory is required."
    fi
}

do_backup () {
    # Apply the logs to each of the backups
    printf "Initial prep of full backup %s\n" "${full_backup_dir}"
    xtrabackup --prepare --apply-log-only --target-dir="${full_backup_dir}"

    for increment in "${incremental_dirs[@]}"; do
        printf "Applying incremental backup %s to %s\n" "${increment}" "${full_backup_dir}"
```

Mark as Complete

```
        xtrabackup --prepare --apply-log-only --incremental-dir="${increment}" --target-dir="${
    done

    printf "Applying final logs to full backup %s\n" "${full_backup_dir}"
    xtrabackup --prepare --target-dir="${full_backup_dir}"
}

sanity_check && do_backup > "${log_file}" 2>&1

# Check the number of reported completions.  Each time a backup is processed,
# an informational "completed OK" and a real version is printed.  At the end of
# the process, a final full apply is performed, generating another 2 messages.
ok_count="$(grep -c 'completed OK' "${log_file}")"

if (( ${ok_count} == ${#full_dirs[@]} + ${#incremental_dirs[@]} + 1 )); then
    cat << EOF
Backup looks to be fully prepared.  Please check the "prepare-progress.log" file
to verify before continuing.

If everything looks correct, you can apply the restored files.

First, stop MySQL and move or remove the contents of the MySQL data directory:

        sudo systemctl stop mysql
        sudo mv /var/lib/mysql/ /tmp/

Then, recreate the data directory and  copy the backup files:

        sudo mkdir /var/lib/mysql
        sudo xtrabackup --copy-back --target-dir=${PWD}/$(basename "${full_backup_dir}")

Afterward the files are copied, adjust the permissions and restart the service:

        sudo chown -R mysql:mysql /var/lib/mysql
        sudo find /var/lib/mysql -type d -exec chmod 750 {} \\;
        sudo systemctl start mysql
EOF
else
    error "It looks like something went wrong.  Check the \"${log_file}\" file for more informa
fi
```

The script looks in the current directory for directories beginning with `full-` or `incremental-`. It uses the MySQL logs to apply the committed transactions to the full backup. Afterwards, it applies any incremental backups to the full backup to update the data with the more recent information, again applying the committed transactions.

Once all of the backups have been combined, the uncommitted transactions are rolled back. At this point, the `full-` backup will represent a consistent set of data that can be moved into MySQL's data directory.

In order to minimize chance of data loss, the script stops short of copying the files into the data directory. This way, the user can manually verify the backup contents and the log file created during this process, and decide what to do with the current contents of the MySQL data directory. The commands needed to restore the files completely are displayed when the command exits.

Save and close the file when you are finished. If you did not do so earlier, mark the file as executable by typing:

```
$ sudo chmod +x /usr/local/bin/prepare-mysql.sh
```

This script is the final script that we run before moving the backup files into MySQL's data directory.

Testing the MySQL Backup and Restore Scripts

Now that the backup and restore scripts are on the server, we should test them.

Perform a Full Backup

Begin by calling the `backup-mysql.sh` script with the `backup` user:

```
$ sudo -u backup backup-mysql.sh
```

```
Output
Backup successful!
Backup created at /backups/mysql/Thu/full-04-20-2017_14-55-17.xbstream
```

If everything went as planned, the script will execute correctly, indicate success, and output the location of the new backup file. As the above output indicates, a daily directory ("Thu" in this case) has been created to house the day's backups. The backup file itself begins with `full-` to express that this is a full backup.

Let's move into the daily backup directory and view the contents:

```
$ cd /backups/mysql/"$(date +%a)"
$ ls
```

```
Output
backup-progress.log  full-04-20-2017_14-55-17.xbstream  xtrabackup_checkpoints  xtrabackup_info
```

Here, we see the actual backup file (`full-04-20-2017_14-55-17.xbstream` in this case), the log of the backup event (`backup-progress.log`), the `xtrabackup_checkpoints` file, which includes metadata about the backed up content, and the `xtrabackup_info` file, which contains additional metadata.

If we tail the `backup-progress.log`, we can confirm that the backup completed successfully.

```
$ tail backup-progress.log
```

```
Output
170420 14:55:19 All tables unlocked
170420 14:55:19 [00] Compressing, encrypting and streaming ib_buffer_pool to <STDOUT>
170420 14:55:19 [00]        ...done
170420 14:55:19 Backup created in directory '/backups/mysql/Thu/'
170420 14:55:19 [00] Compressing, encrypting and streaming backup-my.cnf
170420 14:55:19 [00]        ...done
170420 14:55:19 [00] Compressing, encrypting and streaming xtrabackup_info
170420 14:55:19 [00]        ...done
xtrabackup: Transaction log of lsn (2549956) to (2549965) was copied.
170420 14:55:19 completed OK!
```

If we look at the `xtrabackup_checkpoints` file, we can view information about the backup. While this file provides some information that is useful for administrators, it's mainly used by subsequent backup jobs so that they know what data has already been processed.

This is a copy of a file that's included in each archive. Even though this copy is overwritten with each backup to represent the latest information, each original will still be available inside the backup archive.

```
$ cat xtrabackup_checkpoints
```

```
Output
```

```
backup_type = full-backuped
from_lsn = 0
to_lsn = 2549956
last_lsn = 2549965
compact = 0
recover_binlog_info = 0
```

The example above tells us that a full backup was taken and that the backup covers log sequence number (LSN) 0 to log sequence number 2549956. The `last_lsn` number indicates that some operations occurred during the backup process.

Perform an Incremental Backup

Now that we have a full backup, we can take additional incremental backups. Incremental backups record the changes that have been made since the last backup was performed. The first incremental backup is based on a full backup and subsequent incremental backups are based on the previous incremental backup.

We should add some data to our database before taking another backup so that we can tell which backups have been applied.

Insert another record into the `equipment` table of our `playground` database representing 10 yellow swings. You will be prompted for the MySQL administrative password during this process:

```
$ mysql -u root -p -e 'INSERT INTO playground.equipment (type, quant, color) VALUES ("swing", 1
```

Now that there is more current data than our most recent backup, we can take an incremental backup to capture the changes. The `backup-mysql.sh` script will take an incremental backup if a full backup for the same day exists:

```
$ sudo -u backup backup-mysql.sh
```

```
Output
Backup successful!
Backup created at /backups/mysql/Thu/incremental-04-20-2017_17-15-03.xbstream
```

Check the daily backup directory again to find the incremental backup archive:

```
$ cd /backups/mysql/"$(date +%a)"
$ ls
```

```
Output
backup-progress.log               incremental-04-20-2017_17-15-03.xbstream  xtrabackup_info
full-04-20-2017_14-55-17.xbstream  xtrabackup_checkpoints
```

The contents of the `xtrabackup_checkpoints` file now refer to the most recent incremental backup:

```
$ cat xtrabackup_checkpoints
```

```
Output
backup_type = incremental
from_lsn = 2549956
to_lsn = 2550159
last_lsn = 2550168
compact = 0
recover_binlog_info = 0
```

The backup type is listed as "incremental" and instead of starting from LSN 0 like our full backup, it starts at the LSN where our last backup ended.

Extract the Backups

Next, let's extract the backup files to create backup directories. Due to space and security considerations, this should normally only be done when you are ready to restore the data.

We can extract the backups by passing the `.xbstream` backup files to the `extract-mysql.sh` script. Again, this must be run by the `backup` user:

```
$ sudo -u backup extract-mysql.sh *.xbstream
```

```
Output
Extraction complete! Backup directories have been extracted to the "restore" directory.
```

The above output indicates that the process was completed successfully. If we check the contents of the daily backup directory again, an `extract-progress.log` file and a `restore` directory have been created.

If we tail the extraction log, we can confirm that the latest backup was extracted successfully. The other backup success messages are displayed earlier in the file.

```
$ tail extract-progress.log
```

```
Output
170420 17:23:32 [01] decrypting and decompressing ./performance_schema/socket_instances.frm.qp.
170420 17:23:32 [01] decrypting and decompressing ./performance_schema/events_waits_summary_by_
170420 17:23:32 [01] decrypting and decompressing ./performance_schema/status_by_user.frm.qp.xb
170420 17:23:32 [01] decrypting and decompressing ./performance_schema/replication_group_member
170420 17:23:32 [01] decrypting and decompressing ./xtrabackup_logfile.qp.xbcrypt
170420 17:23:33 completed OK!


Finished work on incremental-04-20-2017_17-15-03.xbstream
```

If we move into the `restore` directory, directories corresponding with the backup files we extracted are now available:

```
$ cd restore
$ ls -F
```

```
Output
full-04-20-2017_14-55-17/  incremental-04-20-2017_17-15-03/
```

The backup directories contains the raw backup files, but they are not yet in a state that MySQL can use though. To fix that, we need to prepare the files.

Prepare the Final Backup

Next, we will prepare the backup files. To do so, you must be in the `restore` directory that contains the `full-` and any `incremental-` backups. The script will apply the changes from any `incremental-` directories onto the `full-` backup directory. Afterwards, it will apply the logs to create a consistent dataset that MySQL can use.

for any reason you don't want to restore some of the changes, now is your last chance to remove those ncremental backup directories from the `restore` directory (the incremental backup files will still be

available in the parent directory). Any remaining `incremental-` directories within the current directory will be applied to the `full-` backup directory.

When you are ready, call the `prepare-mysql.sh` script. Again, make sure you are in the `restore` directory where your individual backup directories are located:

```
$ sudo -u backup prepare-mysql.sh
```

```
Output
Backup looks to be fully prepared.  Please check the "prepare-progress.log" file
to verify before continuing.

If everything looks correct, you can apply the restored files.

First, stop MySQL and move or remove the contents of the MySQL data directory:

        sudo systemctl stop mysql
        sudo mv /var/lib/mysql/ /tmp/

Then, recreate the data directory and  copy the backup files:

        sudo mkdir /var/lib/mysql
        sudo xtrabackup --copy-back --target-dir=/backups/mysql/Thu/restore/full-04-20-2017_14-

Afterward the files are copied, adjust the permissions and restart the service:

        sudo chown -R mysql:mysql /var/lib/mysql
        sudo find /var/lib/mysql -type d -exec chmod 750 {} \;
        sudo systemctl start mysql
```

The output above indicates that the script thinks that the backup is fully prepared and that the `full-` backup now represents a fully consistent dataset. As the output states, you should check the `prepare-progress.log` file to confirm that no errors were reported during the process.

The script stops short of actually copying the files into MySQL's data directory so that you can verify that everything looks correct.

Restore the Backup Data to the MySQL Data Directory

If you are satisfied that everything is in order after reviewing the logs, you can follow the instructions outlined in the `prepare-mysql.sh` output.

First, stop the running MySQL process:

```
$ sudo systemctl stop mysql
```

Since the backup data may conflict with the current contents of the MySQL data directory, we should remove or move the `/var/lib/mysql` directory. If you have space on your filesystem, the best option is to move the current contents to the `/tmp` directory or elsewhere in case something goes wrong:

```
$ sudo mv /var/lib/mysql/ /tmp
```

Recreate an empty `/var/lib/mysql` directory. We will need to fix permissions in a moment, so we do not need to worry about that yet:

```
$ sudo mkdir /var/lib/mysql
```

Now, we can copy the full backup to the MySQL data directory using the `xtrabackup` utility. Substitute the path to your prepared full backup in the command below:

```
$ sudo xtrabackup --copy-back --target-dir=/backups/mysql/Thu/restore/full-04-20-2017_14-55-17
```

A running log of the files being copied will display throughout the process. Once the files are in place, we need to fix the ownership and permissions again so that the MySQL user and group own and can access the restored structure:

```
$ sudo chown -R mysql:mysql /var/lib/mysql
$ sudo find /var/lib/mysql -type d -exec chmod 750 {} \;
```

Our restored files are now in the MySQL data directory.

Start up MySQL again to complete the process:

```
$ sudo systemctl start mysql
```

Check whether the data has been restored by viewing the contents of the `playground.equipment` table. Again, you will be prompted for the MySQL `root` password to continue:

```
$ mysql -u root -p -e 'SELECT * FROM playground.equipment;'
```

```
Output
+----+-------+-------+--------+
| id | type  | quant | color  |
+----+-------+-------+--------+
|  1 | slide |     2 | blue   |
|  2 | swing |    10 | yellow |
+----+-------+-------+--------+
2 rows in set (0.02 sec)
```

Our data has been successfully restored.

After restoring your data, it is important to go back and delete the `restore` directory. Future incremental backups cannot be applied to the full backup once it has been prepared, so we should remove it. Furthermore, the backup directories should not be left unencrypted on disk for security reasons:

```
$ cd ~
$ sudo rm -rf /backups/mysql/"$(date +%a)"/restore
```

The next time we need a clean copies of the backup directories, we can extract them again from the backup files.

Creating a Cron Job to Run Backups Hourly

Now that we've verified that the backup and restore process are working smoothly, we should set up a `cron` job to automatically take regular backups.

We will create a small script within the `/etc/cron.hourly` directory to automatically run our backup script and log the results. The `cron` process will automatically run this every hour:

```
$ sudo nano /etc/cron.hourly/backup-mysql
```

Inside, we will call the backup script with the `systemd-cat` utility so that the output will be available in the journal. We'll mark them with a `backup-mysql` identifier so we can easily filter the logs:

/etc/cron.hourly/backup-mysql

```
#!/bin/bash
sudo -u backup systemd-cat --identifier=backup-mysql /usr/local/bin/backup-mysql.sh
```

Save and close the file when you are finished. Make the script executable by typing:

```
$ sudo chmod +x /etc/cron.hourly/backup-mysql
```

The backup script will now run hourly. The script itself will take care of cleaning up backups older than three days ago.

We can test the `cron` script by running it manually:

```
$ sudo /etc/cron.hourly/backup-mysql
```

After it completes, check the journal for the log messages by typing:

```
$ sudo journalctl -t backup-mysql
```

Output

```
-- Logs begin at Wed 2017-04-19 18:59:23 UTC, end at Thu 2017-04-20 18:54:49 UTC. --
Apr 20 18:35:07 myserver backup-mysql[2302]: Backup successful!
Apr 20 18:35:07 myserver backup-mysql[2302]: Backup created at /backups/mysql/Thu/incremental-0
```

Check back in a few hours to make sure that additional backups are being taken.

Conclusion

In this guide, we've installed the Percona Xtrabackup tools to help create live snapshots of our MySQL data on a regular basis. We configured a MySQL and system backup user, set up an encryption key to secure our backup files, and then set up scripts to automate parts of the backup and restore procedures.

The backup script generates a full backup at the start of each day and incremental backups every hour afterwards, keeping three days of backups at any time. The encrypted files and the encryption key can be used in conjunction with other backup technologies to transfer the data off-site for safekeeping. The extract and prepare scripts let us assemble the backups for the day into a consistent set of data that can be used to restore the system.

Justin Ellingwood

Related Tutorials

How To Set Up Scheduled MongoDB Backups to DigitalOcean Spaces

How To Install MySQL on Ubuntu 18.04

How To Install Linux, Apache, MySQL, PHP (LAMP) stack on Ubuntu 18.04

How To Install and Secure phpMyAdmin with Nginx on Ubuntu 16.04

How To Use Duplicity with GPG to Back Up Data to DigitalOcean Spaces

Contents

Mark as Complete

26 Comments

Leave a comment...

Log In to Comment

jimi *July 14, 2017*

Hi. Something with permissions

```
backup-mysql[34334]: find: failed to restore initial working directory: Permission denied
```

```
ls -lah /home/backup/backups/mysql/
total 12K
drwxrwx--- 4 backup mysql   47 Jul 14 00:01 .
drwxrwx--- 3 backup mysql   18 Jul 13 09:49 ..
-rwxrwx--- 1 backup mysql   32 Jul 13 09:50 encryption_key
drwxrwx--- 2 backup mysql 4.0K Jul 14 12:09 Fri
drwxrwx--- 2 backup mysql 4.0K Jul 13 23:01 Thu
```

edited by kamaln7

anttikoo *August 18, 2017*

Be sure that all files in /var/lib/mysql/ can be accessed by mysql

When making backup, I got InnoDB error 13: mysql has no permission to folder.

I just:

sudo chown -R mysql:mysql /var/lib/mysql

and everything started working.

plindsay *October 27, 2017*

Thanks for the fix! I ran into the same issue.

Mark as Complete

j7an *August 22, 2017*

Great guide. Could you expand it by adding a section on how to use this guide with phpbu for backing up website files with mysql/mariadb and uploading to cloud services?

https://github.com/sebastianfeldmann/phpbu

Thanks.

mszmaida *October 6, 2017*

The presence of high-quality guides like this are one reason that I, as a new developer, am using DigitalOcean.

ajayp *October 26, 2017*

In prepare script apply log is applied to all incremental however Percona have a below note stating . xtrabackup --apply-log-only should be used when merging all incrementals except the last one. That's why the previous line doesn't contain the xtrabackup --apply-log-only option. Even if the xtrabackup --apply-log-only was used on the last step, backup would still be consistent but in that case server would perform the rollback phase

My question is it intentional to do apply log for incremental or overlook ?

Thanks,
Ajay.

jellingwood MOD *October 27, 2017*

@ajayp Yes, good eye, but that was intentional. The `--apply-log-only` option prevents `xtrabackup` from rolling back changes once the incremental delta has been applied to the base backup. The rollback is the final thing we need to do to put the database into a usable state, but it can only be completed when you're done applying incremental backups.

Rather than add the logic to skip applying the `--apply-log-only` flag to the last incremental backup, we just go ahead and treat it like a normal incremental backup and prevent the rollback stage. Then, outside of the loop, we can just manually do the final repair to initiate the rollback, which brings the backup into the same state as if we had not issued `--apply-log-only` on the last incremental backup. So, it's splitting that action into two steps to save having to worry about the iteration number. I hope that helps!

plindsay *October 27, 2017*

Great article, however I ran into a problem when setting up the hourly cron command -

```
systemd-cat: command not found
```

Can anyone advise on how to resolve?

jellingwood MOD *October 27, 2017*

@plindsay Hey there. This article was written with Ubuntu 16.04 in mind, which should include the `systemd-cat` command with the systemd init system. If you're using a different distribution or release, it may rely on a different init system. In that case, you can try changing the cron job to something like this to log to a file instead of `journald`:

/etc/cron.hourly/backup-mysql

```
#!/bin/bash

mkdir -p /var/log/backup-mysql
chown backup:backup /var/log/backup-mysql
sudo -u backup /usr/local/bin/backup-mysql.sh >> /var/log/backup-mysql/backup-mysql.log
```

Mark as Complete

I can't confirm whether this will work with your system, but it will hopefully point you in the right direction.

plindsay *October 27, 2017*

0

Great, thanks @jellingwood - this alternative seems to work well on my droplet (Ubuntu LAMP on 14.04).

danlion *October 29, 2017*

0

One correction to this article (at least that I had to do)...

This:

```
GRANT RELOAD, LOCK TABLES, REPLICATION CLIENT, CREATE TABLESPACE, PROCESS, SUPER, CREATE, I
```

Should also include PROCESS.

So:

```
GRANT PROCESS, RELOAD, LOCK TABLES, REPLICATION CLIENT, CREATE TABLESPACE, PROCESS, SUPER,
```

jellingwood MOD *October 31, 2017*

0

@danlion Hmm, it looks like you have `PROCESS` in there twice now, haha. As long as it works though, go for it!

danlion *October 31, 2017*

0

Hmm (x2), the command at the top of this page does have PROCESS (unless you just changed it).

lngphp *December 8, 2017*

0

Looks like there's an issue with a key:

```
# sudo -u backup backup-mysql.sh
Backup successful!
Backup created at /backups/mysql/Fri/incremental-12-08-2017_19-13-54.xbstream

root@ip-172-31-27-59:/backups/mysql/Fri# cd /backups/mysql/"$(date +%a)"

root@ip-172-31-27-59:/backups/mysql/Fri# ll
total 84128
-rw-r--r-- 1 backup backup    66764 Dec  8 19:14 backup-progress.log
-rw-r--r-- 1 backup backup 85555742 Dec  8 19:12 full-12-08-2017_19-12-10.xbstream
-rw-r--r-- 1 backup backup   510265 Dec  8 19:14 incremental-12-08-2017_19-13-54.xbstream
-rw-r--r-- 1 backup backup      123 Dec  8 19:14 xtrabackup_checkpoints
-rw-r--r-- 1 backup backup      677 Dec  8 19:14 xtrabackup_info

# sudo -u backup extract-mysql.sh *.xbstream
extract-mysql.sh: An unexpected error occurred.  Try checking the "extract-progress.log" fi

# cat extract-progress.log


Extracting file full-12-08-2017_19-12-10.xbstream
```

Mark as Complete

```
mkdir: created directory './restore'
mkdir: created directory './restore/full-12-08-2017_12-26-10'
xtrabackup version 2.4.9 based on MySQL server 5.7.13 Linux (x86_64) (revision id: a467167c
171208 19:28:14 [01] decrypting and decompressing ./ibdata1.qp.xbcrypt
encryption: using gcrypt 1.7.6-beta
encryption: both encryption key and key file specified.
encryption: unable to set libgcrypt cipher key - User defined source 1 : Invalid key length
qpress: Unexpected end of source file -- try the -R flag to recover
Segmentation fault
Error: decrypt and decompress thread 0 failed.


Finished work on full-12-08-2017_19-12-10.xbstream
```

Percona's key command is different: https://www.percona.com/doc/percona-xtrabackup/2.1/innobackupex/encrypted_backups_innobackupex.html

```
openssl enc -aes-256-cbc -pass pass:Password -P -md sha1
```

Even using Percona command to generate key doesn't help - it's failing when extracting it, but if I change `--encrypt-key-file` option to `--encrypt-key` and hardcode a key in both files, it works.

What could it be? Newer version of Percona Backup tool?

notinkham *December 17, 2017*

0

Thanks for this. After whole day this is the only thing that worked since using googling for solution didn't help here - almost no one reported this issue and all suggested solutions didn't work.

masbrow *December 26, 2017*

0

i have same issue, may you explain the sollution?

jellingwood MOD *December 26, 2017*

1

@lngphp @notinkham @masbrow Sorry for the slow response.

It appears that this is related to a bug with the latest version of Xtrabackup. You can find more details here. The developers have already created a fix and committed it to the repository, so whenever they cut a release for 2.3.11 (or 2.4.10 for the 2.4 series), the fix should be included.

If you need to use this in the mean time, you can try downloading directly from the Percona GitHub repository, but you will have to compile the software yourself and this will be more difficult to maintain long term than a package.

Sorry for the inconvenience. Unfortunately, the most recent package introduced this bug.

stevevariss *January 9, 2018*

0

Hey.

When I do an extraction, the extract-progress.log has a few decrypt failures? See below:

180108 23:33:10 [01] decrypting and decompressing
./sys/x@0024wait_classes_global_by_avg_latency.frm.qp.xbcrypt
encryption: using gcrypt 1.6.5
decrypt: failed to close dest file.
encryption: using gcrypt 1.6.5
decrypt: failed to close dest file.
180108 23:33:10 completed OK!

Anything to be concerned about?

Mark as Complete

stevеariss *January 9, 2018*

0

I should mention that the db restore is still successful even with those errors

madhu.avasarala *January 25, 2018*

0

Hi,
Thanks for a great article. The backups were working fine till about a day ago when I am getting the following error, every hour, when the incremental backups are supposed to take place.
Any sugestions on how to debug this issue based on the error log below will be highly appreciated.
Thanks

version*check Connecting to MySQL server with DSN 'dbi:mysql:;mysql*read*default*group=xtrabackup' as 'backup' (using password: YES).
180125 04:17:01 version*check Connected to MySQL server*
*180125 04:17:01 version*check Executing a version check against the server...
180125 04:17:01 version*check Done.*
180125 04:17:01 Connecting to MySQL server host: localhost, user: backup, password: set, port: not set, socket: not set
Using server version 5.7.21-0ubuntu0.16.04.1
*xtrabackup version 2.4.9 based on MySQL server 5.7.13 Linux (x86*64) (revision id: a467167cdd4)
xtrabackup: uses posix*fadvise().*
xtrabackup: Can't change dir to '/var/lib/mysql/' (Errcode: 13 - Permission denied)
*xtrabackup: cannot my*setwd /var/lib/mysql/

stevеariss *January 28, 2018*

0

Sweet! I'm not the only one that has this problem. Did you find a fix?

madhu.avasarala *January 28, 2018*

0

Fixed my problem. As indicated in the error, it was a case of permissions. In the article there is a command to set permissions of /var/lib/mysql as

sudo find /var/lib/mysql -type d -exec chmod 750 {} \;

I don't know how, but probably as a result of updates the directory permissions must have been reset. When the permissions are restored with above command all is well.

ibrasec *February 9, 2018*

0

Thanks for your explanation... this was really helpful .
But Can we use the same 3 shell script (backup-mysql.sh , extract-mysql.sh , prepare-mysql.sh) on ubunut 14.04 to achieve auto backup or not ?

Kooky *February 17, 2018*

0

Hi,

Thank you so much for this topic.

I followed the steps, everything worked perfect for me.

I have a question.

I need to restore the backup to another server. I moved all files (encryption*key, full, incremental xbstream files, xtrabackup*checkpoints etc.) to the new server.

When I exectuded the extract-mysql.sh script I have an error:
xtrabackup: Error: cannot open ./xtrabackup*checkpoints*
*xtrabackup: Error: failed to read metadata from './xtrabackup*checkpoints'
encryption: both encryption key and key file specified.
encryption: unable to set libgcrypt cipher key - User defined source 1 : Invalid key length

When I checked, I understoot that it's a problem with the encryption_key file that I exported.

Mark as Complete

For example, in the script when I replace:
"--encrypt-key-file=${encryption*key*file}"
by: "--encrypt-key-=78JKDSKJ78DSKJB37823BUIKDSKJ32HJ"
it works

I would like to know how can I export the key file

I saw somewhere that text file in some cases can contain the CRLF.

Even when I used:

echo -n "A1EDC73815467C083B0869508406637E" > /data/backups/encryption_key
It's not work.

Some help please!!

afif *April 12, 2018*

0

can i using this for remote host ?

pata2004 *April 23, 2018*

0

In Centos 7 this only work with root user because with backup user I got “InnoDB error 13: mysql has no permission to folder”.
I made:
1- sudo chown -R mysql:mysql /var/lib/mysql
2- sudo find /var/lib/mysql -type d -exec chmod 750 {} \;
But nothing solved my problem with backup user.
Anyone know how put this working with a different user from root?