Be Backup ready for all disasters!

Tutorial

Roman Vynar
Akshay Suryawanshi

April 13, 2015

PERCONA LIVE

Introduction

This is a hands-on tutorial on setting up the different types of MySQL backups for your production systems and recovering from various disasters.

Prerequisites:

- good knowledge of Linux, Shell and MySQL
- be familiar with mysqldump and MySQL binlogs
- a laptop with Virtualbox installed

Topics and tasks to be covered

- Planning, strategies and tools for MySQL backups
- Ensuring the replication consistency
- Role of the delayed slaves
- Binary backups with Percona Xtrabackup and mylvmbackup
- Logical backups with mysqldump and mydumper/myloader
- Binlog backups with mysqlbinlog
- Encryption of backups
- Off-site backups
- Monitoring of backups and its importance
- Saving disk space tips
- Various recovery scenarios including disaster recovery

Virtualbox preparation

There are two pre-installed instances:

- master.vm - MySQL master
- slave.vm - MySQL slave

- ☑ Please copy the images from USB sticks to your laptop
- ☑ Start the Virtualbox application
- ☑ Add both instances using “Machine > Add” menu but do not start them yet

Virtualbox network

Each instance has 2 network adapters:

- Host-only adapter
- NAT

- ☑ Configure host-only network from the main menu: Virtualbox > Preferences > Network > Host-only Networks > “vboxnet0” or “Virtualbox Host-Only Ethernet Adapter” > edit and set: `192.168.56.1 / 255.255.255.0`

  Windows users only: open Setting > Network and click OK to re-save host-only network adapter.

Starting instances

Internal IP addresses statically assigned:

- master.vm - 192.168.56.201
- slave.vm - 192.168.56.202

Both instances are running CentOS 6.6 and have the necessary packages pre-installed.

Unix and MySQL root password: abc123

- ☑ Launch both instances
- ☑ Verify network connectivity

Pre-installed packages

- EPEL repo:
  rpm -Uvh http://mirror.omnilance.com/epel/6/i386/epel-release-6-8.noarch.rpm
- Percona repo:
  rpm -Uvh http://www.percona.com/downloads/percona-release/redhat/0.1-3/percona-release-0.1-3.noarch.rpm
- Packages:
  yum install Percona-Server-server-56 Percona-Server-client-56 Percona-Server-shared-56 percona-xtrabackup percona-toolkit mylvmbackup nsca-client qpress
  rpm -ivh files/mydumper-0.6.3-153.el6.x86_64.rpm

Pre-configuration

We have already installed Percona Server 5.6, setup the replication master > slave and placed the basic /etc/my.cnf and credentials file /root/.my.cnf.

Also we have loaded a sample database “employees” from https://launchpad.net/test-db

Under /root/files/ we have put some files to simplify tutorial tasks.

- ☑ Verify replication status of slave.vm.

Factors to consider for backups

- Application bugs
- Software misconfigurations
- Hardware failures
- People mistakes
- Access breach
- Disasters

Some examples of problems

- A DBA runs DELETE query on the wrong data set
- Your application has a bug that overwrites data
- A table or entire schema was dropped by accident
- InnoDB corruption bug that propagates via replication
- Your database or application was hacked
- A disk failed and RAID array does not recover
- A hurricane puts a few feet of water in your datacenter

Planning and strategies

- Perform both logical and binary backups
- Store backups on more than one server
- Store copies of your backups off-site
- Estimate the backup retention
- Do capacity planning
- Monitor backups
- Verify replication consistency periodically
- Test your backups

Tools for MySQL backups

- mysqldump
- mysqlbinlog
- mydumper / myloader
- Percona Xtrabackup
- mylvmbackup

Other tools:

- s3cmd
- qpress
- gpg

Replica is not a backup

- A replication slave is not a backup!
- A slave is the right place to take backups from.
- A delayed slave can be used for a really fast partial data recovery.

Ensuring replication consistency

- pt-table-checksum
  http://www.percona.com/doc/percona-toolkit/2.1/pt-table-checksum.html

- ☑ Checksum master from any node:
  pt-table-checksum h=192.168.56.201
- ☑ Verify no table diffs:
  pt-table-checksum h=192.168.56.201 --replicate-check-only

Create and fix a replication diff

Let's create some discrepancy in replication.

- Run UPDATE query on the slave:
  mysql> UPDATE employees.dept_manager SET from_date=CURDATE() WHERE emp_no=110022;
- Checksum master again, just this one table:
  pt-table-checksum h=192.168.56.201 --tables employees.dept_manager
- Fix it by running pt-table-sync on the master:
  pt-table-sync --execute --replicate percona.checksums --sync-to-master 192.168.56.202
- Re-checksum the table in question:
  pt-table-checksum h=192.168.56.201 --tables employees.dept_manager

Role of delayed slaves

- pt-slave-delay
  http://www.percona.com/doc/percona-toolkit/2.1/pt-slave-delay.html

- ☑ Run:
  pt-slave-delay --delay 1h 192.168.56.202
- ☑ Check out SHOW SLAVE STATUS

Binary backups: Percona Xtrabackup

Percona Xtrabackup is an open source, free MySQL hot backup software that performs non-blocking backups for InnoDB.

- 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

When do we restore from Xtrabackup?

- Setting up new slaves
- When we lose an entire server due to hardware failure, corruption, etc.
- When the majority of data on the server was lost
- Basically when restoring may take less time that trying to load a logical backup or perform InnoDB recovery

Creating a backup with XtraBackup

From now on, all the tasks we will be performing on the slave.

- mkdir -p /backups/xtrabackup
- Optionally, increase open_file_limit:
  ulimit -n 1000000
- Run Xtrabackup:
  innobackupex --compress --rsync --slave-info /backups/xtrabackup
- ls -la /backups/xtrabackup/

Incremental backup with Xtrabackup

Percona Xtrabackup supports incremental backups, which means that it can copy only the data that has changed since the last full backup.

- Do some change on **Master** for testing purpose:
  mysql> UPDATE employees.dept_manager SET from_date=CURDATE();
- mkdir /backups/xtrabackup_inc
- innobackupex --slave-info --incremental --incremental-basedir=<path to full> /backups/xtrabackup_inc
- Compare size of full vs incremental backup:
  du -sh /backups/xtrabackup*/*

Binary backups: mylvmbackup

mylvmbackup is a tool for quickly creating full physical backups of a MySQL server's data files

http://www.lenzg.net/mylvmbackup/

It performs the actual backup by accessing the file system directly. It is also a requirement that the MySQL server's data directory resides on an LVM volume and the same volume group has as much free space for a snapshot.

Using mylvmbackup

When do we restore from mylvmbackup:

- When the MySQL datadir is huge
- Need to restore all at once

Let's create a backup:

- ☑ mkdir /backups/mylvmbackup
- ☑ mylvmbackup --vgname=Mysql_VG --lvname=data --lvsize=1G --backupdir=/backups/mylvmbackup

It will create a compressed backup while your MySQL server is running.

Logical backups: mysqldump

Simplest way of doing logical backups.

- Dumps tables in a single thread
- Available with a standard MySQL distribution
- Can perform online backup for InnoDB tables

- ☑ mkdir /backups/mysqldump
- ☑ mysqldump --single-transaction --all-databases | gzip > /backups/mysqldump/dump_20141103.sql.gz

Logical backups: mydumper/myloader

mydumper is a tool used for backing up MySQL database servers much faster than the mysqldump.

- Parallelism (hence, speed) and performance (avoids expensive character set conversion routines, efficient code overall)
- Easier to manage output (separate files for tables, dump metadata, etc, easy to view/parse data)
- Consistency - maintains snapshot across all threads, provides accurate master and slave log positions, etc
- Manageability - supports PCRE for specifying database and tables inclusions and exclusions

When do we restore from mydumper?

- Restoring a single table
- Restoring a single schema or rolling forward a single schema to a point in time (involves using binlogs)
- Restoring data in multiple threads

Caveats:

- dumps schema and data into a per-table file
- does not dump triggers and views

Schema backup

- Loop through each DB:
  - Write out ALTER DATABASE DEFAULT CHARACTER SET CHARSET > schema.sql
  - Create schema dump:
    mysqldump -d --single-transaction -R --skip-triggers —ignore-create-error DB >> schema.sql
  - Create post-schema dump:
    mysqldump -d --single-transaction -t --ignore-create-error DB > schema-post.sql

Advanced schema backup

- Loop through each DB:
  - Write out ALTER DATABASE DEFAULT CHARACTER SET CHARSET > schema.sql
  - mysqldump -d --single-transaction -R --skip-triggers —ignore-create-error DB --innodb-optimize-keys > tmp.sql
    cat tmp.sql | grep -v ALTER >> schema.sql
    cat tmp.sql | grep ALTER > schema-post.sql
  - mysqldump -d --single-transaction -t --ignore-create-error DB >> schema-post.sql

Works only with mysqldump from Percona Server package.

See http://www.percona.com/doc/percona-server/5.6/management/innodb_expanded_fast_index_creation.html

Create mydumper data backup

Dump all data with the command like:

- mydumper --outputdir=<backup dir> --host=localhost --kill-long-queries --no-schemas --verbose=3

- ☑ On slave.vm check out and run the script: /root/files/mydumper.sh

mydumper backup structure

- /backups/mydumper/<path>/mysql-schema.sql
- /backups/mydumper/<path>/mysql-schema-post.sql
- /backups/mydumper/<path>/mysql.db.sql.gz
- /backups/mydumper/<path>/mysql.proc.sql.gz
- /backups/mydumper/<path>/mysql.time_zone.sql.gz
- /backups/mydumper/<path>/mysql.user.sql.gz
- …
- similar for all other schemas
- /backups/mydumper/<path>/metadata

Binlog backups: mysqlbinlog 5.6

- mysqlbinlog --read-from-remote-server --raw --stop-never
- Mirror the binlogs on the master to a second server
- Works with 5.1/5.5 servers also
- Roll forward backups even after losing the master
- Useful for point-in-time recovery or disaster recovery
- Easy to backup off-site
- Takes very little resources to run. Can run about anywhere with disk space and writes out binlog files sequentially.

Pulling binlogs with mysqlbinlog

- Ensure it stays running, restart it if it appears to be hanging
- Optionally, verify the file size is the same on master and slave
- Re-transfer files that are partially transferred
- Compress the files after successful transfer

- ☑ mkdir -p /backups/binlogs
- ☑ cd /backups/binlogs
- ☑ mysqlbinlog --raw --read-from-remote-server --stop-never --verify-binlog-checksum --host=192.168.56.201 --stop-never-slave-server-id=999 mysql-bin.000001
- ☑ Compare what was downloaded and master binlogs

Encryption of backups

- To avoid information disclosure, NEVER upload your backups off-site unencrypted
- GPG tools can help you to achieve a strong level of encryption:
  - Encrypt file:
    gpg --batch --yes --encrypt --always-trust --quiet --recipient=my@backups.local --output=file.sql.gpg file.sql
  - Decrypt file:
    gpg --batch --decrypt --passphrase-fd=0 --quiet --output=file.sql file.sql.gpg

Encryption of backups

- Generate GPG key backup@slave.vm for our backups:
  gpg --gen-key
  To gain enough entropy you can run Xtrabackup a few times:
  innobackupex --compress --rsync --slave-info /backups/ xtrabackup
- Encrypt mydumper backup folder:
  /root/files/encrypt.sh
- Decrypt folder:
  /root/files/decrypt.sh
- Backup GPG keys as well:
  gpg --export --armor backup@slave.vm > public_key.asc
  gpg --export-secret-keys --armor backup@slave.vm > private_key.asc
- Keep passphrase and keys backup in redundant location!

Off-site backups to Amazon S3

Amazon S3 is a reasonably priced data storage service. Ideal for off-site file backups, file archiving, web hosting and other data storage needs.

With S3 you can set bucket lifecycle properties so data over X days is archived to Glacier and data over Y days is removed.

s3cmd

S3cmd is a free command line tool and client for uploading, retrieving and managing data in Amazon S3 and other cloud storage service providers that use the S3 protocol, such as Google Cloud Storage or DreamHost DreamObjects.

It is ideal for batch scripts and automated backup upload to S3, triggered from cron, etc.
http://s3tools.org/s3cmd

Using s3cmd

- s3cmd --configure
  - enable https
- Check bucket permissions:
  s3cmd info s3://bucket/

  Beware, check there is no ACL for http://acs.amazonaws.com/groups/global/AuthenticatedUsers, it makes your data public!
- Set lifecycle rule for the bucket as appropriate
- Upload a backup:
  s3cmd --server-side-encryption sync <backup dir>/ s3://bucket/path/

Monitoring of backups and its importance

- Always monitor backup status
- Employ passive checks
- Consider checking freshness of backups
- Some kind of heartbeat for mysqlbinlog needed
- Most of issues with backups are due to problems with FLUSH TABLE WITH READ LOCK or long running queries. So MySQL monitoring itself is recommended (you can use Percona Monitoring Plugins for this).

Monitoring backups with Nagios

- With Nagios, you can employ NSCA alerts:
  cat /root/files/monitor.sh

Nagios option “freshness_threshold” could be set for a service check so if your NSCA report has not checked in within a certain period it will alert you.

```
define service{
        use                             passive-service
        host_name                       slave.vm
        service_description             Backup xtrabackup
        freshness_threshold             129600
        check_command                   check_dummy!36 hours
}
```

Saving disk space

- Enable compression
- Perform incremental backups with Xtrabackup
- Hardlink files:
  - If md5sum(file1) == md5sum(file2):
    - cp -al file1 file2

- ☑ Create one more mydumper backup to have two: /root/files/mydumper.sh
- ☑ Check out and run the script: /root/files/hardlink.sh
- ☑ du -sh /backups/mydumper/*

Recovery scenarios

Partial data recovery:

- Recovering a single table or dataset
- Involves using a logical backup
- Sometimes applying selective events further from the binlogs too

Full data recovery:

- Can be done from any type of a full backup

Point-in-time recovery (disaster scenario):

- Involves full data recovery and replaying the binlogs to the desired point in time

Partial data recovery: a single table reload

- Update table employees.dept_emp table using some garbage values or imagine a statement was accidentally executed without a where clause.
  On the master, run:
  mysql> UPDATE employees.dept_emp SET to_date=CURDATE();

- Restore the whole table from the slave:
  mysql -h 192.168.56.201 employees
  mysql> TRUNCATE TABLE employees.dept_emp;
  zcat /backups/mydumper/<path>/employees.dept_emp.sql |
  mysql -h 192.168.56.201 employees

- Verify the data.

Partial data recovery from mydumper

Let's now reload one schema with the different name on the slave.

- cd /backups/mydumper/<path>
- mysql> CREATE DATABASE employees_restore;
- cat employees-schema.sql | mysql employees_restore
- for FILE in `ls employees.*`; do
  zcat $FILE | mysql employees_restore ;
  done
- cat employees-schema-post.sql | mysql employees_restore

Partial data recovery from mydumper

Now the same task but using multi-threaded tool myloader.

- mysql> CREATE DATABASE employees_restore2;
- cat employees-schema.sql | mysql employees_restore2
- mkdir tmp
- cp -al employees.* tmp/
- cp -al metadata tmp/
- myloader --host=localhost --directory=tmp --threads=8 --queries-per-transaction=10 --verbose=3 --database=employees_restore2
- cat employees-schema-post.sql | mysql employees_restore2

If you want myloader to replicate data, add --enable-binlog option.

Full data recovery from Xtrabackup

- service mysql stop
- rm -rf /data/mysql
- cp -r /backups/xtrabackup/<path> /data/mysql
- innobackupex --decompress /data/mysql/
- innobackupex --apply-log --use-memory=256M /data/mysql/
- chown -R mysql:mysql /data/mysql
- service mysql start
- cat /data/mysql/xtrabackup_slave_info
- mysql> CHANGE MASTER TO MASTER_LOG_FILE='<BINLOG>', MASTER_LOG_POS=<POS>, MASTER_HOST='192.168.56.201', MASTER_USER='repl', MASTER_PASSWORD='abc123';
- mysql> START SLAVE;

Reloading from incremental Xtrabackup

- service mysql stop
- rm -rf /data/mysql
- cp -r /backups/xtrabackup/<path to full> /data/mysql
- innobackupex --decompress /data/mysql/
- innobackupex --apply-log --redo-only --use-memory=256M /data/mysql/
- innobackupex --apply-log --redo-only --use-memory=256M /data/mysql/ --incremental-dir=/backups/xtrabackup_inc/<path to incremental>

If you want to apply more incremental backups, repeat the last step for them. It is important that you do this in the chronological order in which the backups were done.

You can check the file xtrabackup_checkpoints at the directory of each one for order.

Reloading from incremental Xtrabackup

- innobackupex --apply-log --use-memory=256M /data/mysql/
- chown -R mysql:mysql /data/mysql
- service mysql start
- cat /data/mysql/xtrabackup_slave_info
- mysql> CHANGE MASTER TO MASTER_LOG_FILE='<binlog>', MASTER_LOG_POS=<POS>, MASTER_HOST='192.168.56.201', MASTER_USER='repl', MASTER_PASSWORD='abc123';
- mysql> START SLAVE;
- mysql> SHOW SLAVE STATUS\G

Full data recovery from mylvmbackup

- service mysql stop
- rm -rf /data/mysql
- cd /backups/mylvmbackup
- tar zxf <backup file>
- mv backup/mysql /data/mysql
- service mysql start
- mysql> SHOW SLAVE STATUS\G

Point-in-time recovery

Assume we have only one server master.vm and all the next actions we will be doing there.

- Create Xtrabackup backup:
  innobackupex --compress --rsync /root
- Do some notable changes in mysql and flush logs for testing, e.g.:
  mysql> DROP TABLE employees.salaries;
  mysql> FLUSH LOGS;
- Pull binlogs locally using mysqlbinlog simulating remote downloading:
  mkdir /root/binlogs
  cd /root/binlogs
  mysqlbinlog --raw --read-from-remote-server --stop-never --verify-binlog-checksum --host=localhost --stop-never-slave-server-id=999 mysql-bin.000001
- Check and compare if the are all downloaded:
  ls -la /root/binlogs
  ls -la /var/lib/mysql

Point-in-time recovery

- Stop mysqlbinlog
- Vanish MySQL:
  service mysql stop
  rm -rf /var/lib/mysql
- Restore from Xtrabackup:
  mv /root/<path> /var/lib/mysql
  innobackupex --decompress /var/lib/mysql
  innobackupex --apply-log --use-memory=256M /var/lib/mysql
  chown -R mysql:mysql /var/lib/mysql
  service mysql start
- Verify that MySQL data is not the recent, i.e. at the point of taking Xtrabackup.

Point-in-time recovery

- Roll forward the binlogs from backup:
  cat /var/lib/mysql/xtrabackup_binlog_info
  mysqlbinlog --start-position=<POS> /root/binlogs/<starting binlog> | mysql

Load all the binlogs after this one till the last we have.

- Check out that all the recent MySQL changes were applied after time Xtrabackup was done. We have completed a disaster recovery.

The same way we can restore to any point in time, this is limited to how old full backup we have, binary or logical, and presence of all the related binlog files.

“Percona Backup Service”

- Let the Percona Managed Services team worry about your MySQL backups and data recovery! http://www.percona.com/products/percona-managed-services/percona-backup-service

- Reliable MySQL Backups
- Efficient MySQL Data Recovery
- Cost Effective and Highly Flexible

Questions?

Thank you for attending!

Percona Live 2015

PERCONA LIVE

www.percona.com/live