Ideal Techniques for making Major MySQL Upgrades easier!

Arunjith Aravindan

Senior MySQL DBA, Percona

Arunjith Aravindan

- ❖ **Has been consulting on Open Source and MySQL for 13+ years.**
- ❖ **Joined Percona in July 2014, and consults with Percona's Managed Services customers on building and maintaining reliable and high-performance MySQL infrastructures.**
- ❖ **https://www.percona.com/blog/author/arunjith-aravindan/**

Agenda

- Why must MySQL be upgraded?
- How Percona Utilities help Version Upgrades ?
- Major Version Upgrade Process!
- How to Minimize the Upgrade Downtime?
- How To Upgrade Asynchronous Replication Environment?
- How to upgrade Percona Xtradb cluster environment?

Why must MySQL be upgraded?

It is essential to Upgrade MySQL to the Most Recent Version.

- MySQL 5.7 – End of life planned until October, 2023.
- New Features.
- Performance Advantages.
- Bug corrections.
- Obsolete versions are vulnerable.

Major Version vs. Minor Version

In MySQL, a **major** version upgrade would be 5.6 -> 5.7 -> 8.x.

Minor versions, such as 5.6.x->5.6.y, 5.7.x->5.7.y, and 8.x->8.y, are more common.

There is a significant difference between updating to a major version and upgrading to a minor version.

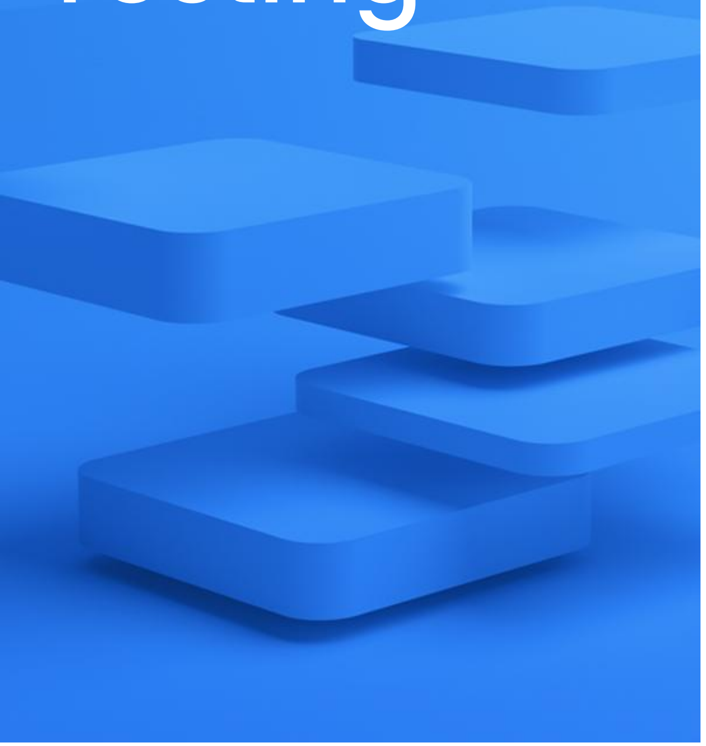
> mysql_upgrade [MySQL utility for upgrading databases to new MySQL versions]
>
> Upgrade both the system tables and the data.
>
> mysql_upgrade -s
>
> [-s, --upgrade-system-tables Only upgrade the system tables, do not try to upgrade the data.]
>
> As of MySQL 8.0.16, the MySQL server performs the upgrade tasks previously handled by mysql_upgrade. mysql_upgrade is not needed and is deprecated as of that version.

But why test?

Application & Server side Testing

- It costs time and money! Infrastructure costs, schedule delays, application and/or database changes
- Replication is NOT supported from new major versions to previous major versions
- High risk of downtime and few rollback options, all are painful.
  - Lose Writes
  - Risk Split Brain
  - Fail forward and apply needed changes to Application or DB

Application
&
Server side
Testing



Do pt-upgrade and Upgrade Checker Utility do the same tests?

- The pt-upgrade tool aids in testing application queries and generates reports on how each queries performs on servers running various versions of MySQL.
- MySQL Shell Upgrade Checker is a utility that helps in compatibility tests between MySQL 5.7 instances and MySQL 8.0 upgrades, which is part of the mysql-shell-utilities.

Server side Testing

MySQL Shell Upgrade Checker

Server side Testing

MySQL Shell Upgrade Checker

Built-in function util.checkForServerUpgrade()

Checks for compatibility such as the examples mentioned below:

- Usage of old temporal type
- Usage of db objects with names conflicting with new reserved keywords
- Removed system variables
- Around 21 checks are performed by the tool, which results in a detailed report of Errors, Warnings, and Notices.

MySQL Shell Upgrade Checker

How to generate a report using Upgrade Checker Utility

```
mysqlsh -- util checkForServerUpgrade 'root@localhost:3306'
--target-version=8.0.31 --config-path=/etc/my.cnf >
CheckForServerUpgrade_Report.txt
Please provide the password for 'mysqluser@localhost:3306':


$ mysqlsh
MySQL  JS > util.checkForServerUpgrade('root@localhost:3306', {
"targetVersion":"8.0.29", "configPath":"/etc/my.cnf"})
Please provide the password for 'mysqluser@localhost:3306':


MySQL  JS > \exit
Bye!
```

MySQL Shell Upgrade Checker

Compatibility Checks Example

```
1) Usage of old temporal type
  No issues found

2) Usage of db objects with names conflicting with reserved keywords in 8.0
  No issues found

3) Usage of utf8mb3 charset
  No issues found

4) Table names in the mysql schema conflicting with new tables in 8.0
  No issues found

5) Partitioned tables using engines with non native partitioning
  No issues found

6) Foreign key constraint names longer than 64 characters
  No issues found

7) Usage of obsolete MAXDB sql_mode flag
  No issues found

8) Usage of obsolete sql_mode flags
  No issues found

9) ENUM/SET column definitions containing elements longer than 255 characters
  No issues found

10) Usage of partitioned tables in shared tablespaces
  No issues found
```

Usage of old temporal type

MySQL changed the temporal types in MySQL 5.6.4, and it introduced a new feature: microseconds resolution in the TIME, TIMESTAMP and DATETIME types.

Tables should be altered to use the new format before upgrading to MySQL 5.7 or running the alter part of the mysql_upgrade takes time while you run it.

```
error : Table rebuild required. Please do "ALTER TABLE `t1` FORCE" or dump/reload to fix it!
Running : ALTER TABLE `test`.`t1` FORCE
```

If it isn't done, though, the upgrade can still be done without changing all of the tables and upgrading just system tables; nevertheless, you won't be able to use microseconds, and it will take up more disc space. But it is always better to upgrade all tables.

However, if you have to upgrade to 5.7, you can change the format later also with alter table or using pt-online-schema-change.

Usage of db objects with names conflicting with new reserved keywords.

DB objects & reserved keywords

Example:

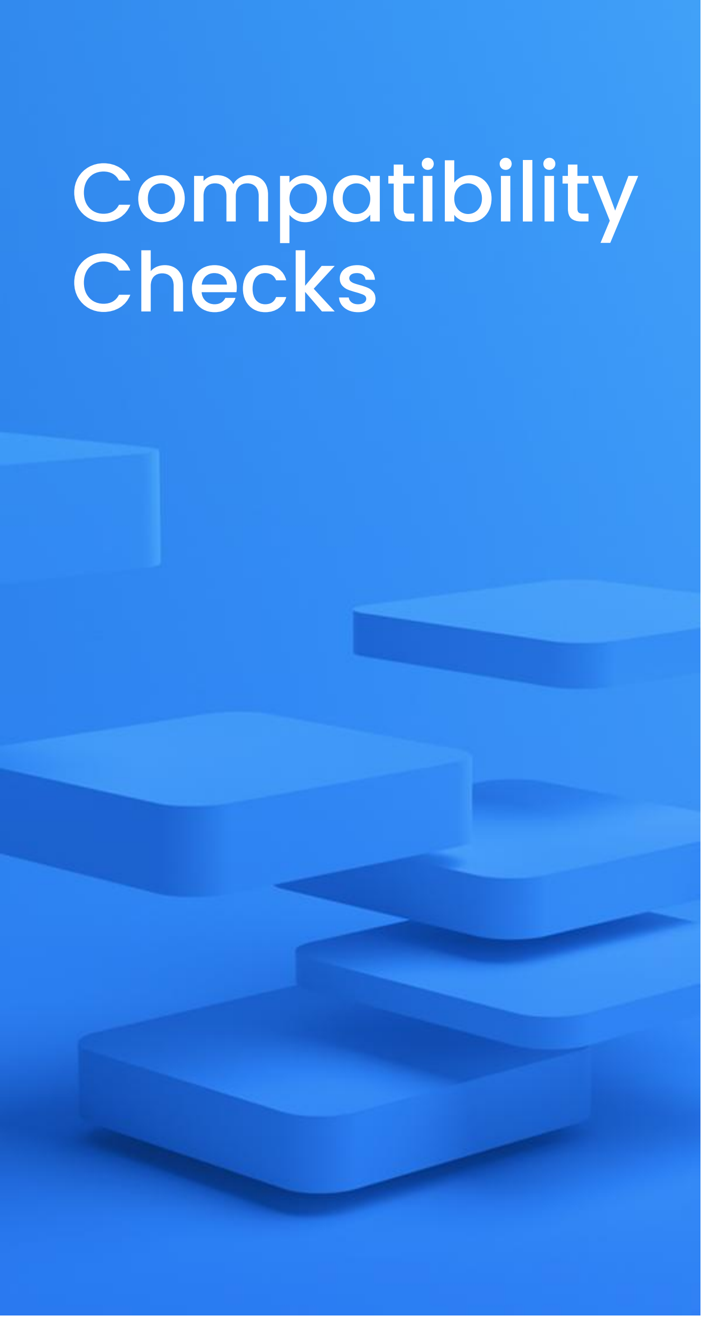
```
Warning: The following objects have names that conflict with new reserved
  keywords. Ensure queries sent by your applications use `quotes` when
  referring to them or they will result in errors.
 More information:
```

https://dev.mysql.com/doc/refman/en/keywords.html

```
 db.t1.rank - Column name
 dbr.t2.reads - Column name
```

PERCONA FOSDEM'23

Compatibility Checks



Removed System Variables

Example:

```
Error: Following system variables that were detected as being used will be
   removed. Please update your system to not rely on them before the upgrade.
```

More information:

https://dev.mysql.com/doc/refman/8.0/en/added-deprecated-removed.html#optvars-removed

```
  query_cache_limit - is set and will be removed
  query_cache_size - is set and will be removed
  query_cache_type - is set and will be removed
  show_compatibility_56 - is set and will be removed
```

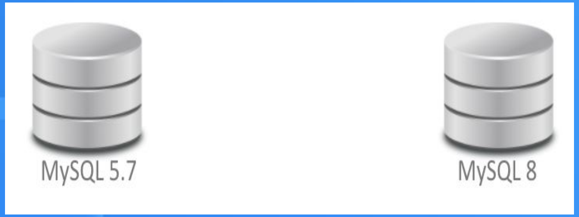
Percona Utilities help with Version Upgrades

- pt-upgrade
- pt-query-digest
- pt-config-diff

Process Overview

1. Build a Test Environment
2. Confirm Application will work on new Major version MySQL
3. Plan Upgrade Strategy - in place or stand-up a new environment
4. Execute Upgrade or Cutover to New Environment.

pt-upgrade



Hardware & Software Requirements

- Two servers that meet production specs.
- Connected to the same network.
- Test server one with the current MySQL version running in the production.
- Test server two with the target version.

For example, to test MySQL 5.7 and MySQL 8, we build the two instances, one with MySQL 5.7 and the other with MySQL 8, both from a recent production backup.

pt-config-diff

Validate configuration differences

pt-config-diff

The tool aids in determining the differences in MySQL settings between files and server variables. This allows us to compare the upgraded version to the previous version, allowing us to validate the configuration differences.

The tool assists us in determining the configuration difference between freshly installed MySQL instances, the old version, and the targeted version, as well as reviewing and evaluating the default configuration differences.

We may perform the same thing while setting up the test environment and before starting the tests to check the MySQL configuration changes, as well as during the production upgrade process to confirm the different MySQL instance settings.

```
pt-config-diff  h=<host1> h=<host2>  --report-width=200
```

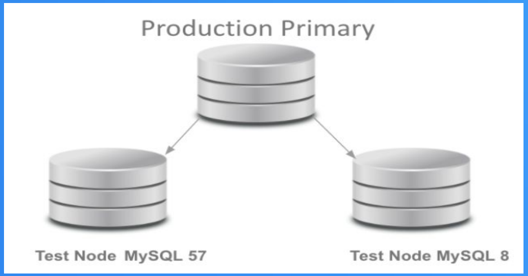
PERCONA FOSDEM'23

Testing Overview

Replication Compatibility

Read Only Test

Read/Write Test



Replication Compatibility Test

Set up replication for both test nodes for a week, replicating from the production primary to see if the replication works from the lower current production version to the newer version, i.e. covering the actual application workload.

Percona XtraBackup can be used to backup and restore the backup from the production node to the test nodes.

Before we begin pt-upgrade testing, it’s important to stop replication on both test nodes at the same binary log position to confirm the data in both nodes are identical.

Slow query log

Query collection

The slow log can be used to collect all queries from the production nodes using long_query_time = 0 and log_slow_rate_limit =1.

Stop the test nodes replication as immediately as the slow log gathering begins.

Ensure that all application queries are covered by the slow log collection.

Specific queries for the weekly, monthly, and annual jobs may be tested separately.

Query digest preparation

The pt-query-digest tool can help with query digest preparation for pt-upgrade testing.

In the digest, for example, we can take a max of 50 samples per query.

Example :

```
pt-query-digest --sample 50 --no-report --output slowlog <slow_log_file> > <digest>.out
```

pt-upgrade

Read-only Test

All queries will be played back using the pt-upgrade tool in read-only mode a couple of times on test nodes logging the results.

Discard the first run’s results because this is just to warm up the Innodb Buffer pool, and run the same command again.

Example :

```
pt-upgrade h=Test1 h=Test2 --max-examples=1
<digest>.out 1>   pt-upgrade_results.out 2>
pt-upgrade_results.err
```

pt-upgrade

Read-Write test

Important to complete **after** read only test is complete.

Recommended to take a backup prior to proceeding with RW testing.

All queries will be played back in read-write mode enabling the –no-read-only option once on the test nodes, logging the results.

Example :

```
pt-upgrade h=Test1 h=Test2  –no-read-only
--max-examples=1 <digest>.out 1>
pt-upgrade_results.out 2> pt-upgrade_results.err
```

pt-upgrade Example Report

```
#-----------------------------------------------------------------------
# Logs
#-----------------------------------------------------------------------

File: pt_upgrade_report_readwrite.out
Size: 55741200

#-----------------------------------------------------------------------
# Hosts
#-----------------------------------------------------------------------

host1:

  DSN:       h=<IP>
  hostname:  <hostname>
  MySQL:     (Ubuntu) 5.7.33

host2:

  DSN:       h=<IP>
  hostname:  <hostname>
  MySQL:     MySQL Community Server - GPL 8.0.15
```

pt-upgrade Example Report

```
##########################################################################
# Query class D3F12166B8804700
##########################################################################

Reporting class because there are 1 row diffs.

Total queries      1
Unique queries     1
Discarded queries  0

select @@sql_mode

##
## Row diffs: 1
##

-- 1.

@ row 1
<
NO_AUTO_VALUE_ON_ZERO,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION
_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION
>
NO_AUTO_VALUE_ON_ZERO,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION
_BY_ZERO,NO_ENGINE_SUBSTITUTION

SELECT @@SQL_MODE
```

pt-upgrade Example Report

explicit_defaults_for_timestamp :

This system variable determines whether the server enables certain nonstandard behaviors for default values and NULL-value handling in TIMESTAMP columns.

```
################################################################################
# Query class B14313EB59AC6113
################################################################################

Reporting class because there are 1 query errors.

Total queries      1
Unique queries     1
Discarded queries  0

insert into testtable (e_id, date) values(?+)

##
## Query errors diffs: 1
##

-- 1.

No error

vs.

DBD::mysql::st execute failed: Column 'date' cannot be null [for Statement "INSERT INTO testtable
(e_id, date VALUES (20, NULL)"]

INSERT INTO testtable (e_id, date) VALUES (20, NULL)

mysql> SHOW CREATE TABLE testtable\G
*************************** 1. row ***************************
       Table: testtable
Create Table: CREATE TABLE `testtable` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `e_id` int(11) DEFAULT NULL,
  `date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`),
  KEY `index1` (`e_id`)
) ENGINE=InnoDB AUTO_INCREMENT=347416796 DEFAULT CHARSET=utf8 ROW_FORMAT=COMPRESSED
ENCRYPTION='Y'
```

pt-upgrade Example Report

explicit_defaults_for_timestamp :

MySQL 8.0 : By default, explicit_defaults_for_timestamp is enabled, which disables the nonstandard behaviors. Disabling explicit_defaults_for_timestamp results in a warning.

MySQL 5.7:

```
mysql>  CREATE TABLE `testtable` (
    ->   `id` bigint(20) NOT NULL AUTO_INCREMENT,
    ->   `e_id` int(11) DEFAULT NULL,
    ->   `date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
    ->   PRIMARY KEY (`id`),
    ->   KEY `index1` (`e_id`)
    -> ) ENGINE=InnoDB AUTO_INCREMENT=347416795 DEFAULT CHARSET=utf8 ROW_FORMAT=COMPRESSED ENCRYPTION='Y';
Query OK, 0 rows affected (0.08 sec)

mysql> INSERT INTO testtable (e_id,date) VALUES (10,  NULL);
Query OK, 1 row affected (0.01 sec)
```

MySQL 8.0.15:

```
mysql>  CREATE TABLE `testtable` (
    ->   `id` bigint(20) NOT NULL AUTO_INCREMENT,
    ->   `e_id` int(11) DEFAULT NULL,
    ->   `date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
    ->   PRIMARY KEY (`id`),
    ->   KEY `index1` (`e_id`)
    -> ) ENGINE=InnoDB AUTO_INCREMENT=347416795 DEFAULT CHARSET=utf8 ROW_FORMAT=COMPRESSED ENCRYPTION='Y';
Query OK, 0 rows affected, 1 warning (0.08 sec)

mysql> INSERT INTO testtable (e_id,date) VALUES (10,  NULL);
ERROR 1048 (23000): Column 'date' cannot be null
mysql> INSERT INTO testtable (e_id,date) VALUES (10,'2022-03-07 03:38:04');
Query OK, 1 row affected (0.01 sec)

mysql>  CREATE TABLE `testtable1` (
    ->   `id` bigint(20) NOT NULL AUTO_INCREMENT,
    ->   `e_id` int(11) DEFAULT NULL,
    ->   `date` timestamp DEFAULT CURRENT_TIMESTAMP,
    ->   PRIMARY KEY (`id`),
    ->   KEY `index1` (`e_id`)
    -> ) ENGINE=InnoDB AUTO_INCREMENT=347416795 DEFAULT CHARSET=utf8 ROW_FORMAT=COMPRESSED ENCRYPTION='Y';
Query OK, 0 rows affected, 1 warning (0.05 sec)

mysql> INSERT INTO testtable1 (e_id,date) VALUES (10,  NULL);
Query OK, 1 row affected (0.00 sec)
```

pt-upgrade Example Report

```
#############################################################################
# Query class E3BD0796B9DDFF8E
#############################################################################

Reporting class because there are 1 query diffs.
Total queries      1
Unique queries     1
Discarded queries  0

select * from testtable where col1=? and (col2=?) order by col3 desc

##
## Query time diffs: 1
##

-- 1.

0.137593 vs. 3.904580 seconds (28.4x increase)

SELECT * FROM stock_move_reqs  WHERE colm1=0 AND (col2=1) ORDER BY
col3 DESC
```

pt-upgrade Example Report

```
##############################################################################
# Query class 7A4E2F660EB50074
##############################################################################

Reporting class because there are 1 row diffs.

Total queries      1
Unique queries     1
Discarded queries  0

select * from testtable

##
## Row diffs: 1
##

-- 1.

@ first 3 of 2 missing rows                ⇐ The two records are missing.
< 347416796,10,2022-03-07 07:07:53
< 347416797,10,2022-03-07 07:07:55
<

@ row 1                                    ⇐ Between the two instances, this record does not match.
< 347416795,10,2022-03-07 03:50:16
> 347416795,10,2022-03-07 03:38:04

select * from testtable

MySQL 5.7:
mysql> select * from testtable;
+-----------+------+---------------------+
| id        | e_id | date                |
+-----------+------+---------------------+
| 347416795 |   10 | 2022-03-07 03:50:16 |
| 347416796 |   10 | 2022-03-07 07:07:53 |
| 347416797 |   10 | 2022-03-07 07:07:55 |
+-----------+------+---------------------+

MySQL 8:
mysql> select * from testtable;
+-----------+------+---------------------+
| id        | e_id | date                |
+-----------+------+---------------------+
| 347416795 |   10 | 2022-03-07 03:38:04 |
+-----------+------+---------------------+
```

Plan Upgrade Strategy

In-Place Upgrade

- Less additional infrastructure cost (still need the test nodes)
- Upgrade completed over weeks with cooldown periods between reader node upgrades
- Need to failover production traffic, for short downtime need to have good HA tools

Stand-up a New Environment and Cutover

- Will have additional infrastructure cost
- Upgrade both OS and DBMS at the same time
- Allows upgrade of hardware concurrent to other
- Only one cutover window

In-Place Upgrade (Async)

An in-place Major version upgrade.

- Set innodb_fast_shutdown=0
- Stop the MySQL and backup the config (my.cnf).
- Remove the old binaries to install the targeted version.
- Update the (my.cnf)
- Start MySQL server.
- Run mysql_upgrade(As of MySQL 8.0.16, the MySQL server performs the upgrade tasks previously handled by mysql_upgrade.)
- Restart MySQL Server

In-Place Upgrade (Async)

If you use XA transactions with InnoDB, run XA RECOVER before upgrading to check for uncommitted XA transactions. If results are returned, either commit or rollback the XA transactions by issuing an XA COMMIT or XA ROLLBACK statement.

If we're doing an in-place update, we'll upgrade the replication instances first, then fail over to the current master utilising technologies like orchestrator and MHA, etc.

In-Place Upgrade (Async)

In-Place Upgrade (Async)

What is Orchestrator ?

Orchestrator is a replication topology manager for MySQL.

The replication tree's topology and status are automatically identified and monitored.

To check the status and perform activities like failovers, you can utilise a GUI, CLI, or API.

What is VIP controller ?

The custom ip controller script manages the primary, and replica, correspondingly the script manage the writer vip and [optional] reader vip. The script will look to identify the master by a RW state to add the writer vip, and will look to identify the slave by a RO state to add the reader vip.

In-Place Upgrade (Async)

What is Proxy SQL ?

ProxySQL is a high-performance SQL proxy, which is a Load Balancer.

How does ProxySQL decide who the new master is?

ProxySQL has no idea how the topology appears, which is crucial information.ProxySQL monitors the MySQL servers' "read only" variables, and the server with read only=off will get the writes.If the old master fails and the topology is changed, the read only variables on the new master must be adjusted. Of course, we can use tools like MHA or Orchestrator to help us with this.

Or with help of hooks in `PreGracefulTakeoverProcesses` and `PostGracefulTakeoverProcesses` for Orchestrator

Cutover (Async)

Stand up New Environment & Cutover (Async)

Provision a duplicate environment with the same number of servers with the same hardware specs and same operating system.

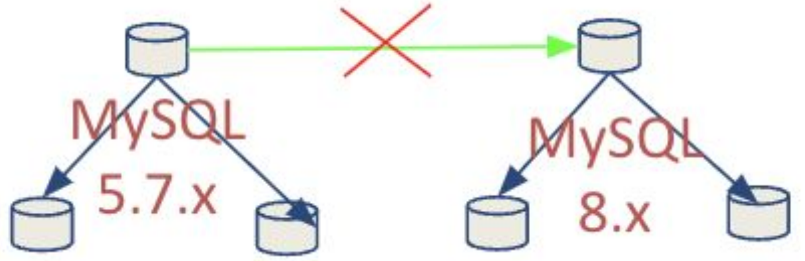
On the newly provided hardware, the target MySQL version will be installed. The new environment will be set up, and the production data will be recovered. Run pt-config-diff to verify MySQL configurations.

Replication from the current source to the newly built environment will be established.

At the cutover time, all writes on the current source will halt, and the application will redirect traffic to the new source. Cutover can be done using a VIP or manually redirecting the application

Once writes are being received on new environment, you are in a fail forward situation.

The old environment will be torn down.



Percona XtraDB Cluster or Galera Considerations

Rolling upgrade

We can use this approach for upgrading Percona XtraDB Cluster without downtime to the latest 5.7 version.

A "rolling upgrade" avoids the need to shut down the entire cluster during the upgrade process.

This method may be used for both major (from 5.6 to 5.7 version) and minor (from 5.7.x to 5.7.y) upgrades.

Rolling upgrades to 5.7 from earlier versions than 5.6 are not supported.

Therefore if you are running Percona XtraDB Cluster version 5.5, it is recommended to shutdown all nodes, then remove and re-create the cluster from scratch.

Mixing PXC 5.7 and PXC 8.0 nodes is not recommended.

Rolling upgrade

It is not advised to mix PXC 5.7 and PXC 8.0 nodes.

The variable pxc-encrypt-cluster-traffic (set to ON versus OFF in PXC 5.7)

We cannot join a 5.7 node to a PXC 8.0 cluster unless traffic encryption is set on the node, as the cluster may have some nodes with traffic encryption enabled and others with traffic encryption disabled.

Each node in your cluster must have the same SSL certificates or try to join a cluster running PXC 5.7 which unencrypted cluster traffic, the node will not be able to join.

In-Place with Downtime

Shut down the cluster and update each node to PXC 8.0 from 5.7

This strategy necessitates application downtime because we are shutting down all cluster nodes and updating.

Before performing an upgrade, it is essential that you maintain backups.

Cutover method

Alternatively, we can build a new PXC 8.0 cluster environment enabling async replication from the PXC 5.7 cluster and do a last-minute cutover to reduce application downtime.

Provision a duplicate environment with the same number of servers with the same hardware specs and same operating system.

On the newly provided hardware, the target MySQL version will be installed. The new cluster environment will be set up, and the production data will be recovered. Run pt-config-diff to verify MySQL configurations.

Async replication from the current source to the newly built environment will be established (Any one of the 5.7 nodes to one of the PXC 8.0 nodes.)

At the cutover time, all writes on the current source will halt, and the application will redirect traffic to the new source. Cutover can be done using a VIP or manually redirecting the application

Once writes are being received on new environment, you are in a fail forward situation. The old environment will be torn down.

Final Notes

Test extensively prior to upgrade.

Limit the risk of encountering unforeseen situations.

Address any unfavorable results prior to upgrade.

Downgrading a major is not feasible, once writes are sent to the source.

Be equipped with the tools needed.

Reduce the application downtime

References

Percona Utilities That Make Major MySQL Version Upgrades Easier :

https://www.percona.com/blog/percona-utilities-that-make-major-mysql-version-upgrades-easier/

Two Extremely Useful Tools (pt-upgrade and checkForServerUpgrade) for MySQL Upgrade Testing :

https://www.percona.com/blog/two-extremely-useful-tools-pt-upgrade-and-checkforserverupgrade-for-mysql-upgrade-testing/

Questions?

Thank you!

Percona is a world-class open source database software, support, and services company focused on helping you scale and innovate with speed as you grow.

83M+	100K+	800+	40%
Software Downloads	Blog Views Per Month	Customers	YoY MRR Growth
TTM as of November 2022			As of Q4 2021

Trusted by...

- More than a third of the Fortune 50
- 4 of the top 6 Retailers
- 3 of the top 5 Healthcare Companies
- 9 of the top 10 Tech Companies
- 6 of the top 10 Gaming Companies
- 4 of the top 5 Manufacturing Companies

Percona Live Denver'23 Ticket Raffle

https://learn.percona.com/win-percona-live-pass-fosdem

1. Scan QR code
2. Fill out a form

That's it!

One winner will be selected randomly

careers@percona.com

Are you passionate about Open Source?!

We're hiring!

Join us!

#RemoteWork

www.percona.com/about-percona/careers

Thank You!

Daniil Bazhenov

Community Manager, Percona

daniil.bazhenov@percona.com

linkedin.com/in/dbazhenov/

Twitter: @dbazhenov

percona.com