Percona XtraDB Cluster Documentation

5.7.44-31.65 (2023-12-26)

Percona Technical Documentation Team

Percona LLC and/or its affiliates © 2023 — Cookie Consent

Table of contents

1. Percona XtraDB Cluster 5.7 Documentation 5
2. Release notes 7
2.1 Percona XtraDB Cluster 5.7 Release notes index 7
2.2 Percona XtraDB Cluster 5.7.44-31.65 (2023-12-26) 8
2.3 Percona XtraDB Cluster 5.7.43-31.65 (2023-08-31) 8
2.4 Percona XtraDB Cluster 5.7.42-31.65 (2023-06-08) 9
2.5 Percona XtraDB Cluster 5.7.41-31.65 (2023-04-05) 10
2.6 Percona XtraDB Cluster 5.7.40-31.63 (2022-12-29) 11
2.7 Percona XtraDB Cluster 5.7.39-31.61 (2022-08-29) 12
2.8 Percona XtraDB Cluster 5.7.38-31.59 (2022-06-29) 12
2.9 Percona XtraDB Cluster 5.7.37-31.57 (2022-05-18) 13
2.10 Percona XtraDB Cluster 5.7.36-31.55 14
2.11 Percona XtraDB Cluster 5.7.35-31.53 15
2.12 Percona XtraDB Cluster 5.7.34-31.51 16
2.13 Percona XtraDB Cluster 5.7.33-31.49 16
2.14 Percona XtraDB Cluster 5.7.32-31.47 17
2.15 Percona XtraDB Cluster 5.7.31-31.45 17
2.16 Percona XtraDB Cluster 5.7.31-31.45.2 18
2.17 Percona XtraDB Cluster 5.7.31-31.45.3 18
2.18 Percona XtraDB Cluster 5.7.30-31.43 18
2.19 Percona XtraDB Cluster 5.7.29-31.43 19
2.20 Percona XtraDB Cluster 5.7.28-31.41 20
2.21 Percona XtraDB Cluster 5.7.28-31.41.2 20
2.22 Percona XtraDB Cluster 5.7.27-31.39 21
2.23 Percona XtraDB Cluster 5.7.26-31.37 21
2.24 Percona XtraDB Cluster 5.7.25-31.35 22
2.25 Percona XtraDB Cluster 5.7.24-31.33 22
2.26 Percona XtraDB Cluster 5.7.23-31.31 23
2.27 Percona XtraDB Cluster 5.7.23-31.31.2 24
2.28 Percona XtraDB Cluster 5.7.22-29.26 25
2.29 Percona XtraDB Cluster 5.7.21-29.26 26
2.30 Percona XtraDB Cluster 5.7.20-29.24 27
2.31 Percona XtraDB Cluster 5.7.19-29.22 28
2.32 Percona XtraDB Cluster 5.7.19-29.22-3 30
2.33 Percona XtraDB Cluster 5.7.18-29.20 31

Percona LLC and/or its affiliates © 2023 — Cookie Consent

2.34 Percona XtraDB Cluster 5.7.17-29.20 32
2.35 Percona XtraDB Cluster 5.7.17-27.20 33
2.36 Percona XtraDB Cluster 5.7.16-27.19 34
2.37 Percona XtraDB Cluster 5.7.14-26.17 36
2.38 Percona XtraDB Cluster 5.7.12-5rc1-26.16 37
2.39 Percona XtraDB Cluster 5.7.11-4beta-25.14.2 38
3. Introduction 40
3.1 About Percona XtraDB Cluster 40
3.2 Percona XtraDB Cluster Limitations 41
3.3 Percona XtraDB Cluster 5.7 version numbers 43
3.4 Telemetry on Percona XtraDB Cluster 43
4. Getting Started 45
4.1 Quick Start Guide for Percona XtraDB Cluster 45
4.2 Installing Percona XtraDB Cluster 46
4.3 Configuring Nodes for Write-Set Replication 46
4.4 Bootstrapping the First Node 49
4.5 Adding Nodes to Cluster 50
4.6 Verifying Replication 52
5. Features 54
5.1 High Availability 54
5.2 Multi-Source Replication 54
5.3 PXC Strict Mode 56
6. PXC Security 62
6.1 Security Basics 62
6.2 Securing the Network 63
6.3 Encrypting PXC Traffic 65
7. User's Manual 74
7.1 State Snapshot Transfer 74
7.2 Percona XtraBackup SST Configuration 76
7.3 Restarting the cluster nodes 85
7.4 Cluster Failover 86
7.5 Monitoring the cluster 87
7.6 Certification in Percona XtraDB Cluster 90
7.7 Percona XtraDB Cluster threading model 94
7.8 Understanding GCache and Record-Set Cache 95
7.9 Perfomance Schema Instrumentation 96
7.10 Data at Rest Encryption 98
7.11 Configure Systemd TimeoutStartSec 106

Percona LLC and/or its affiliates © 2023 — Cookie Consent

8. Flexibility 108
8.1 Binlogging and replication improvements 108
8.2 InnoDB Full-Text Search improvements 110
8.3 Multiple page asynchronous I/O requests 111
9. Diagnostics 113
9.1 InnoDB Page Fragmentation Counters 113
9.2 Using libcoredumper 114
9.3 Stack Trace 115
10. How-tos 117
10.1 Upgrading Percona XtraDB Cluster 117
10.2 Crash Recovery 121
10.3 Configuring Percona XtraDB Cluster on CentOS 125
10.4 Configuring Percona XtraDB Cluster on Ubuntu 131
10.5 Setting up Galera Arbitrator 136
10.6 How to set up a three-node cluster on a single box 138
10.7 How to set up a three-node cluster in EC2 environment 142
10.8 Load balancing with HAProxy 144
10.9 Load balancing with ProxySQL 146
10.10 Setting up PXC reference architecture with HAProxy 157
11. References 165
11.1 Index of wsrep status variables 165
11.2 Index of wsrep system variables 170
11.3 Index of wsrep_provider options 191
11.4 Index of files created by PXC 215
11.5 Frequently Asked Questions 218
11.6 Glossary 221
11.7 Copyright and licensing information 225
11.8 Trademark policy 225

Percona LLC and/or its affiliates © 2023 — Cookie Consent

1. Percona XtraDB Cluster 5.7 Documentation

Percona XtraDB Cluster is a database clustering solution for MySQL. It ensures high availability, prevents downtime and data loss, and provides linear scalability for a growing environment.

This documentation is for the latest release: Percona XtraDB Cluster 5.7.44-31.65 (Release Notes).

Features of Percona XtraDB Cluster include:

Synchronous replication <Database replication>

Data is written to all nodes simultaneously, or not written at all if it fails even on a single node.

Multi-source replication

Any node can trigger a data update.

True parallel replication <Database replication>

Multiple threads on slave performing replication on row level.

Automatic node provisioning

You simply add a node and it automatically syncs.

Data consistency

Percona XtraDB Cluster ensures that data is automatically synchronized on all `nodes <Node>` in your cluster.

PXC Strict Mode

Avoids the use of experimental and unsupported features.

Configuration script for ProxySQL

Percona provides a ProxySQL package with the `proxysql-admin` tool that automatically configures Percona XtraDB Cluster nodes.

See also

Load balancing with ProxySQL

Automatic configuration of SSL encryption

Percona XtraDB Cluster includes the `pxc-encrypt-cluster-traffic` variable that enables automatic configuration of SSL encryption.

Optimized Performance

Percona XtraDB Cluster performance is optimized to scale with a growing production workload.

For more information, see the following blog posts:

- How We Made Percona XtraDB Cluster Scale
- Performance improvements in Percona XtraDB Cluster 5.7.17-29.20

Percona LLC and/or its affiliates © 2023 — Cookie Consent

Percona XtraDB Cluster is fully compatible with MySQL Server Community Edition, Percona Server, and MariaDB. It provides a robust application compatibility: there is no or minimal application changes required.

Last update: 2023-02-01

Percona LLC and/or its affiliates © 2023 — Cookie Consent

2. Release notes

2.1 Percona XtraDB Cluster 5.7 Release notes index

- Percona XtraDB Cluster 5.7.44-31.65 (2023-12-26)
- Percona XtraDB Cluster 5.7.43-31.65 (2023-08-31)
- Percona XtraDB Cluster 5.7.42-31.65 (2023-06-08)
- Percona XtraDB Cluster 5.7.41-31.65 (2023-04-05)
- Percona XtraDB Cluster 5.7.40-31.63 (2022-12-29)
- Percona XtraDB Cluster 5.7.39-31.61 (2022-08-29)
- Percona XtraDB Cluster 5.7.38-31.59 (2022-06-29)
- *Percona XtraDB Cluster* 5.7.37-31.57 (2022-05-18)
- Percona XtraDB Cluster 5.7.36-31.55 (2022-02-16)
- Percona XtraDB Cluster 5.7.35-31.53 (2021-11-18)
- Percona XtraDB Cluster 5.7.34-31.51 (2021-07-19)
- Percona XtraDB Cluster 5.7.33-31.49 (2021-03-22)
- Percona XtraDB Cluster 5.7.32-31.47 (2021-01-12)
- Percona XtraDB Cluster 5.7.31-31-45.3 (2020-10-22)
- Percona XtraDB Cluster 5.7.31-31.45.2 (2020-10-09)
- Percona XtraDB Cluster 5.7.31-31.45 (2020-09-24)
- Percona XtraDB Cluster 5.7.30-31.43 (2020-06-25)
- Percona XtraDB Cluster 5.7.29-31.43 (2020-05-08)
- Percona XtraDB Cluster 5.7.28-31.41.2 (2020-04-14)
- Percona XtraDB Cluster 5.7.28-31.41 (2019-12-16)
- Percona XtraDB Cluster 5.7.27-31.39 (2019-09-18)
- Percona XtraDB Cluster 5.7.26-31.37 (2019-06-26)
- Percona XtraDB Cluster 5.7.25-31.35 (2019-02-28)
- Percona XtraDB Cluster 5.7.24-31.33 (2019-01-04)
- Percona XtraDB Cluster 5.7.23-31.31.2 (2018-10-02)
- Percona XtraDB Cluster 5.7.23-31.31 (2018-09-26)
- Percona XtraDB Cluster 5.7.22-29.26 (2018-06-29)
- Percona XtraDB Cluster 5.7.21-29.26 (2018-03-02)
- Percona XtraDB Cluster 5.7.20-29.24 (2018-01-26)
- Percona XtraDB Cluster 5.7.19-29.22-3 (2017-10-27)
- Percona XtraDB Cluster 5.7.19-29.22 (2017-09-22)
- Percona XtraDB Cluster 5.7.18-29.20 (2017-06-02)
- Percona XtraDB Cluster 5.7.17-29.20 (2017-04-19)
- Percona XtraDB Cluster 5.7.17-27.20 (2017-03-16)
- Percona XtraDB Cluster 5.7.16-27.19 (2016-12-15)
- Percona XtraDB Cluster 5.7.14-26.17 (2016-09-29)
- Percona XtraDB Cluster 5.7.12-5rc1-26.16 (2016-08-09)

Percona LLC and/or its affiliates © 2023 — Cookie Consent

- Percona XtraDB Cluster 5.7.11-4beta-25.14.2 (2016-06-09)

Last update: 2023-12-07

2.2 Percona XtraDB Cluster 5.7.44-31.65 (2023-12-26)

Important

Percona XtraDB Cluster 5.7.44-31.65 is the final release of the Percona XtraDB Cluster 5.7 series.

Do you need help upgrading to MySQL 8.0? Do you need to stay on MySQL 5.7 a bit longer? We will support you either way.

Percona XtraDB Cluster (PXC) supports critical business applications in your public, private, or hybrid cloud environment.

2.2.1 Release highlights

Percona XtraDB Cluster implements telemetry that fills the gaps in our understanding of how you use XtraDB Cluster to improve our products. Participation in the anonymous program is optional. You can opt-out if you prefer not to share this information. Find more details in the Telemetry on Percona XtraDB Cluster document.

Improvements and bug fixes provided by Oracle for MySQL 5.7.44 and included in the Percona XtraDB Cluster are the following:

- Upgraded the linked Open SSL library to OpenSSL 3.0.10
- Removed the printed query string limit to display the characters for a detected deadlock section of the engine status log

Find the complete list of additions or bug fixes in the MySQL 5.7.44 Release Notes.

2.2.2 Useful links

Install Percona XtraDB Cluster

Percona Product downloads

The Percona XtraBackup GitHub repository

Contribute to the documentation

For training, contact Percona Training - Start learning now

Last update: 2023-12-07

2.3 Percona XtraDB Cluster 5.7.43-31.65 (2023-08-31)

Percona XtraDB Cluster (PXC) supports critical business applications in your public, private, or hybrid cloud environment. Our free, open source, enterprise-grade solution includes the high availability and the security features your business requires to meet your customer expectations and business goals.

Percona LLC and/or its affiliates © 2023 — Cookie Consent

2.3.1 Release highlights

The PXC-4218 bug fix requires the upgrade procedure in the Upgrading Percona XtraDB Cluster guide. Be sure to open another session and run `mysql_upgrade` as part of the process.

Improvements and bug fixes provided by Oracle for MySQL 5.7.43 and included in Percona XtraDB Cluster are the following:

- OpenSSL 1.1.1 library has been upgraded to OpenSSL 3.0.9.

Find the full list of bug fixes and changes in the MySQL 5.7.43 Release Notes.

2.3.2 Bug fixes

- PXC-4218: The output of SHOW VARIABLES LIKE 'wsrep_provider_options' was truncated. This bug fix requires the upgrade procedure in the Upgrading Percona XtraDB Cluster guide. Be sure to open another session and run `mysql_upgrade` as part of the process.
- PXC-4241: `Percona-XtraDB-Cluster-server-57` RPM package on EL9 did not obsolete `mariadb-connector-c-config.noarch`.

2.3.3 Useful links

Install Percona XtraDB Cluster

Percona Product downloads

The Percona XtraBackup GitHub location

Contribute to the documentation

For training, contact Percona Training - Start learning now

Last update: 2023-08-31

2.4 Percona XtraDB Cluster 5.7.42-31.65 (2023-06-08)

Release date	**June 08, 2023**
Install instructions	Install Percona XtraDB Cluster

Percona XtraDB Cluster (PXC) supports critical business applications in your public, private, or hybrid cloud environment. Our free, open source, enterprise-grade solution includes the high availability and the security features your business requires to meet your customer expectations and business goals.

2.4.1 Release highlights

Improvements and bug fixes provided by Oracle for MySQL 5.7.42 and included in Percona Server for MySQL are the following:

- In InnoDB, online DDL operations are prevented from accessing out-of-bounds memory

Find the full list of bug fixes and changes in the MySQL 5.7.42 Release Notes.

Percona LLC and/or its affiliates © 2023 — Cookie Consent

2.4.2 Bug fixes

- PXC-667 : The server exited during a BF-abort of the active HANDLER `<table>` OPEN AS `<alias>`.
- PXC-679 : Undetected state gap discovery causes server to hang on shutdown.

2.4.3 Useful links

Percona Product downloads

The Percona XtraBackup GitHub location

Contribute to the documentation

For training, contact Percona Training - Start learning now

Last update: 2023-06-05

2.5 Percona XtraDB Cluster 5.7.41-31.65 (2023-04-05)

Release date	**April 5, 2023**
Install instructions	Install Percona XtraDB Cluster

Percona XtraDB Cluster (PXC) supports critical business applications in your public, private, or hybrid cloud environment. Our free, open source, enterprise-grade solution includes the high availability and the security features your business requires to meet your customer expectations and business goals.

2.5.1 Release highlights

Percona has removed an Oracle patch for `mysqldump` that performed, at the beginning of the dump, a `FLUSH_TABLES_WITH_READ_LOCK` to get consistent `GTID_EXECUTED` because the patch required additional user privileges, even when the user does not use GTID-based replication. The following bugs based on this patch were submitted to Oracle:

- MySQL 109701
- MySQL 109685

The Percona solution uses `START TRANSACTION WITH CONSISTENT SNAPSHOT`.

Improvements and bug fixes provided by Oracle for MySQL 5.7.41 and included in Percona Server for MySQL are the following:

- Updated the linked OpenSSL library for MySQL Server to 1.1.1s
- Updated the bundled zlib library to zlib 1.2.13. This zlib library version is the minimum supported.
- While the SQL thread handled a transaction, issuing `STOP SLAVE_SQL_THREAD` caused replication to stop immediately instead of waiting for the event group to complete before the shutdown.

Find the full list of bug fixes and changes in the MySQL 5.7.41 Release Notes.

2.5.2 Bug fixes

- PXC-2687 : A brute force abort of a stored procedure caused a write-after-free error.

Percona and/or its affiliates © 2023 — Cookie Consent

2.5.3 Useful links

Percona Product downloads

The Percona XtraBackup GitHub location

Contribute to the documentation

For training, contact Percona Training - Start learning now

Last update: 2023-04-05

2.6 Percona XtraDB Cluster 5.7.40-31.63 (2022-12-29)

Release date	**December 29, 2022**
Install instructions	Install Percona XtraDB Cluster
Download this version	Percona XtraDB Cluster

Percona XtraDB Cluster (PXC) supports critical business applications in your public, private, or hybrid cloud environment. Our free, open source, enterprise-grade solution includes the high availability and the security features your business requires to meet your customer expectations and business goals.

2.6.1 Release highlights

Percona XtraDB Cluster is based on Percona Server for MySQL. Find a complete list of improvements and bug fixes in the Percona Server for MySQL 5.7.40-43 (2022-11-28).

The improvements and bug fixes for MySQL 5.7.40, provided by Oracle, and included in Percona Server for MySQL are the following:

- ISO 8601 timestamps in log messages did not consider daylight saving time when --log-timestamps=SYSTEM was used.
- The GRANT OPTION privilege was treated as related to database operations.
- In specific cases, a TRUNCATE TABLE operation failed to release an acquired mutex.
- A descending b-tree scan raised a debug assertion failure in debug builds.

Find the full list of bug fixes and changes in the MySQL 5.7.40 Release Notes.

2.6.2 Bug fixes

- PXC-3821 : A fix to prevent the replication of a Performance Schema table truncation across the cluster.
- PXC-512 : Replication slave from Percona XtraDB Cluster can exit if `wsrep_forced_binlog_format` =ROW.
- PXC-4077 : Skipping transactions with an empty GTID event introduced GTID inconsistencies between PXC nodes.
- PXC-4087 : A replicated transaction that aborted a local transaction while that transaction executed a rollback caused a server exit.

2.6.3 Useful links

The Percona XtraBackup GitHub location

Percona LLC and/or its affiliates © 2023 — Cookie Consent

Contribute to the documentation

Last update: 2022-12-29

2.7 Percona XtraDB Cluster 5.7.39-31.61 (2022-08-29)

Percona XtraDB Cluster (PXC) supports critical business applications in your public, private, or hybrid cloud environment. Our free, open source, enterprise-grade solution includes the high availability and the security features your business requires to meet your customer expectations and business goals.

2.7.1 Release Highlights

Percona XtraDB Cluster is based on Percona Server for MySQL. Find a complete list of improvements and bug fixes in the Percona Server for MySQL 5.7.39-42 (2022-08-15).

The improvements and bug fixes for MySQL 5.7.39, provided by Oracle, and included in Percona Server for MySQL are the following:

- Enable the `performance_schema_show_processlist` system variable to use an alternative implementation of the `SHOW PROCESSLIST` statement. Instead of querying the thread manager and holding a global mutex, which may affect performance, this implementation queries the active thread data from the Performance Schema and does not require a mutex.

A fresh installation of MySQL 5.7.39 or higher automatically creates the `processlist` table. This table is not automatically created during an upgrade. For an upgrade, you must create the table manually.

- The system variable, `myisam_repair_threads`, and the myisamchk `--parallel-recover` option are removed.

Find the complete list of bug fixes and changes in the MySQL 5.7.39 Release Notes.

Percona has added no improvements or bug fixes for this release.

2.7.2 Useful Links

The Percona XtraDB Cluster installation instructions

The Percona XtraBackup downloads

The Percona XtraBackup GitHub location

To contribute to the documentation, review the Documentation Contribution Guide

Last update: 2022-08-29

2.8 Percona XtraDB Cluster 5.7.38-31.59 (2022-06-29)

Percona XtraDB Cluster (PXC) supports critical business applications in your public, private, or hybrid cloud environment. Our free, open source, enterprise-grade solution includes the high availability and security features your business requires to meet your customer expectations and business goals.

Percona and/or its affiliates © 2023 — Cookie Consent

2.8.1 Release Highlights

Improvements and bug fixes for MySQL 5.7.38, provided by Oracle, and included in Percona Server for MySQL are the following:

- In certain scenarios, a MySQL server connection could fail if the .ibd file for the partition was missing.
- A statement was not written to the slow query log if the statement contained errors and could not be parsed.

Find the complete list of bug fixes and changes in MySQL 5.7.38 Release Notes.

2.8.2 Bugs Fixed

- PXC-3118: A fix for when, using a thread pool, a brute force abort for a metadata locking (MDL) subsystem conflict stalled.
- PXC-3951: Updated how sst-idle-timeout calculates disk usage. The default timeout value for the variable is 120 seconds. If the joiner node does not detect a disk usage difference in that time, the node terminates the SST. The current calculation could return the same value for 120 seconds.
- PXC-2367: When enabled, the event scheduler filled the error logs with a “Ready state reached” message. The message’s logging level was reduced to “debug” and only logged in the error logs when wsrep_debug = 1.
- PXC-3961: Reduced the strictness of checks which had marked GNo=0 invalid. In Percona XtraDB Cluster, writesets are replicated to other nodes with GNo=0. The generation of seqno or GTID occurs later in the pre-commit phase. A recent version had increased the strictness of these checks which caused the GTID event from 5.6 to be considered invalid.

2.8.3 Useful Links

The Percona XtraDB Cluster installation instructions

The Percona XtraBackup downloads

The Percona XtraBackup GitHub location

To contribute to the documentation, review the Documentation Contribution Guide

---

Last update: 2022-07-20

2.9 Percona XtraDB Cluster 5.7.37-31.57 (2022-05-18)

Percona XtraDB Cluster (PXC) supports critical business applications in your public, private, or hybrid cloud environment. Our free, open source, enterprise-grade solution includes the high availability and security features your business requires to meet your customer expectations and business goals.

Percona LLC and/or its affiliates © 2023 — Cookie Consent

2.9.1 Release Highlights

The following lists a number of the notable updates and fixes for MySQL 5.7.37, provided by Oracle, and included in Percona Server for MySQL:

- The performance on debug builds has been improved by optimizing the buf_validate() function in the *InnoDB* sources.
- Fix for when a query using an index that differs from the primary key of the partitioned table results in excessive CPU load.
- Enabling `PAD_CHAR_TO_FULL_LENGTH` SQL mode on a replica server added trailing spaces to a replication channel's name in the replication metadata repository tables. Attempts to identify the channel using the padded name caused errors. The SQL mode is disabled when reading from those tables.

Find the complete list of bug fixes and changes in MySQL 5.7.37 Release Notes.

2.9.2 Bugs Fixed

- PXC-3388: When the joiner failed and the donor did not abort the SST. The cluster remained in donor/desynced state.
- PXC-3609: The binary log status was updated when the binary log was disabled.
- PXC-3796: The Garbd IP was not visible in the `wsrep_incoming_addresses` status variable.
- PXC-3848: Issuing an `ALTER USER CURRENT_USER()` command crashed the connected cluster node.
- PXC-3914: Upgraded the version of socat used in the Docker image.

2.9.3 Useful Links

The Percona XtraDB Cluster installation instructions

The Percona XtraBackup downloads

The Percona XtraBackup GitHub location

To contribute to the documentation, review the Documentation Contribution Guide

---

Last update: 2022-08-24

2.10 Percona XtraDB Cluster 5.7.36-31.55

- **Date**

  February 16, 2022

- **Installation**

  Installing Percona XtraDB Cluster

Percona XtraDB Cluster (PXC) supports critical business applications in your public, private, or hybrid cloud environment. Our free, open source, enterprise-grade solution includes the high availability and security features your business requires to meet your customer expectations and business goals.

Percona and/or its affiliates © 2023 — Cookie Consent

2.10.1 Release Highlights

The following list contains some of the bug fixes for MySQL 5.7.36, provided by Oracle, and included in Percona XtraDB Cluster for MySQL:

- Fix for the possibility of a deadlock or failure when an undo log truncate operation is initiated after an upgrade from *MySQL* 5.6 to *MySQL* 5.7.
- Fix for when a parent table initiates a cascading `SET NULL` operation on the child table, the virtual column can be set to NULL instead of the value derived from the parent table.
- On a view, the query digest for each SELECT statement is now based on the SELECT statement and not the view definition, which was the case for earlier versions.

Find the complete list of bug fixes and changes in MySQL 5.7.36 Release Notes.

2.10.2 Bugs Fixed

- PXC-3838: Documentation - remove extra space in the Bootstrapping the first node document.
- PXC-3739: Fix for FLUSH TABLES ... FOR EXPORT staying locked after a session ends.
- PXC-3766: Fix the behavior when SST always runs a version-check procedure that causes unallowed external network communication.

Last update: 2022-08-24

2.11 Percona XtraDB Cluster 5.7.35-31.53

- **Date**

  November 18, 2021

- **Installation**

  Installing Percona XtraDB Cluster

Percona XtraDB Cluster (PXC) supports critical business applications in your public, private, or hybrid cloud environment. Our free, open source, enterprise-grade solution includes the high availability and security features your business requires to meet your customer expectations and business goals.

2.11.1 Release Highlights

The following are some of the notable fixes for MySQL 5.7.35, provided by Oracle, and included in this release:

- #104373: Fixes failure of `OPTIMIZE TABLE` command writing to the binary log and replicated to replicas.
- #104451: Fixes which event turns on the `LOG_EVENT_THREAD_SPECIFIC_F` flag.

For more information, see the MySQL 5.7.35 Release Notes

The following are the notable fixes for Galera Cluster, provided by Codership, and included in this release:

- #381: Disables binary log purging when the `mysqld` starts with `--wsrep-recover` option.
- #25551: Disables tables without a primary key from the parallel applying of write sets.

Percona and/or its affiliates © 2023 — Cookie Consent

2.11.2 Bugs Fixed

- PXC-3589: Documentation: Updates in Percona XtraDB Cluster Limitations that the `LOCK=NONE` clause is no longer allowed in an INPLACE ALTER TABLE statement. (Thanks to user Brendan Byrd for reporting this issue)
- PXC-3637: Changes the service start sequence to allow more time for mounting local or remote directories with large amounts of data. (Thanks to user Eric Gonyea for reporting this issue)
- PXC-3741: Fix when a network issue causes the Incremental State Transfer (IST) receiver to stall.

Last update: 2022-08-24

2.12 Percona XtraDB Cluster 5.7.34-31.51

- **Date**

  July 19, 2021

- **Installation**

  Installing Percona XtraDB Cluster

2.12.1 Improvements

- PXC-3634: Erroneous documentation on bootstrapping the XtraDB Cluster (Thanks to user Craig Fisher for reporting this issue)
- PXC-3092: Log a warning at startup if a keyring is specified, but the cluster traffic encryption is turned off

2.12.2 Bugs Fixed

- PXC-3679: SST fails after the update of socat to '1.7.4.0'
- PXC-3611: "Encryption can't find master key" after SST when keyring_file is used if the keyring.backup file exists
- PXC-3608: Attempting to read a FK may cause a memory access violation and server exit.
- PXC-2650: Lack of support for the data-at-rest encryption options are not mentioned in documentation
- PXC-3464: Data is not propagated with SET SESSION sql_log_bin = 0
- PXC-3666: Session with wsrep_on=0 blocks TOI transactions
- PXC-3596: Node stuck in aborting SST
- PXC-3226: Results from CHECK TABLE from PXC server can cause the client libraries to crash
- PXC-3146: Galera/SST does not look at the default data directory location for SSL certs

---

Last update: 2022-08-24

2.13 Percona XtraDB Cluster 5.7.33-31.49

- **Date**

  March 22, 2021

- **Installation**

Percona and/or its affiliates © 2023 — Cookie Consent

Installing Percona XtraDB Cluster

This release fixes security vulnerability CVE-2021-27928, a similar issue to CVE-2020-15180

2.13.1 Bugs Fixed

- PXC-3536: Modify processing to not allow threads/queries to be killed if the thread is in TOI
- PXC-3565: Correct Performance of SELECT in PXC
- PXC-3508: Explicitly set the dhparam option with socat to bypass the use of the old certs

---

Last update: 2022-08-24

2.14 Percona XtraDB Cluster 5.7.32-31.47

- **Date**

  January 12, 2021

- **Installation**

  Installing Percona XtraDB Cluster

2.14.1 Bugs Fixed

- PXC-3468: Resolve package conflict when installing PXC 5.7 on RHEL/CentOS8
- PXC-3459: Modify to pass correct data from row_ins_foreign_check_on_constraint() to wsrep_append_foreign_key()
- PXC-3418: Prevent DDL-DML deadlock by making in-place ALTER take shared MDL for the whole duration.
- PXC-2264: Update Data at Rest Encryption documentation on upgrade and compatibility issues to explain incompatibility when the donor is <= 5.7.21 and joiner is >= 5.7.22
- PXC-3501: Modify wsrep_row_upd_check_foreign_constraints() to include foreign key dependencies in the writesets for DELETE query (Thanks to user Steven Gales for reporting this issue)
- PXC-3442: Fix crash when log_slave_updates=ON and consistency check statement is executed
- PXC-3424: Fix error handling when the donor is not able to serve SST

Last update: 2022-08-24

2.15 Percona XtraDB Cluster 5.7.31-31.45

- **Date**

  September 24, 2020

- **Installation**

  Installing Percona XtraDB Cluster

2.15.1 Improvements

- PXC-2187: Enhance SST documentation to include a warning about the use of command-line parameters

Percona and/or its affiliates © 2023 — Cookie Consent

2.15.2 Bugs Fixed

- PXC-3352: Modify wsrep_row_upd_check_foreign_constraints() to remove the check for DELETE
- PXC-3243: Modify the BF-abort process to propagate and abort and retry the Stored Procedure instead of the statement
- PXC-3371: Fix Directory creation in build-binary.sh
- PXC-3370: Provide binary tarball with shared libs and glibc suffix & minimal tarballs
- PXC-3281: Modify config to add default socket location

Last update: 2022-08-24

2.16 Percona XtraDB Cluster 5.7.31-31.45.2

- **Date**

  October 9, 2020

- **Installation**

  Installing Percona XtraDB Cluster

This release fixes the security vulnerability CVE-2020-15180

Last update: 2022-08-24

2.17 Percona XtraDB Cluster 5.7.31-31.45.3

- **Date**

  October 22, 2020

- **Installation**

  Installing Percona XtraDB Cluster

2.17.1 Bugs Fixed

- PXC-3456: Allow specific characters in SST method names and SST request data.

Last update: 2022-08-24

2.18 Percona XtraDB Cluster 5.7.30-31.43

- **Date**

  June 25, 2020

- **Installation**

  Installing Percona XtraDB Cluster

Percona and/or its affiliates © 2023 — Cookie Consent

2.18.1 Bugs Fixed

- PXC-3170: Backport PXC-3154 - Thread pooling could hang on shutdown
- PXC-3165: Allow COM_FIELD_LIST to be executed when WSREP was not ready

---

Last update: 2022-08-24

2.19 Percona XtraDB Cluster 5.7.29-31.43

- **Date**

  May 8, 2020

- **Installation**

  Installing Percona XtraDB Cluster

2.19.1 Improvements

- PXC-3002: The PXC mysql-systemd parameter service_startup_timeout terminates any startup process after a configurable time. Added a "Disable" option to the service_startup_timeout for workloads which require more time.
- PXC-2259: Updated the "Index of files created by PXC" document with additional file names and descriptions.
- PXC-2197: Modified the SST Documentation to Include Package Dependencies for Percona XtraBackup (PXB).
- PXC-2602: Added the ability to configure xbstream options with wsrep_sst_xtrabackup.

2.19.2 Bugs Fixed

- PXC-2954: DDL to add FK on one node fails but completes on other nodes causing inconsistency
- PXC-2705: Executing parallel LOAD DATA queries against the same table and same source data and the number of rows is greater then 10k, the cluster consistency was broken.
- PXC-2683: An issue may occur if a user set innodb_locks_unsafe_for_binlog=1. The variable is deprecated. The processing for the variable was removed.
- PXC-3202: In CentOS 8, the POSTUN scriptlet in an rpm package would expand into an "empty string" and caused an error.
- PXC-3190: Added to the execution of Cleanup functions to remove CREATE USER from SHOW PROCESSLIST.
- PXC-3076: Modified the Galera SConstruct file to remove python3 components.
- PXC-2969: Modified the "Load balancing with Proxy-SQL" documentation to include the Criteria for Use.
- PXC-2904: Ensured the "Percona-XtraDB-Cluster-57" yum package installed the required xtrabackup version.
- PXC-2958: Modified the User Documentation to include wsrep_certification_rules and the cert.optimistic_pa option.
- PXC-2912: Modified the netcat Configuration to Include the -N option, which is required by more recent versions of netcat. The option allows the shutdown of the network socket after the input EOF.
- PXC-2974: Modified Percona XtraDB Cluster (PXC) Dockerfile to Integrate the Galera WSREP recovery process.

Percona and/or its affiliates © 2023 — Cookie Consent

- PS-6979: Modify the processing to call clean up functions to remove CREATE USER statement from the processlist after the statement has completed (Upstream #99200)
- PXC-2684: Modified error handling to prevent deadlock when stored procedure was aborted.

Last update: 2022-08-24

2.20 Percona XtraDB Cluster 5.7.28-31.41

Percona is happy to announce the release of Percona XtraDB Cluster 5.7.28-31.41 on December 16, 2019. Binaries are available from the downloads section or from our software repositories.

Percona XtraDB Cluster 5.7.28-31.41 is now the current release, based on the following:

- Percona Server for MySQL 5.7.28-31
- Galera/Codership WSREP API Release 5.7.28
- Galera Replication library 3.28

Percona XtraDB Cluster 5.7.28-31.41 requires Percona XtraBackup 2.4.17.

2.20.1 Bugs Fixed

- PXC-2729: A cluster node could hang when trying to access a table which was being updated by another node.
- PXC-2704: After a row was updated with a variable-length unique key, the entire cluster could crash.

Other bugs fixed: PXC-2670

Last update: 2022-07-20

2.21 Percona XtraDB Cluster 5.7.28-31.41.2

- **Date**

  April 14, 2020

- **Installation**

  Installing Percona XtraDB Cluster

Percona XtraDB Cluster 5.7.28-31.41.2 requires Percona XtraBackup 2.4.20.

This release fixes security vulnerability CVE-2020-10996

2.21.1 Bugs Fixed

- PXC-3117: Transition key was hardcoded
- PXB-2142: Transition key was written to backup / stream

Last update: 2022-08-24

Percona and/or its affiliates © 2023 — Cookie Consent

2.22 Percona XtraDB Cluster 5.7.27-31.39

Percona is happy to announce the release of Percona XtraDB Cluster 5.7.27-31.39 on September 18, 2019. Binaries are available from the downloads section or from our software repositories.

Percona XtraDB Cluster 5.7.27-31.39 is now the current release, based on the following:

- Percona Server for MySQL 5.7.27-30
- Galera/Codership WSREP API Release 5.7.27
- Galera Replication library 3.28

2.22.1 Bugs Fixed

- #2432: PXC was not updating the information_schema user/client statistics properly.
- #2555: SST initialization delay: fixed a bug where the SST process took too long to detect if a child process was running.
- #2557: Fixed a crash when a node goes NON-PRIMARY and SHOW STATUS is executed.
- #2592: PXC restarting automatically on data inconsistency.
- #2605: PXC could crash when log_slow_verbosity included InnoDB. Fixed upstream PS-5820.
- #2639: Fixed an issue where a SQL admin command (like OPTIMIZE) could cause a deadlock.

Last update: 2022-07-20

2.23 Percona XtraDB Cluster 5.7.26-31.37

Percona is glad to announce the release of Percona XtraDB Cluster 5.7.26-31.37 on June 26, 2019. Binaries are available from the downloads section or from our software repositories.

Percona XtraDB Cluster 5.7.26-31.37 is now the current release, based on the following:

- Percona Server for MySQL 5.7.26-29
- Galera Replication library 3.26
- Galera/Codership WSREP API Release 5.7.25

2.23.1 Bugs Fixed

- #2480: In some cases, Percona XtraDB Cluster could not replicate `CURRENT_USER()` used in the `ALTER` statement. `USER()` and `CURRENT_USER()` are no longer allowed in any `ALTER` statement since they fail when replicated.
- #2487: The case when a DDL or DML action was in progress from one client and the provider was updated from another client could result in a race condition.
- #2490: Percona XtraDB Cluster could crash when `binlog_space_limit` was set to a value other than zero during `wsrep_recover` mode.
- #2491: SST could fail if the donor had encrypted undo logs.
- #2537: Nodes could crash after an attempt to set a password using `mysqladmin`
- #2497: The user can set the preferred donor by setting the `wsrep_sst_donor` variable. An IP address is not valid as the value of this variable. If the user still used an IP address, an error message was produced that

Percona LLC and/or its affiliates © 2023 — Cookie Consent

did not provide sufficient information. The error message has been improved to suggest that the user check the value of the `wsrep_sst_donor` for an IP address.

Other bugs fixed: #2276, #2292, #2476, #2560

Last update: 2022-07-20

2.24 Percona XtraDB Cluster 5.7.25-31.35

Percona is glad to announce the release of Percona XtraDB Cluster 5.7.25-31.35 on February 28, 2019. Binaries are available from the downloads section or from our software repositories.

This release of Percona XtraDB Cluster includes the support of Ubuntu 18.10 (Cosmic Cuttlefish). Percona XtraDB Cluster 5.7.25-31.35 is now the current release, based on the following:

- Percona Server for MySQL 5.7.25
- Galera Replication library 3.25
- Galera/Codership WSREP API Release 5.7.24

2.24.1 Bugs Fixed

- #2346: `mysqld` could crash when executing `mysqldump --single-transaction` while the binary log is disabled. This problem was also reported in #1711, #2371, #2419.
- #2388: In some cases, `DROP FUNCTION` with an explicit name was not replicated.

Other bugs fixed: #1711, #2371, #2419

Last update: 2022-07-20

2.25 Percona XtraDB Cluster 5.7.24-31.33

Percona is glad to announce the release of Percona XtraDB Cluster 5.7.24-31.33 on January 4, 2019. Binaries are available from the downloads section or from our software repositories.

Percona XtraDB Cluster 5.7.24-31.33 is now the current release, based on the following:

- Percona Server for MySQL 5.7.24
- Galera Replication library 3.25
- Galera/Codership WSREP API Release 5.7.24

Percona and/or its affiliates © 2023 — Cookie Consent

2.25.1 Deprecated

The following variables are deprecated starting from this release:

- `wsrep_preordered` was used to turn on transparent handling of preordered replication events applied locally first before being replicated to other nodes in the cluster. It is not needed anymore due to the carried out performance fix eliminating the lag in asynchronous replication channel and cluster replication.
- `innodb_disallow_writes` usage to make *InnoDB* avoid writes during was deprecated in favor of the `innodb_read_only` variable.
- `wsrep_drupal_282555_workaround` avoided the duplicate value creation caused by buggy auto-increment logic, but the correspondent bug is already fixed.
- session-level variable `binlog_format=STATEMENT` was enabled only for `pt-table-checksum`, which would be addressed in following releases of the *Percona Toolkit*.

2.25.2 Fixed Bugs

- PXC-2220: Starting two instances of Percona XtraDB Cluster on the same node could cause writing transactions to a page store instead of a galera.cache ring buffer, resulting in huge memory consumption because of retaining already applied write-sets.
- PXC-2230:

`gcs.fc_limit=0` not allowed as dynamic setting to avoid generating flow control on every message was still possible in `my.cnf` due to the inconsistent check.

- PXC-2238: setting `read_only=1` caused race condition.
- PXC-1131: `mysqld-systemd` threw an error at *MySQL* restart in case of non-existing error-log in Centos/RHEL7.
- PXC-2269: being not dynamic, the `pxc_encrypt_cluster_traffic` variable was erroneously allowed to be changed by a `SET GLOBAL` statement.
- PXC-2275: checking `wsrep_node_address` value in the `wsrep_sst_common` command line parser caused parsing the wrong variable.

Last update: 2022-07-20

2.26 Percona XtraDB Cluster 5.7.23-31.31

This release has been superseded by 5.7.23-31.31.2 after a critical regression was found.

Please update to the latest release.

Percona is glad to announce the release of Percona XtraDB Cluster 5.7.23-31.31 on September 26, 2018. Binaries are available from the downloads section or from our software repositories.

Percona XtraDB Cluster 5.7.23-31.31 is now the current release, based on the following:

- Percona Server for MySQL 5.7.23
- Galera Replication library 3.24
- Galera/Codership WSREP API Release 5.7.23

Percona and/or its affiliates © 2023 — Cookie Consent

2.26.1 Deprecated

The following variables are deprecated starting from this release:

- `wsrep_convert_lock_to_trx`

This variable, which defines whether locking sessions should be converted to transactions, is deprecated in Percona XtraDB Cluster 5.7.23-31.31 because it is rarely used in practice.

2.26.2 Fixed Bugs

- PXC-1017: Memcached access to InnoDB was not replicated by Galera
- PXC-2164: The script prevented SELinux from being enabled
- PXC-2155: `wsrep_sst_xtrabackup-v2` did not delete all folders on cleanup
- PXC-2160: In some cases, the MySQL version was not detected correctly with the `Xtrabackup-v2` method of .
- PXC-2199: When the `DROP TRIGGER IF EXISTS` statement was run for a not existing trigger, the node GTID was incremented instead of the cluster GTID.
- PXC-2209: The compression dictionary was not replicated in PXC.
- PXC-2202: In some cases, a disconnected cluster node was not shut down.
- PXC-2165: could fail if either `wsrep_node_address` or `wsrep_sst_receive_address` were not specified.
- PXC-2213: NULL/VOID DDL transactions could commit in a wrong order.

Last update: 2022-07-20

2.27 Percona XtraDB Cluster 5.7.23-31.31.2

To resolve a critical regression, Percona announces the release of Percona XtraDB Cluster 5.7.23-31.31.2 on October 2, 2018. Binaries are available from the downloads section or from our software repositories.

This release resolves a critical regression in the upstream wsrep library and supersedes `5.7.23-31.31`.

Percona XtraDB Cluster 5.7.23-31.31.2 is now the current release, based on the following:

- Percona Server 5.7.23-23
- Galera Replication library 3.24
- Galera/Codership WSREP API Release 5.7.23

All Percona software is open-source and free.

2.27.1 Fixed Bugs

- #2254: A cluster conflict could cause a crash in Percona XtraDB Cluster 5.7.23 if autocommit=off.

Last update: 2022-07-20

Percona and/or its affiliates © 2023 — Cookie Consent

2.28 Percona XtraDB Cluster 5.7.22-29.26

Percona is glad to announce the release of Percona XtraDB Cluster 5.7.22-29.26 on June 29, 2018. Binaries are available from the downloads section or from our software repositories.

Percona XtraDB Cluster 5.7.22-29.26 is now the current release, based on the following:

- Percona Server for MySQL 5.7.22
- Galera Replication library 3.23
- Galera/Codership WSREP API Release 5.7.21

2.28.1 Deprecated

The following variables are deprecated starting from this release:

- `wsrep-force-binlog-format`
- `wsrep_sst_method = mysqldump`

As long as the use of `binlog_format=ROW` is enforced in 5.7, `wsrep_forced_binlog_format` variable is much less significant. The same is related to `mysqldump`, as `xtrabackup` is now the recommended SST method.

2.28.2 New features

- PXC-907: New variable `wsrep_RSU_commit_timeout` allows to configure RSU wait for active commit connection timeout (in microseconds).
- PXC-2111: Percona XtraDB Cluster now supports the `keyring_vault` plugin, which allows to store the master key in a vault server.
- Percona XtraDB Cluster 5.7.22 depends on Percona XtraBackup 2.4.12 in order to fully support vault plugin functionality.

2.28.3 Fixed Bugs

- PXC-2127: Percona XtraDB Cluster shutdown process hung if `thread_handling` option was set to `pool-of-threads` due to a regression in `5.7.21`.
- PXC-2128: Duplicated auto-increment values were set for the concurrent sessions on cluster reconfiguration due to the erroneous readjustment.
- PXC-2059: Error message about the necessity of the `SUPER` privilege appearing in case of the `CREATE TRIGGER` statements fail due to enabled WSREP was made more clear.
- PXC-2061: Wrong values could be read, depending on timing, when read causality was enforced with `wsrep_sync_wait=1`, because of waiting on the commit monitor to be flushed instead of waiting on the apply monitor.
- PXC-2073: `CREATE TABLE AS SELECT` statement was not replicated in case if result set was empty.
- PXC-2087: Cluster was entering the deadlock state if table had an unique key and `INSERT ... ON DUPLICATE KEY UPDATE` statement was executed.
- PXC-2091: Check for the maximum number of rows, that can be replicated as a part of a single transaction because of the Galera limit, was enforced even when replication was disabled with `wsrep_on=OFF`.

Percona and/or its affiliates © 2023 — Cookie Consent

- PXC-2103: Interruption of the local running transaction in a `COMMIT` state by a replicated background transaction while waiting for the binlog backup protection caused the commit fail and, eventually, an assert in Galera.
- PXC-2130: Percona XtraDB Cluster failed to build with Python 3.
- PXC-2142: Replacing Percona Server with Percona XtraDB Cluster on CentOS 7 with the `yum swap` command produced a broken symlink in place of the `/etc/my.cnf` configuration file.
- PXC-2154: rsync SST is now aborted with error message if used on node with `keyring_vault` plugin configured, because it doesn't support `keyring_vault`. Also Percona doesn't recommend using rsync-based SST for data-at-rest encryption with keyring.
- PXB-1544: `xtrabackup --copy-back` didn't read which encryption plugin to use from `plugin-load` setting of the `my.cnf` configuration file.
- PXB-1540: Meeting a zero sized keyring file, *Percona XtraBackup* was removing and immediately recreating it, and this could affect external software noticing the file had undergo some manipulations.

Other bugs fixed: PXC-2072 "flush table for export should be blocked with mode=ENFORCING".

Last update: 2022-07-20

2.29 Percona XtraDB Cluster 5.7.21-29.26

Percona is glad to announce the release of Percona XtraDB Cluster 5.7.21-29.26 on March 02, 2018. Binaries are available from the downloads section or from our software repositories.

Percona XtraDB Cluster 5.7.20-29.24 is now the current release, based on the following:

- Percona Server for MySQL 5.7.21
- Galera Replication library 3.23
- Galera/Codership WSREP API Release 5.7.21

Starting from now, Percona XtraDB Cluster issue tracking system was moved from launchpad to JIRA. All Percona software is open-source and free.

2.29.1 Fixed Bugs

- PXC-2039: Node consistency was compromised for `INSERT INTO ... ON DUPLICATE KEY UPDATE` workload because the regression introduced in Percona XtraDB Cluster `5.7.17-29.20` made it possible to abort local transactions without further re-evaluation in case of a lock conflict.
- PXC-2054 Redo optimized DDL operations (like sorted index build) were not blocked in case of a running backup process, leading to the SST fail. To fix this, `--lock-ddl` option blocks now all DDL during the **xtrabackup** backup stage.
- General code improvement was made in the GTID event handling, when events are captured as a part of the slave replication and appended to the galera replicated write-set. This fixed PXC-2041 (starting async slave on a single node Percona XtraDB Cluster led to a crash) and PXC-2058 (binlog-based master-slave replication broke the cluster) caused by the incorrect handling in the GTID append logic.
- An issue caused by noncoincidence between the order of recovered transaction and the global seqno assigned to the transaction was fixed ensuring that the updated recovery wsrep coordinates are persisted.
- PXC-904: Replication filters were not working with account management statements like `CREATE USER` in case of galera replication; as a result such commands were blocked by the replication filters on async slave nodes but not on galera ones.

Percona and/or its affiliates © 2023 — Cookie Consent

- PXC-2043: SST script was trying to use `pv` (the pipe viewer) for `progress` and `rlimit` options even on nodes with no `pv` installed, resulting in SST fail instead of just ignoring these options for inappropriate nodes.
- PXC-911: When node's own IP address was defined in the `wsrep_cluster_address` variable, the node was receiving "no messages seen in" warnings from it's own IP address in the info log.

This release also contains fixes for the following CVE issues: CVE-2018-2565, CVE-2018-2573, CVE-2018-2576, CVE-2018-2583, CVE-2018-2586, CVE-2018-2590, CVE-2018-2612, CVE-2018-2600, CVE-2018-2622, CVE-2018-2640, CVE-2018-2645, CVE-2018-2646, CVE-2018-2647, CVE-2018-2665, CVE-2018-2667, CVE-2018-2668, CVE-2018-2696, CVE-2018-2703, CVE-2017-3737.

Last update: 2022-07-20

2.30 Percona XtraDB Cluster 5.7.20-29.24

Percona is glad to announce the release of Percona XtraDB Cluster 5.7.20-29.24 on January 26, 2018. Binaries are available from the downloads section or from our software repositories.

Note

Due to new package dependency, Ubuntu/Debian users should use `apt-get dist-upgrade`, `apt upgrade`, or `apt-get install percona-xtradb-cluster-57` to upgrade.

Percona XtraDB Cluster 5.7.20-29.24 is now the current release, based on the following:

- Percona Server 5.7.20-18
- Galera Replication library 3.22
- Galera/Codership WSREP API Release 5.7.20

All Percona software is open-source and free.

2.30.1 NEW FEATURES:

- Ubuntu 17.10 Artful Aardvark is now supported.
- PXC-737: freezing gcache purge was implemented to facilitate node joining through IST, avoiding time consuming SST process.
- PXC-822: a usability improvement was made to timeout error messages, the name of the configuration variable which caused the timeout was added to the message.
- PXC-866: a new variable `wsrep_last_applied`, in addition to `wsrep_last_committed` one, was introduced to clearly separate last committed and last applied transaction numbers.
- PXC-868: on the Joiner, during SST, `tmpdir` variable under `[sst]` section can be used to specify temporary SST files storage different from the default `datadir/.sst` one.

2.30.2 Fixed Bugs

- PXC-889: fixed an issue where a node with an invalid value for `wsrep_provider` was allowed to start up and operate in standalone mode, which could lead to data inconsistency. The node will now abort in this case. Bug fixed #1728774

Percona and/or its affiliates © 2023 — Cookie Consent

- PXC-806: fixed an abort caused by an early read of the `query_id`, ensuring valid ids are assigned to subsequent transactions.
- PXC-850: ensured that a node, because of data inconsistency, isolates itself before leaving the cluster, thus allowing pending nodes to re-evaluate the quorum. Bug fixed #1704404
- PXC-867: `wsrep_sst_rsync` script was overwriting `wsrep_debug` configuration setting making it not to be taken into account.
- PXC-873: fixed formatting issue in the error message appearing when SST is not possible due to a timeout. Bug fixed #1720094
- PXC-874: PXC acting as async slave reported unhandled transaction errors, namely "Rolling back unfinished transaction".
- PXC-875: fixed an issue where toggling `wsrep_provider` off and on failed to reset some internal variables and resulted in PXC logging an "Unsupported protocol downgrade" warning. Bug fixed #1379204
- PXC-877: fixed PXC hang caused by an internal deadlock.
- PXC-878: thread failed to mark exit from the InnoDB server concurrency and therefore never got un-register in InnoDB concurrency system.
- PXC-879: fixed a bug where a `LOAD DATA` command used with GTIDs was executed on one node, but the other nodes would receive less rows than the first one. Bug fixed #1741818
- PXC-880: insert to table without primary key was possible with insertable view if `pxc_strict_mode` variable was set to ENFORCING. Bug fixed #1722493
- PXC-883: fixed `ROLLBACK TO SAVEPOINT` incorrect operation on slaves by avoiding useless wsrep plugin register for a savepoint rollback. Bug fixed #1700593
- PXC-885: fixed IST hang when `keyring_file_data` is set. Bug fixed #1728688
- PXC-887: gcache page files were unnecessarily created due to an error in projecting gcache free size when configured to recover on restart.
- PXC-895: fixed transaction loss after recovery by avoiding interruption of the binlog recovery based on wsrep saved position. Bug fixed 🔧1734113
- PXC-897: fixed empty `gtid_executed` variable after recovering the position of a node with `--wsrep_recover`.
- PXC-906: fixed certification failure in the case of a node restarting at the same time when frequent `TRUNCATE TABLE` commands and DML writes occur simultaneously on other nodes. Bug fixed #1737731
- PXC-909: qpress package was turned into a dependency from suggested/recommended one on Debian 9.
- PXC-903 and PXC-910: init.d/systemctl scripts on Debian 9 were updated to avoid starting wsrep-recover if there was no crash, and to fix an infinite loop at mysqladmin ping fail because of nonexistent ping user.
- PXC-915: suppressing DDL/TOI replication in case of `sql_log_bin` zero value didn't work when DDL statement was modifying an existing table, resulting in an error.

Last update: 2022-07-20

2.31 Percona XtraDB Cluster 5.7.19-29.22

Percona is glad to announce the release of Percona XtraDB Cluster 5.7.19-29.22 on September 22, 2017. Binaries are available from the downloads section or from our software repositories.

Percona and/or its affiliates © 2023 — Cookie Consent

Percona XtraDB Cluster 5.7.19-29.22 is now the current release, based on the following:

- Percona Server 5.7.19-17
- Galera Replication library 3.22
- wsrep API version 29

All Percona software is open-source and free.

2.31.1 Upgrade Instructions

After you upgrade each node to Percona XtraDB Cluster 5.7.19-29.22, run the following command on one of the nodes:

```
$ mysql -uroot -p < /usr/share/mysql/pxc_cluster_view.sql
```

Then restart all nodes, one at a time:

```
$ sudo service mysql restart
```

2.31.2 New Features

- Introduced the `pxc_cluster_view` table to get a unified view of the cluster. This table is exposed through the performance schema.

```
mysql> select * from pxc_cluster_view;
------------------------------------------------------------------------------
HOST_NAME  UUID                                  STATUS  LOCAL_INDEX  SEGMENT
------------------------------------------------------------------------------
n1         b25bfd59-93ad-11e7-99c7-7b26c63037a2  DONOR   0            0
n2         be7eae92-93ad-11e7-88d8-92f8234d6ce2  JOINER  1            0
------------------------------------------------------------------------------
2 rows in set (0.01 sec)
```

- PXC-803: Added support for new features in Percona XtraBackup 2.4.7:
  - `wsrep_debug` enables debug logging
  - `encrypt_threads` specifies the number of threads that XtraBackup should use for encrypting data (when `encrypt=1`). This value is passed using the `--encrypt-threads` option in XtraBackup.
  - `backup_threads` specifies the number of threads that XtraBackup should use to create backups. See the `--parallel` option in XtraBackup.

2.31.3 Improvements

- PXC-835: Limited `wsrep_node_name` to 64 bytes.
- PXC-846: Improved logging to report reason of IST failure.
- PXC-851: Added version compatibility check during SST with XtraBackup:
  - If donor is 5.6 and joiner is 5.7: A warning is printed to perform `mysql_upgrade`.
  - If donor is 5.7 and joiner is 5.6: An error is printed and SST is rejected.

Percona and/or its affiliates © 2023 — Cookie Consent

2.31.4 Fixed Bugs

- PXC-825: Fixed script for SST with XtraBackup (`wsrep_sst_xtrabackup-v2`) to include the `--defaults-group-suffix` when logging to syslog. For more information, see #1559498.
- PXC-826: Fixed multi-source replication to PXC node slave. For more information, see #1676464.
- PXC-827: Fixed handling of different binlog names between donor and joiner nodes when GTID is enabled. For more information, see #1690398.
- PXC-830: Rejected the `RESET MASTER` operation when wsrep provider is enabled and `gtid_mode` is set to `ON`. For more information, see #1249284.
- PXC-833: Fixed connection failure handling during SST by making the donor retry connection to joiner every second for a maximum of 30 retries. For more information, see #1696273.
- PXC-839: Fixed GTID inconsistency when setting `gtid_next`.
- PXC-840: Fixed typo in alias for `systemd` configuration.
- PXC-841: Added check to avoid replication of DDL if `sql_log_bin` is disabled. For more information, see #1706820.
- PXC-842: Fixed deadlocks during Load Data Infile (LDI) with `log-bin` disabled by ensuring that a new transaction (of 10 000 rows) starts only after the previous one is committed by both wsrep and InnoDB. For more information, see #1706514.
- PXC-843: Fixed situation where the joiner hangs after SST has failed by dropping all transactions in the receive queue. For more information, see #1707633.
- PXC-853: Fixed cluster recovery by enabling `wsrep_ready` whenever nodes become PRIMARY.
- PXC-862: Fixed script for SST with XtraBackup (`wsrep_sst_xtrabackup-v2`) to use the `ssl-dhparams` value from the configuration file.

!!! note

```
As part of fix for [PXC-827](https://jira.percona.com/browse/PXC-827),
version communication was added to the SST protocol.
As a result, newer version of PXC (as of 5.7.19 and later)
cannot act as donor when joining an older version PXC node (prior to 5.7.19).
It will work fine vice versa:
old node can act as donor when joining nodes with new version.
```

Last update: 2022-07-20

2.32 Percona XtraDB Cluster 5.7.19-29.22-3

Percona is glad to announce the release of Percona XtraDB Cluster 5.7.19-29.22-3 on October 27, 2017. Binaries are available from the downloads section or from our software repositories.

Percona XtraDB Cluster 5.7.19-29.22-3 is now the current release, based on the following:

- Percona Server 5.7.19-17
- Galera Replication library 3.22
- wsrep API version 29

All Percona software is open-source and free.

Percona and/or its affiliates © 2023 — Cookie Consent

2.32.1 Fixed Bugs

- Added access checks for DDL commands to make sure they do not get replicated if they failed without proper permissions. Previously, when a user tried to perform certain DDL actions that failed locally due to lack of privileges, the command could still be replicated to other nodes, because access checks were performed after replication.

This vulnerability is identified as CVE-2017-15365.

Last update: 2022-07-20

2.33 Percona XtraDB Cluster 5.7.18-29.20

Percona is glad to announce the release of *Percona XtraDB Cluster* 5.7.18-29.20 on June 2, 2017. Binaries are available from the downloads section or from our software repositories.

Note

Due to new package dependency, Ubuntu/Debian users should use `apt-get dist-upgrade` or `apt-get install percona-xtradb-cluster-57` to upgrade.

Percona XtraDB Cluster 5.7.18-29.20 is now the current release, based on the following:

- Percona Server 5.7.18-15
- Galera Replication library 3.20
- wsrep API version 29

All Percona software is open-source and free.

2.33.1 Fixed Bugs

- PXC-749: Fixed memory leak when running `INSERT` on a table without primary key defined and `wsrep_certify_nonPK` disabled (set to `0`).

!!! note

```
  It is recommended to have primary keys defined on all tables for correct write set
replication.
```

- PXC-812: Fixed SST script to leave the DONOR keyring when JOINER clears the datadir.
- PXC-813: Fixed SST script to use UTC time format.
- PXC-816: Fixed hook for caching GTID events in asynchronous replication. For more information, see #1681831.
- PXC-820: Enabled querying of `pxc_maint_mode` by another client during the transition period.
- PXC-823: Fixed SST flow to gracefully shut down JOINER node if SST fails because DONOR leaves the cluster due to network failure. This ensures that the DONOR is then able to recover to synced state when network connectivity is restored For more information, see #1684810.
- PXC-824: Fixed graceful shutdown of Percona XtraDB Cluster node to wait until applier thread finishes.

Percona LLC and/or its affiliates © 2023 — Cookie Consent

2.33.2 Other Improvements

- PXC-819: Added five new status variables to expose required values from `wsrep_ist_receive_status` and `wsrep_flow_control_interval` as numbers, rather than strings that need to be parsed:
  - `wsrep_flow_control_interval_low`
  - `wsrep_flow_control_interval_high`
  - `wsrep_ist_receive_seqno_start`
  - `wsrep_ist_receive_seqno_current`
  - `wsrep_ist_receive_seqno_end`

Last update: 2022-07-20

2.34 Percona XtraDB Cluster 5.7.17-29.20

Percona is glad to announce the release of *Percona XtraDB Cluster* 5.7.17-29.20 on April 19, 2017. Binaries are available from the downloads section or from our software repositories.

Percona XtraDB Cluster 5.7.17-29.20 is now the current release, based on the following:

- Percona Server 5.7.17-13
- Galera Replication library 3.20
- wsrep API version 29

All Percona software is open-source and free.

2.34.1 Performance Improvements

This release was focused on performance and scaling capability with increasing workload threads. Tests show up to 10 times increase in performance.

2.34.2 Fixed Bugs

- Improved parallelism for better scaling with multiple threads.
- Updated semantics for gcache page cleanup to trigger when either `gcache.keep_pages_size` or `gcache.keep_pages_count` exceeds the limit, instead of both at the same time.
- Improved SST and IST log messages for better readability and unification.
- Excluded the `garbd` node from flow control calculations.
- Added extra checks to verify that SSL files (certificate, certificate authority, and key) are compatible before opening connection.
- Added validations for `DISCARD TABLESPACE` and `IMPORT TABLESPACE` in PXC Strict Mode to prevent data inconsistency.
- Added support for passing the XtraBackup buffer pool size with the `use-memory` option under `[xtrabackup]` and the `innodb_buffer_pool_size` option under `[mysqld]` when the `--use-memory` option is not passed with the `inno-apply-opts` option under `[sst]`.
- Added the `wsrep_flow_control_status` variable to indicate if node is in flow control (paused).
- Fixed gcache page cleanup not triggering when limits are exceeded.
- PXC-766: Added the `wsrep_ist_receive_status` variable to show progress during an IST.

Percona and/or its affiliates © 2023 — Cookie Consent

- Allowed `CREATE TABLE ... AS SELECT` (CTAS) statements with temporary tables (`CREATE TEMPORARY TABLE ... AS SELECT`) in PXC Strict Mode. For more information, see #1666899.
- PXC-782: Updated `xtrabackup-v2` script to use the `tmpdir` option (if it is set under `[sst]`, `[xtrabackup]` or `[mysqld]`, in that order).
- PXC-783: Improved the wsrep stage framework.
- PXC-784: Fixed the `pc.recovery` procedure to abort if the `gvwstate.dat` file is empty or invalid, and fall back to normal joining process. For more information, see #1669333.
- PXC-794: Updated the `sockopt` option to include a comma at the beginning if it is not set by the user.
- PXC-795: Set `--parallel=4` as default option for `wsrep_sst_xtrabackup-v2` to run four threads with XtraBackup.
- PXC-797: Blocked `wsrep_desync` toggling while node is paused to avoid halting the cluster when running `FLUSH TABLES WITH READ LOCK`. For more information, see #1370532.
- PXC-805: Inherited upstream fix to avoid using deprecated variables, such as `INFORMATION_SCHEMA.SESSION_VARIABLE`. For more information, see #1676401.
- PXC-811: Changed default values for the following variables:
  - `fc_limit` from `16` to `100`
  - `send_window` from `4` to `10`
  - `user_send_window` from `2` to `4`
- Moved wsrep settings into a separate configuration file (`/etc/my.cnf.d/wsrep.cnf`).
- Fixed `mysqladmin shutdown` to correctly stop the server on systems using `systemd`.
- Fixed several minor packaging and dependency issues.

Last update: 2022-07-20

2.35 Percona XtraDB Cluster 5.7.17-27.20

Percona is glad to announce the release of *Percona XtraDB Cluster* 5.7.17-27.20 on March 16, 2017. Binaries are available from the downloads section or from our software repositories.

Percona XtraDB Cluster 5.7.17-27.20 is now the current release, based on the following:

- Percona Server 5.7.17-11
- Galera Replication library 3.20
- wsrep API version 27

All Percona software is open-source and free. Details of this release can be found in the 5.7.17-27.20 milestone on Launchpad.

2.35.1 Fixed Bugs

- BLD-512: Fixed startup of `garbd` on Ubuntu 16.04.2 LTS (Xenial Xerus).
- BLD-519: Added the `garbd` debug package to the repository.
- BLD-569: Fixed `grabd` script to return non-zero if it fails to start.
- BLD-570: Fixed service script for `garbd` on Ubuntu 16.04.2 LTS (Xenial Xerus) and Ubuntu 16.10 (Yakkety Yak).
- BLD-593: Limited the use of `rm` and `chown` by `mysqld_safe` to avoid exploits of the CVE-2016-5617 vulnerability. For more information, see #1660265.

Percona and/or its affiliates © 2023 — Cookie Consent

Credit to Dawid Golunski (https://legalhackers.com).

- BLD-610: Added version number to the dependency requirements of the full RPM package.
- BLD-643: Fixed `systemctl` to mark `mysql` process as inactive after it fails to start and not attempt to start it again. For more information, see #1662292.
- BLD-644: Added the `which` package to Percona XtraDB Cluster dependencies on CentOS 7. For more information, see #1661398.
- BLD-645: Fixed `mysqld_safe` to support options with a forward slash (`/`). For more information, see #1652838.
- BLD-647: Fixed `systemctl` to show correct status for `mysql` on CentOS 7. For more information, see #1644382.

Last update: 2022-07-20

2.36 Percona XtraDB Cluster 5.7.16-27.19

Percona is glad to announce the release of *Percona XtraDB Cluster* 5.7.16-27.19 on December 15, 2016. Binaries are available from the downloads section or from our software repositories.

Percona XtraDB Cluster 5.7.16-27.19 is now the current release, based on the following:

- Percona Server 5.7.16-10
- Galera Replication library 3.19
- wsrep API version 27

All Percona software is open-source and free. Details of this release can be found in the 5.7.16-27.19 milestone on Launchpad.

2.36.1 Deprecated

- The following encryption modes:
  - `encrypt=1`
  - `encrypt=2`
  - `encrypt=3`

The default is `encrypt=0` with encryption disabled. The recommended mode now is the new `encrypt=4`, which uses SSL files generated by MySQL.

For more information, see Encrypting PXC Traffic.

2.36.2 New Features

- Added `encrypt=4` mode for SST encryption that uses SSL files generated by MySQL. Modes `1`, `2`, and `3` will soon be deprecated.

Percona and/or its affiliates © 2023 — Cookie Consent

- ProxySQL assisted maintenance mode that enables you to take a node down without adjusting ProxySQL manually. The mode is controlled using the `pxc_maint_mode` variable, which can be set to one of the following values:
  - `DISABLED`: This is the default state that tells ProxySQL to route traffic to the node as usual.
  - `SHUTDOWN`: This state is set automatically when you initiate node shutdown.
  - `MAINTENANCE`: You can change to this state if you need to perform maintenace on a node without shutting it down.

For more information, see Assisted Maintenance Mode.

- Simplified SSL configuration for Galera/SST traffic with `pxc-encrypt-cluster-traffic` option, which auto-configures SSL encryption.

For more information, see SSL Automatic Configuration.

- Added the `wsrep_flow_control_interval` status variable that displays the lower and upper limits of the flow control system used for the Galera receive queue.

2.36.3 Fixed Bugs

- Optimized IST donor selection logic to avoid SST. Child processes are now cleaned-up and node state is resumed if SST fails.
- Added `init.ok` to the list of files that do not get removed during SST.
- Fixed error with ASIO library not acknowledging an `EPOLLIN` event when building Galera.
- Fixed stalling of DML workload on slave node caused by `FLUSH TABLE` executed on master.

For more information, see #1629296.

- Fixed `super_read_only` to not apply to Galera replication applier.

For more information, see #1634295.

- Redirected `netcat` output to `stdout` to avoid it in the log.

For more information, see #1625968.

- Enabled replication of `ALTER USER` statements.

For more information, see #1376269.

- Changed the `wsrep_max_ws_rows` variable to ignore non-replicated write-sets generated by DML action on temporary tables (explict or implicit).

For more information, see #1638138.

- Fixed SST to fail with an error if SSL is not supported by `socat`, instead of switching to unencrypted mode.
- Fixed SST with SSL to auto-generate a 2048-bit `dhparams` file for versions of `socat` before 1.7.3. These older versions use 512-bit `dhparams` file by default that get rejected by newer clients with `dh key too small` error.
- PXC-731: Changed the `wsrep_cluster_name` variable to read-only, because changing it dynamically leads to high overhead.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

For more information, see #1620439.

- PXC-732: Improved error message when any of the SSL files required for SST are missing.
- PXC-735: Fixed SST to fail with an error when `netcat` is used (`transferfmt` set to `nc`) with SSL encryption (`encrypt` set to `2`, `3` or `4`), instead of silently switching to unencrypted mode.
- Fixed faulty switch case that caused cluster to stall when the `repl.commit_order` variable was set to `2` (`LOCAL_OOOC` mode that should allow out-of-order committing for local transactions).

Last update: 2022-07-20

2.37 Percona XtraDB Cluster 5.7.14-26.17

> **Note**
>
> This release is dedicated to the memory of Federico Goncalvez, our colleague with Percona's Uruguayan team until his tragic death on September 6, 2016.
>
> Fede, you are missed.

Percona is glad to announce the release of Percona XtraDB Cluster 5.7.14-26.17 on September 29, 2016. Binaries are available from the downloads area or from our software repositories.

Percona XtraDB Cluster 5.7.14-26.17 is the first GA release in the 5.7 series, based on the following:

- Percona Server 5.7.14-8
- Galera Replicator 3.17

For information about the changes and new features introduced in Percona Server 5.7, see Changed in Percona Server 5.7.

2.37.1 New Features

This is a list of the most important features introduced in Percona XtraDB Cluster 5.7 compared to version 5.6

- PXC Strict Mode saves your workload from experimental and unsupported features.
- Support for monitoring Galera Library instruments and other wsrep instruments as part of Performance Schema.
- Support for encrypted tablespaces in Multi-Master Topology, which enables Percona XtraDB Cluster to wire encrypted tablespace to new booting node.
- Compatibility with ProxySQL, including a quick configuration script.
- Support for monitoring Percona XtraDB Cluster nodes using Percona Monitoring and Management
- More stable and robust operation with MySQL and Percona Server version 5.7.14, as well as Galera 3.17 compatibility. Includes all upstream bug fixes, improved logging and more.
- Simplified packaging for Percona XtraDB Cluster to a single package that installs everything it needs, including the Galera library.
- Support for latest XtraBackup with enhanced security checks.

Percona and/or its affiliates © 2023 — Cookie Consent

2.37.2 Bug Fixes

- Fixed crash when a local transaction (such as `EXPLAIN` or `SHOW`) is interrupted by a replicated transaction with higher priopiry (like `ALTER` that changes table structure and can thus affect the result of the local transaction).
- Fixed DONOR node getting stuck in `Joined` state after successful SST.
- Fixed error message when altering non-existent table with `pxc-strict-mode` enabled.
- Fixed path to directory in `percona-xtradb-cluster-shared.conf`.
- Fixed setting of `seqno` in `grastate.dat` to `-1` on clean shutdown.
- Fixed failure of asynchronous TOI actions (like `DROP`) for non-primary nodes.
- Fixed replacing of `my.cnf` during upgrade from 5.6 to 5.7.

2.37.3 Security Fixes

- CVE-2016-6662
- CVE-2016-6663
- CVE-2016-6664

For more information, see https://www.percona.com/blog/2016/09/12/database-affected-cve-2016-6662/

2.37.4 Other Improvements

Added support of `defaults-group-suffix` for SST scripts.

Last update: 2022-08-24

2.38 Percona XtraDB Cluster 5.7.12-5rc1-26.16

Percona is glad to announce the release of Percona XtraDB Cluster 5.7.12-5rc1-26.16 on August 9, 2016. Binaries are available from the downloads area or from our software repositories.

Percona XtraDB Cluster 5.7.12-5rc1-26.16 is based on the following:

- Percona Server 5.7.12
- Galera Replicator 3.16

2.38.1 New Features

- **PXC Strict Mode**: Use the `pxc_strict_mode` variable in the configuration file or the `--pxc-strict-mode` option during `mysqld` startup. For more information, see PXC Strict Mode.
- **Galera instruments exposed in Performance Schema**: This includes mutexes, condition variables, file instances, and threads.

2.38.2 Bug Fixes

- Fixed error messages.
- Fixed the failure of SST via `mysqldump` with `gtid_mode=ON`.

Percona and/or its affiliates © 2023 — Cookie Consent

- Added check for TOI that ensures node readiness to process DDL+DML before starting the execution.
- Removed protection against repeated calls of `wsrep->pause()` on the same node to allow parallel RSU operation.
- Changed `wsrep_row_upd_check_foreign_constraints` to ensure that `fk-reference-table` is open before marking it open.
- Fixed error when running `SHOW STATUS` during group state update.
- Corrected the return code of `sst_flush_tables()` function to return a non-negative error code and thus pass assertion.
- Fixed memory leak and stale pointer due to stats not freeing when toggling the `wsrep_provider` variable.
- Fixed failure of `ROLLBACK` to register `wsrep_handler`
- Fixed failure of symmetric encryption during SST.

2.38.3 Other Changes

- Added support for sending the keyring when performing encrypted SST. For more information, see Encrypting PXC Traffic.
- Changed the code of `THD_PROC_INFO` to reflect what the thread is currently doing.
- Using XtraBackup as the SST method now requires Percona XtraBackup 2.4.4 or later.
- Improved rollback process to ensure that when a transaction is rolled back, any statements open by the transaction are also rolled back.
- Removed the `sst_special_dirs` variable.
- Disabled switching of `slave_preserve_commit_order` to `ON` when running PXC in cluster mode, as it conflicts with existing multi-master commit ordering resolution algorithm in Galera.
- Changed the default `my.cnf` configuration.
- Other low-level fixes and improvements for better stability.

Last update: 2022-08-24

2.39 Percona XtraDB Cluster 5.7.11-4beta-25.14.2

Percona is glad to announce the release of Percona XtraDB Cluster 5.7.11-4beta-25.14.2 on June 9, 2016. Binaries are available from downloads area or from our software repositories.

Note

This release is available only from the testing repository. It is not meant for upgrade from Percona XtraDB Cluster 5.6 and earlier versions. Only fresh installation is supported.

Percona XtraDB Cluster 5.7.11-4beta-25.14.2 is based on the following:

- Percona Server 5.7.11-4
- Galera Replicator 3.14.2

Percona and/or its affiliates © 2023 — Cookie Consent

This is the first beta release in the Percona XtraDB Cluster 5.7 series. It includes all changes from upstream releases and the following changes:

- Percona XtraDB Cluster 5.7 does not include `wsrep_sst_xtrabackup`. It has been replace by `wsrep_sst_xtrabackup_v2`.
- The `wsrep_mysql_replication_bundle` variable has been removed.

Last update: 2022-08-24

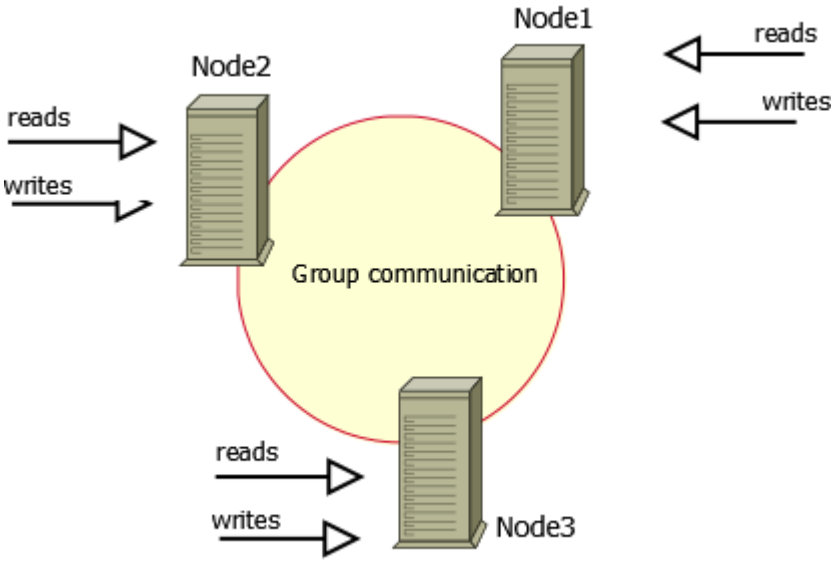
Percona and/or its affiliates © 2023 — Cookie Consent

3. Introduction

3.1 About Percona XtraDB Cluster

Percona XtraDB Cluster is a fully open-source high-availability solution for MySQL. It integrates Percona Server and Percona XtraBackup with the Galera library to enable synchronous multi-source replication.

A *cluster* consists of *nodes*, where each node contains the same set of data synchronized accross nodes. The recommended configuration is to have at least 3 nodes, but you can have 2 nodes as well. Each node is a regular MySQL Server instance (for example, Percona Server). You can convert an existing MySQL Server instance to a node and run the cluster using this node as a base. You can also detach any node from the cluster and use it as a regular MySQL Server instance.



Benefits:

- When you execute a query, it is executed locally on the node. All data is available locally, no need for remote access.
- No central management. You can loose any node at any point of time, and the cluster will continue to function without any data loss.
- Good solution for scaling a read workload. You can put read queries to any of the nodes.

Drawbacks:

- Overhead of provisioning new node. When you add a new node, it has to copy the full data set from one of existing nodes. If it is 100GB, it copies 100GB.
- This can't be used as an effective write scaling solution. There might be some improvements in write throughput when you run write traffic to 2 nodes versus all traffic to 1 node, but you can't expect a lot. All writes still have to go on all nodes.
- You have several duplicates of the data, for 3 nodes you have 3 duplicates.

Percona and/or its affiliates © 2023 — Cookie Consent

3.1.1 Components

Percona XtraDB Cluster is based on Percona Server running with the XtraDB storage engine. It uses the Galera library, which is an implementation of the write set replication (wsrep) API developed by Codership Oy. The default and recommended data transfer method is via Percona XtraBackup.

Last update: 2022-08-23

3.2 Percona XtraDB Cluster Limitations

The following limitations apply to Percona XtraDB Cluster:

- Replication works only with *InnoDB* storage engine. Any writes to tables of other types, including system (`mysql.\*`) tables, are not replicated. However, `DDL` statements are replicated in the statement level, and changes to `mysql.\*` tables are replicated that way. So you can safely issue `CREATE USER...`, but issuing `INSERT INTO mysql.user...` will not be replicated. You can enable experimental *MyISAM* replication support using the `wsrep_replicate_myisam` variable.
- Unsupported queries:
  - `LOCK TABLES` and `UNLOCK TABLES` is not supported in multi-source setups
  - Lock functions, such as `GET_LOCK()`, `RELEASE_LOCK()`, and so on

> **See also**
>
> *MySQL* Documentation:
>
> - LOCK TABLES AND UNLOCK TABLES statements
> - Locking functions

  - Query log cannot be directed to table.

If you enable query logging, you must forward the log to a file:

```
log_output = FILE
```

Use `general_log` and `general_log_file` to choose query logging and the log file name.

- Maximum allowed transaction size is defined by the `wsrep_max_ws_rows` and `wsrep_max_ws_size` variables. `LOAD DATA INFILE` processing will commit every 10,000 rows. So large transactions due to `LOAD DATA` will be split to series of small transactions.
- Due to cluster-level optimistic concurrency control, a transaction issuing a `COMMIT` may still be aborted at that stage. There can be two transactions writing to the same rows and committing in separate Percona XtraDB Cluster nodes, and only one of the them can successfully commit. The failing one will be aborted. For cluster-level aborts, Percona XtraDB Cluster gives back deadlock error code:

Percona and/or its affiliates © 2023 — Cookie Consent

```
(Error: 1213 SQLSTATE: 40001  (ER_LOCK_DEADLOCK)).
```

- XA transactions are not supported due to possible rollback on commit.
- The write throughput of the whole cluster is limited by the weakest node. If one node becomes slow, the whole cluster slows down. If you have requirements for stable high performance, then it should be supported by corresponding hardware.
- The minimal recommended size of cluster is three nodes. The third node can be an arbitrator.
- InnoDB fake changes feature is not supported. This feature has been removed.
- `enforce_storage_engine=InnoDB` is not compatible with `wsrep_replicate_myisam=OFF` (default).

See also

Percona Server for MySQL documentation: enforcing storage engine

- When running Percona XtraDB Cluster in cluster mode, avoid `ALTER TABLE ... IMPORT/EXPORT` workloads. It can lead to node inconsistency if not executed in sync on all nodes.
- All tables must have the primary key. This ensures that the same rows appear in the same order on different nodes. The `DELETE` statement is not supported on tables without a primary key.
- Percona Server 5.7 data at rest encryption is similar to the MySQL 5.7 data-at-rest encryption. Review the available encryption features for Percona Server for MySQL 5.7. Percona Server 8.0 provides more encryption features and options which are not available in this version.

See also

Galera Documentation: Tables without Primary Keys

- Avoid reusing the names of a persistent table for a temporary table. Although MySQL allows a temporary table and a persistent table to have the same name, this approach is not recommended. If a persistent table name matches a temporary table name, Galera Cluster blocks the replication to that table.

With wsrep_debug set to *1*, the error log may contain the following message:

```
... [Note] WSREP: TO BEGIN: -1, 0 : create table t (i int) engine=innodb
... [Note] WSREP: TO isolation skipped for: 1, sql: create table t (i int)
engine=innodb.Only temporary tables affected.
```

- As of version 5.7.32-13.47, an INPLACE ALTER TABLE query takes an internal shared lock on the table during the execution of the query. The `LOCK=NONE` clause is no longer allowed for all of the INPLACE ALTER TABLE queries due to this change.

This change addresses a deadlock, which could cause a cluster node to hang in the following scenario:

- An INPLACE `ALTER TABLE` query in one session or being applied as Total Order Isolation (TOI)
- A DML on the same table from another session

Do not use one or more dot characters (.) when defining the values for the following variables:

- log_bin
- log_bin_index

Percona and/or its affiliates © 2023 — Cookie Consent

MySQL and **XtraBackup** handles the value in different ways and this difference causes unpredictable behavior.

Last update: 2022-08-23

3.3 Percona XtraDB Cluster 5.7 version numbers

A version number identifies the product release. The product contains the latest Generally Available (GA) features at the time of that release. Percona XtraDB Cluster assigns a set of unique numbers for each version in increasing order. The numbers are a combination of the Percona Server for MySQL version and internal version numbers denoting the build.

Percona uses semantic version numbering, which follows the pattern of base version, minor build, and an optional custom build. Percona assigns unique, non-negative integers in increasing order for each minor build release. The version number combines the base Percona Server for MySQL version number, the minor build version, and the custom build version, if needed.

The version numbers for Percona XtraDB Cluster 5.7.40-31.63 define the following information:

- Base version - the leftmost set of numbers indicate the Percona Server for MySQL version used as a base. An increase in base version resets the minor build version and the custom build version to 0.
- WSREP API version - the version of the WSREP API
- Minor build version - an internal number that denotes which version. When this number increases by one, the custom build fix is reset to 0.
- Custom build version - an optional number assigned to custom builds used for bug fixes. The software features, unless they're included in the bug fix, don't change.

Last update: 2022-12-29

3.4 Telemetry on Percona XtraDB Cluster

Percona telemetry fills in the gaps in our understanding of how you use the Percona XtraDB Cluster to improve our products. Participation in the anonymous program is optional. You can opt out if you prefer to not share this information.

3.4.1 What information is collected

At this time, telemetry is added only to the Percona packages and Docker images. Percona XtraDB Cluster collects only information about the installation environment. Future releases may add additional metrics.

Be assured that access to this raw data is rigorously controlled. Percona does not collect personal data. All data is anonymous and cannot be traced to a specific user. To learn more about our privacy practices, read our Percona Privacy statement.

An example of the data collected is the following:

```
[{"id" : "c416c3ee-48cd-471c-9733-37c2886f8231",
"product_family" : "PRODUCT_FAMILY_PXC",
```


Percona and/or its affiliates © 2023 — Cookie Consent

```
"instanceId" : "6aef422e-56a7-4530-af9d-94cc02198343",
"createTime" : "2023-10-16T10:46:23Z",
"metrics":
[{"key" : "deployment","value" : "PACKAGE"},
{"key" : "pillar_version","value" : "5.7.44-31.65"},
{"key" : "OS","value" : "Oracle Linux Server 8.8"},
{"key" : "hardware_arch","value" : "x86_64 x86_64"}]}]
```

3.4.2 Disable telemetry

Starting with Percona XtraDB Cluster 5.7.44-31.65, telemetry is enabled by default. If you decide not to send usage data to Percona, you can set the `PERCONA_TELEMETRY_DISABLE=1` environment variable for either the root user or the operating system before the installation process.

Debian-derived distribution **Red Hat-derived distribution** **DOCKER**

Add the environment variable before the installation process.

```
$ sudo PERCONA_TELEMETRY_DISABLE=1 apt install percona-xtradb-cluster
```

Add the environment variable before the install process.

```
$ sudo PERCONA_TELEMETRY_DISABLE=1 yum install percona-xtradb-cluster
```

Add the environment variable when running a command in a new container.

```
$ docker run -d -e MYSQL_ROOT_PASSWORD=test1234# -e PERCONA_TELEMETRY_DISABLE=1 -e
CLUSTER_NAME=pxc-cluster1 --name=pxc-node1 percona/percona-xtradb-cluster:5.7
```

Last update: 2023-12-07

Percona and/or its affiliates © 2023 — Cookie Consent

4. Getting Started

4.1 Quick Start Guide for Percona XtraDB Cluster

This guide describes the procedure for setting up Percona XtraDB Cluster.

Examples provided in this guide assume there are three Percona XtraDB Cluster nodes, as a common choice for trying out and testing:

Node	Host	IP
Node 1	pxc1	192.168.70.61
Node 2	pxc2	192.168.70.62
Node 3	pxc3	192.168.70.63

Note

Avoid creating a cluster with two or any even number of nodes, because this can lead to *split brain*.

For more information, see Cluster Failover.

The following procedure provides an overview with links to details for every step:

- Install Percona XtraDB Cluster on all nodes and set up root access for them.

It is recommended to install from official Percona repositories:

- On Red Hat and CentOS, install using YUM.
- On Debian and Ubuntu, install using APT.
- Configure all nodes with relevant settings required for write-set replication.

This includes path to the Galera library, location of other nodes, etc.

- Bootstrap the first node to initialize the cluster.

This must be the node with your main database, which will be used as the data source for the cluster.

- Add other nodes to the cluster.

Data on new nodes joining the cluster is overwritten in order to synchronize it with the cluster.

- Verify replication.

Although cluster initialization and node provisioning is performed automatically, it is a good idea to ensure that changes on one node actually replicate to other nodes.

- Install ProxySQL.

To complete the deployment of the cluster, a high-availability proxy is required. We recommend installing ProxySQL on client nodes for efficient workload management across the cluster without any changes to the applications that generate queries.

Percona and/or its affiliates © 2023 — Cookie Consent

4.1.1 Percona Monitoring and Management

Percona Monitoring and Management is the best choice for managing and monitoring Percona XtraDB Cluster performance. It provides visibility for the cluster and enables efficient troubleshooting.

Last update: 2022-07-20

4.2 Installing Percona XtraDB Cluster

Install Percona XtraDB Cluster on all hosts that you are planning to use as cluster nodes and ensure that you have root access to the MySQL server on each one.

It is recommended to install Percona XtraDB Cluster from official Percona software repositories using the corresponding package manager for your system:

- Debian or Ubuntu
- Red Hat or CentOS

4.2.1 Installation Alternatives

Percona also provides a generic tarball with all required files and binaries for manual installation:

- Installing Percona XtraDB Cluster from Binary Tarball

If you want to build Percona XtraDB Cluster from source, see Compiling and Installing from Source Code.

If you want to run Percona XtraDB Cluster using Docker, see Running Percona XtraDB Cluster in a Docker Container.

Last update: 2022-07-20

4.3 Configuring Nodes for Write-Set Replication

After installing Percona XtraDB Cluster on a node, configure it with information about the cluster.

Note

Make sure that the Percona XtraDB Cluster server is not running.

```
$ sudo service mysql stop
```

Configuration examples assume there are three Percona XtraDB Cluster nodes:

Node	Host	IP
Node 1	pxc1	192.168.70.61
Node 2	pxc2	192.168.70.62
Node 3	pxc3	192.168.70.63

Percona and/or its affiliates © 2023 — Cookie Consent

If you are running Debian or Ubuntu, add the following configuration variables to */etc/percona-xtradb-cluster.conf.d/wsrep.cnf* on the first node:

```
[mysqld]
wsrep_provider=/usr/lib/libgalera_smm.so
wsrep_cluster_name=pxc-cluster
wsrep_cluster_address=gcomm://192.168.70.61,192.168.70.62,192.168.70.63
wsrep_node_name=pxc1
wsrep_node_address=192.168.70.61
wsrep_sst_method=xtrabackup-v2
wsrep_sst_auth=sstuser:passw0rd
pxc_strict_mode=ENFORCING
binlog_format=ROW
default_storage_engine=InnoDB
innodb_autoinc_lock_mode=2
```

If you are running Red Hat or CentOS, add the following configuration variables to */etc/percona-xtradb-cluster.conf.d/wsrep.cnf* on the first node:

```
[mysqld]
 wsrep_provider=/usr/lib64/galera3/libgalera_smm.so
 wsrep_cluster_name=pxc-cluster
 wsrep_cluster_address=gcomm://192.168.70.61,192.168.70.62,192.168.70.63
 wsrep_node_name=pxc1
 wsrep_node_address=192.168.70.61
 wsrep_sst_method=xtrabackup-v2
 wsrep_sst_auth=sstuser:passw0rd
 pxc_strict_mode=ENFORCING
 binlog_format=ROW
 default_storage_engine=InnoDB
 innodb_autoinc_lock_mode=2
```

Use the same configuration for the second and third nodes, except the `wsrep_node_name` and `wsrep_node_address` variables:

- For the second node:

```
wsrep_node_name=pxc2
wsrep_node_address=192.168.70.62
```

- For the third node:

```
wsrep_node_name=pxc3
wsrep_node_address=192.168.70.63
```

4.3.1 Configuration Reference

`wsrep_provider`

Specify the path to the Galera library.

Percona and/or its affiliates © 2023 — Cookie Consent

Note

The location depends on the distribution:

- Debian or Ubuntu: `/usr/lib/libgalera_smm.so`
- Red Hat or CentOS: `/usr/lib64/galera3/libgalera_smm.so`

`wsrep_cluster_name`

Specify the logical name for your cluster. It must be the same for all nodes in your cluster.

`wsrep_cluster_address`

Specify the IP addresses of nodes in your cluster. At least one is required for a node to join the cluster, but it is recommended to list addresses of all nodes. This way if the first node in the list is not available, the joining node can use other addresses.

Note

No addresses are required for the initial node in the cluster. However, it is recommended to specify them and properly bootstrap the first node. This will ensure that the node is able to rejoin the cluster if it goes down in the future.

`wsrep_node_name`

Specify the logical name for each individual node. If this variable is not specified, the host name will be used.

`wsrep_node_address`

Specify the IP address of this particular node.

`wsrep_sst_method`

By default, Percona XtraDB Cluster uses Percona XtraBackup for *State Snapshot Transfer* (SST). Setting `wsrep_sst_method=xtrabackup-v2` is highly recommended. This method requires a user for SST to be set up on the initial node. Provide SST user credentials with the `wsrep_sst_auth` variable.

`wsrep_sst_auth`

Specify authentication credentials for SST as `<sstuser>:<sst_pass>`. You must create this user when Bootstrapping the First Node and provide necessary privileges for it:

```
mysql> CREATE USER 'sstuser'@'localhost' IDENTIFIED BY 'passw0rd';
mysql> GRANT RELOAD, LOCK TABLES, PROCESS, REPLICATION CLIENT ON *.* TO
  'sstuser'@'localhost';
mysql> FLUSH PRIVILEGES;
```

For more information, see Privileges for Percona XtraBackup.

`pxc_strict_mode`

PXC Strict Mode is enabled by default and set to `ENFORCING`, which blocks the use of experimental and unsupported features in Percona XtraDB Cluster.

`binlog_format`

Galera supports only row-level replication, so set `binlog_format=ROW`.

Percona and/or its affiliates © 2023 — Cookie Consent

`default_storage_engine`

Galera fully supports only the InnoDB storage engine. It will not work correctly with MyISAM or any other non-transactional storage engines. Set this variable to `default_storage_engine=InnoDB`.

`innodb_autoinc_lock_mode`

Galera supports only interleaved (`2`) lock mode for InnoDB. Setting the traditional (`0`) or consecutive (`1`) lock mode can cause replication to fail due to unresolved deadlocks. Set this variable to `innodb_autoinc_lock_mode=2`.

4.3.2 Next Steps

After you configure all your nodes, initialize Percona XtraDB Cluster by bootstrapping the first node according to the procedure described in Bootstrapping the First Node.

Last update: 2022-07-20

4.4 Bootstrapping the First Node

After you configure all PXC nodes, initialize the cluster by bootstrapping the first node. The initial node must contain all the data that you want to be replicated to other nodes.

Bootstrapping implies starting the first node without any known cluster addresses: if the `wsrep_cluster_address` variable is empty, *Percona XtraDB Cluster* assumes that this is the first node and initializes the cluster.

Instead of changing the configuration, start the first node with the following command on Debian or Ubuntu:

```
[root@pxc1 ~]# /etc/init.d/mysql bootstrap-pxc
```

Start the first node with the following command on RedHat or CentOS:

```
[root@pxc1 ~]# systemctl start mysql@bootstrap.service
```

When you start the node using the `bootstrap.server` command, it runs in bootstrap mode with `wsrep_cluster_address=gcomm://`. This setting tells the node to initialize the cluster with the `wsrep_cluster_conf_id` variable set to `1`. After you add other nodes to the cluster, you can then restart this node as normal, and it will use standard configuration again.

Note

A service started with `mysql@bootstrap` must be stopped using the same command. For example, the `systemctl stop mysql` command does not stop an instance started with the `mysql@bootstrap` command.

To make sure that the cluster has been initialized, run the following:

```
mysql@pxc1> show status like 'wsrep%';
```

The output shows that the cluster size is 1 node, it is the primary component, the node is in `Synced` state, it is fully connected and ready for write-set replication.

Percona and/or its affiliates © 2023 — Cookie Consent

```
+----------------------------+--------------------------------------+
| Variable_name              | Value                                |
+----------------------------+--------------------------------------+
| wsrep_local_state_uuid     | c2883338-834d-11e2-0800-03c9c68e41ec |
| ...                        | ...                                  |
| wsrep_local_state          | 4                                    |
| wsrep_local_state_comment  | Synced                               |
| ...                        | ...                                  |
| wsrep_cluster_size         | 1                                    |
| wsrep_cluster_status       | Primary                              |
| wsrep_connected            | ON                                   |
| ...                        | ...                                  |
| wsrep_ready                | ON                                   |
+----------------------------+--------------------------------------+
40 rows in set (0.01 sec)
```

Before adding other nodes to your new cluster, create a user for SST and provide the necessary privileges for that user account. The credentials must match those specified when Configuring Nodes for Write-Set Replication.

```
mysql@pxc1> CREATE USER 'sstuser'@'localhost' IDENTIFIED BY 'passw0rd';
mysql@pxc1> GRANT RELOAD, LOCK TABLES, PROCESS, REPLICATION CLIENT ON *.* TO
  'sstuser'@'localhost';
mysql@pxc1> FLUSH PRIVILEGES;
```

For more information, see Privileges for Percona XtraBackup.

4.4.1 Next Steps

After initializing the cluster, you can add other nodes.

Last update: 2022-07-20

4.5 Adding Nodes to Cluster

New `nodes <Node>` that are properly configured are provisioned automatically. When you start a node with the address of at least one other running `node` in the `wsrep_cluster_address` variable, it automatically joins the cluster and synchronizes with it.

> **Note**
>
> Any existing data and configuration will be overwritten to match the data and configuration of the DONOR node. Do not join several nodes at the same time to avoid overhead due to large amounts of traffic when a new node joins.

By default, Percona XtraDB Cluster uses Percona XtraBackup for *State Snapshot Transfer* (SST). This requires the following:

- Set the `wsrep_sst_method` variable to `xtrabackup-v2` and provide SST user credentials with the `wsrep_sst_auth` variable.

For more information, see Configuring Nodes for Write-Set Replication.

- Create a user for SST on the initial node.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

For more information, see Bootstrapping the First Node.

4.5.1 Starting the Second Node

Start the second node by using either of the following commands:

```
[root@pxc2 ~]# /etc/init.d/mysql start
```

or

```
[root@pxc2 ~]# systemctl start mysql
```

After the server starts, it should receive SST automatically.

To check the status of the second node, run the following:

```
mysql@pxc2> show status like 'wsrep%';
```

The output shows that the new node has been successfully added to the cluster. Cluster size is now 2 nodes, it is the primary component, and it is fully connected and ready to receive write-set replication.

```
+----------------------------+--------------------------------------+
| Variable_name              | Value                                |
+----------------------------+--------------------------------------+
| wsrep_local_state_uuid     | c2883338-834d-11e2-0800-03c9c68e41ec |
| ...                        | ...                                  |
| wsrep_local_state          | 4                                    |
| wsrep_local_state_comment  | Synced                               |
| ...                        | ...                                  |
| wsrep_cluster_size         | 2                                    |
| wsrep_cluster_status       | Primary                              |
| wsrep_connected            | ON                                   |
| ...                        | ...                                  |
| wsrep_ready                | ON                                   |
+----------------------------+--------------------------------------+
40 rows in set (0.01 sec)
```

If the state of the second node is `Synced` as in the previous example, then the node received full SST, is synchronized with the cluster, and you can proceed to add the next node.

Note

If the state of the node is `Joiner`, it means that SST hasn't finished. Do not add new nodes until all others are in `Synced` state.

4.5.2 Starting the Third Node

To add the third node, start the node using either command:

```
[root@pxc3 ~]# /etc/init.d/mysql start
```

or

```
[root@pxc3 ~]# systemctl start mysql
```


Percona and/or its affiliates © 2023 — Cookie Consent

To check the status of the third node, run the following:

```
mysql@pxc3> show status like 'wsrep%';
```

The output shows that the new node has been successfully added to the cluster. Cluster size is now 3 nodes, it is the primary component, and it is fully connected and ready to receive write-set replication.

```
+----------------------------+--------------------------------------+
| Variable_name              | Value                                |
+----------------------------+--------------------------------------+
| wsrep_local_state_uuid     | c2883338-834d-11e2-0800-03c9c68e41ec |
| ...                        | ...                                  |
| wsrep_local_state          | 4                                    |
| wsrep_local_state_comment  | Synced                               |
| ...                        | ...                                  |
| wsrep_cluster_size         | 3                                    |
| wsrep_cluster_status       | Primary                              |
| wsrep_connected            | ON                                   |
| ...                        | ...                                  |
| wsrep_ready                | ON                                   |
+----------------------------+--------------------------------------+
40 rows in set (0.01 sec)
```

4.5.3 Next Steps

When you add all nodes to the cluster, you can verify replication by running queries and manipulating data on nodes to see if these changes are synchronized accross the cluster.

Last update: 2022-07-20

4.6 Verifying Replication

Use the following procedure to verify replication by creating a new database on the second node, creating a table for that database on the third node, and adding some records to the table on the first node.

- Create a new database on the second node:

```
mysql@pxc2> CREATE DATABASE percona;
```

The following output confirms that a new database has been created:

```
Query OK, 1 row affected (0.01 sec)
```

- Switch to a newly created database:

```
mysql@pxc3> USE percona;
```

The following output confirms that a database has been changed:

```
Database changed
```

- Create a table on the third node:

Percona LLC and/or its affiliates © 2023 — Cookie Consent

```
mysql@pxc3> CREATE TABLE example (node_id INT PRIMARY KEY, node_name VARCHAR(30));
```

The following output confirms that a table has been created:

```
Query OK, 0 rows affected (0.05 sec)
```

- Insert records on the first node:

```
mysql@pxc1> INSERT INTO percona.example VALUES (1, 'percona1');
```

The following output confirms that the records have been inserted:

```
Query OK, 1 row affected (0.02 sec)
```

- Retrieve rows from that table on the second node:

```
mysql@pxc2> SELECT * FROM percona.example;
```

The following output confirms that all the rows have been retrieved:

```
+---------+-----------+
| node_id | node_name |
+---------+-----------+
|       1 | percona1  |
+---------+-----------+
1 row in set (0.00 sec)
```

4.6.1 Next Steps

- Consider installing ProxySQL on client nodes for efficient workload management across the cluster without any changes to the applications that generate queries. This is the recommended high-availability solution for Percona XtraDB Cluster. For more information, see Load balancing with ProxySQL.
- Percona Monitoring and Management is the best choice for managing and monitoring Percona XtraDB Cluster performance. It provides visibility for the cluster and enables efficient troubleshooting.

Last update: 2022-07-20

Percona and/or its affiliates © 2023 — Cookie Consent

5. Features

5.1 High Availability

In a basic setup with 3 nodes, Percona XtraDB Cluster will continue to function if you take any of the nodes down. At any point in time, you can shut down any node to perform maintenance or make configuration changes. Even in unplanned situations (like a node crashing or if it becomes unavailable over the network), the Percona XtraDB Cluster will continue to work and you'll be able to run queries on working nodes.

If there were changes to data while a node was down, there are two options that the node may use when it joins the cluster again:

- **State Snapshot Transfer** (SST) is when all data is copied from one node to another.

SST is usually used when a new node joins the cluster and receives all data from an existing node. There are three methods of SST available in Percona XtraDB Cluster:

- **mysqldump**
- **rsync**
- **xtrabackup**.

The downside of `mysqldump` and `rsync` is that your cluster becomes **READ-ONLY** while data is being copied (SST applies the **FLUSH TABLES WITH READ LOCK** command).

SST using `xtrabackup` does not require the **READ LOCK** command for the entire syncing process, only for syncing .frm files (the same as with a regular backup).

- **Incremental State Transfer** (IST) is when only incremental changes are copied from one node to another.

Even without locking your cluster in read-only state, SST may be intrusive and disrupt normal operation of your services. IST lets you avoid that. If a node goes down for a short period of time, it can fetch only those changes that happened while it was down. IST is implemeted using a caching mechanism on nodes. Each node contains a cache, ring-buffer (the size is configurable) of last N changes, and the node is able to transfer part of this cache. Obviously, IST can be done only if the amount of changes needed to transfer is less than N. If it exceeds N, then the joining node has to perform SST.

You can monitor the current state of a node using the following command:

```
SHOW STATUS LIKE 'wsrep_local_state_comment';
```

When a node is in `Synced (6)` state, it is part of the cluster and ready to handle traffic.

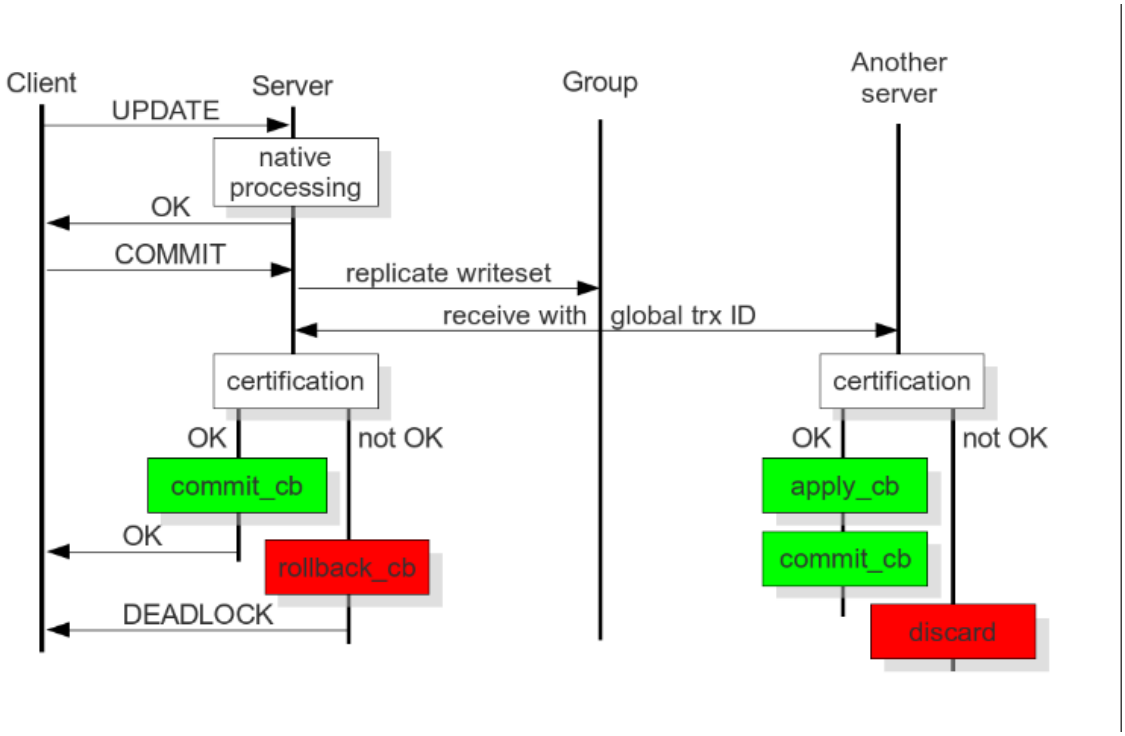
Last update: 2022-07-20

5.2 Multi-Source Replication

Multi-source replication means that you can write to any node and be sure that the write will be consistent for all nodes in the cluster. This is different from regular MySQL replication, where you have to apply writes to source to ensure that it will be synced.

With multi-source replication any write is either committed on all nodes or not committed at all. The following diagram shows how it works for two nodes, but the same logic is applied with any number of nodes in the cluster:

Percona LLC and/or its affiliates © 2023 — Cookie Consent



All queries are executed locally on the node, and there is special handling only on `COMMIT`. When the `COMMIT` query is issued, the transaction has to pass certification on all nodes. If it does not pass, you will receive `ERROR` as the response for that query. After that, the transaction is applied on the local node.

Response time of `COMMIT` includes the following:

- Network round-trip time
- Certification time
- Local applying

Note

Applying the transaction on remote nodes does not affect the response time of `COMMIT`, because it happens in the background after the response on certification.

There are two important consequences of this architecture:

- Several appliers can be used in parallel. This enables truly parallel replication. A replica can have many parallel threads configured using the `wsrep_slave_threads` variable.
- There might be a small period of time when a replica is out of sync. This happens because the source may apply events faster than the replica. And if you do read from the replica, you may read the data that has not changed yet. You can see that from the diagram.

However, this behavior can be changed by setting the `wsrep_causal_reads=ON` variable. In this case, the read on the replica will wait until the event is applied (this will obviously increase the response time of the read). The gap between the replica and the source is the reason why this replication is called *virtually synchronous replication*, and not *real synchronous replication*.

The described behavior of `COMMIT` also has another serious implication. If you run write transactions to two different nodes, the cluster will use an optimistic locking model. This means a transaction will not check on

Percona and/or its affiliates © 2023 — Cookie Consent

possible locking conflicts during the individual queries, but rather on the `COMMIT` stage, and you may get `ERROR` response on `COMMIT`.

This is mentioned because it is one of the incompatibilities with regular *InnoDB* that you might experience. With InnoDB, `DEADLOCK` and `LOCK TIMEOUT` errors usually happen in response to a particular query, but not on `COMMIT`. It is good practice to check the error codes after a `COMMIT` query, but there are still many applications that do not do that.

If you plan to use multi-source replication and run write transactions on several nodes, you may need to make sure you handle the responses on `COMMIT` queries.

Last update: 2022-08-23

5.3 PXC Strict Mode

PXC Strict Mode (`pxc_strict_mode`) is designed to help control behavior of cluster when an experimental/unsupported feature is used. It performs a number of validations at startup and during runtime.

Depending on the actual mode you select, upon encountering a failed validation, the server will either throw an error (halting startup or denying the operation), or log a warning and continue running as normal. The following modes are available:

- `DISABLED`: Do not perform strict mode validations and run as normal; no error or warning is logged for use of experimental/unsupported feature.
- `PERMISSIVE`: If a vaidation fails, log a warning and continue running as normal.
- `ENFORCING`: If a validation fails during startup, halt the server and throw an error. If a validation fails during runtime, deny the operation and throw an error.
- `MASTER`: The same as `ENFORCING` except that explicit table locking is allowed: the validation of explicit table locking is not performed. This mode can be used with clusters in which write operations are isolated to a single node.

Important

Setting `pxc_strict_mode` from `DISABLED` or `PERMISSIVE` to either `ENFORCING` or `MASTER` requires that the following conditions are met:

- `wsrep_replicate_myisam=OFF`
- `binlog_format=ROW`
- `log_output=FILE/NONE`
- transaction isolation level != SERIALIZABLE

By default, PXC Strict Mode is set to `ENFORCING`, except if the node is acting as a standalone server (wsrep_provider=none) or the node is bootstrapping, then PXC Strict Mode defaults to `DISABLED`.

It is recommended to keep PXC Strict Mode set to `ENFORCING`, because in this case whenever Percona XtraDB Cluster encounters an experimental feature or an unsupported operation, the server will deny it. This will force you to re-evaluate your Percona XtraDB Cluster configuration without risking the consistency of your data.

If you are planning to set PXC Strict Mode to anything else than `ENFORCING`, you should be aware of the limitations and effects that this may have on data integrity. For more information, see Validations.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

To set the mode, use the `pxc_strict_mode` variable in the configuration file or the `--pxc-strict-mode` option during `mysqld` startup.

Important

It is better to start the server with the necessary mode (the default `ENFORCING` is highly recommended). However, you can dynamically change it during runtime. For example, to set PXC Strict Mode to `PERMISSIVE`, run the following command:

```
mysql> SET GLOBAL pxc_strict_mode=PERMISSIVE;
```

To further ensure data consistency, it is important to have all nodes in the cluster running with the same configuration, including the value of `pxc_strict_mode` variable.

5.3.1 Validations

PXC Strict Mode validations are designed to ensure optimal operation for common cluster setups that do not require experimental features and do not rely on operations not supported by Percona XtraDB Cluster.

Warning

If an unsupported operation is performed on a node with `pxc_strict_mode` set to `DISABLED` or `PERMISSIVE`, it will not be validated on nodes where it is replicated to, even if the destination node has `pxc_strict_mode` set to `ENFORCING`.

This section describes the purpose and consequences of each validation.

Storage engine

Percona XtraDB Cluster supports only transactional storage engine (XtraDB or InnoDB) as PXC needs capability of the storage engine to rollback any given transaction. Storage engines, such as *MyISAM*, *MEMORY*, *CSV* are not supported.

- Data manipulation statements that perform writing to table (for example, `INSERT`, `UPDATE`, `DELETE`, etc.)
- The following administrative statements: `CHECK`, `OPTIMIZE`, `REPAIR`, and `ANALYZE`
- `TRUNCATE TABLE` and `ALTER TABLE`

Strict Mode	At startup	At runtime
DISABLED	No validation is performed	Allow write/DML, ALTER, TRUNCATE and ADMIN (check, optimize, repair, analyze) operations on all persistent tables (including one that resides in non-transactional storage engine).
PERMISSIVE	No validation is performed.	All operations are permitted, but a warning is logged when an undesirable operation is performed on an unsupported table.
ENFORCING or MASTER	No validation is performed.	write/DML, ALTER, TRUNCATE, and ADMIN operations on each persistent table that is not implemented using a supported storage engine logs an error.

In order to balance backward compatibility and existing application, Percona XtraDB Cluster allows table created with a non-transactional storage engine to co-exist but DML, ALTER, TRUNCATE and ADMIN (ANALYZE, CHECK, etc....) operations on such tables are blocked in ENFORCING mode.

Percona and/or its affiliates © 2023 — Cookie Consent

Altering unsupported table from non-transactional storage engine to a transactional storage engine is allowed.

As operation checks are applicable only to persistent tables, temporary tables are not replicated by Galera Cluster so *pxc-strict-mode* enforcement is not applicable to temporary table. Same is applicable to performance-schema. (Check is also not applied for system tables located in the mysql database.)

MyISAM replication

Percona XtraDB Cluster provides experimental support for replication of tables that use the MyISAM storage engine. Due to the non-transactional nature of MyISAM, it is not likely to ever be fully supported in Percona XtraDB Cluster.

MyISAM replication is controlled using the `wsrep_replicate_myisam` variable, which is set to `OFF` by default. Due to its unreliability, MyISAM replication should not be enabled if you want to ensure data consistency.

Strict Mode	At startup	At runtime
DISABLED	No validation is performed.	You can set `wsrep_replicate_myisam` to any value.
PERMISSIVE	If `wsrep_replicate_myisam` is set to `ON`, a warning is logged and startup continues.	It is permitted to change `wsrep_replicate_myisam` to any value, but if you set it to `ON`, a warning is logged.
ENFORCING or MASTER	If `wsrep_replicate_myisam` is set to `ON`, an error is logged and startup is aborted.	Any attempt to change `wsrep_replicate_myisam` to `ON` fails and an error is logged. Setting it to OFF is welcome operation and doesn't result in an error.

Note

The `wsrep_replicate_myisam` variable controls *replication* for MyISAM tables, and this validation only checks whether it is allowed. Undesirable operations for MyISAM tables are restricted using the Storage engine validation.

Binary log format

Percona XtraDB Cluster supports only the default row-based binary logging format. Setting the `binlog_format` variable to anything but `ROW` at startup is not allowed, because this changes the global scope, which must be set to `ROW`. Validation is performed only at runtime and against session scope.

Strict Mode	At startup	At runtime
DISABLED	No check is enforced.	You can set `binlog_format` to any value.
PERMISSIVE	No check is enforced.	It is permitted to change `binlog_format` to any value, but if you set it to anything other than `ROW`, a warning is logged.
ENFORCING or MASTER	No check is enforced.	At runtime, any attempt to change `binlog_format` to anything other than `ROW` fails and an error is logged. Setting it to ROW is welcome operation and doesn't result in an error.

Percona and/or its affiliates © 2023 — Cookie Consent

Note

Setting *binlog_format* at global level to STATEMENT/MIXED is not allowed under any mode and even during startup (startup setting are global setting).

Tables without the primary key

Percona XtraDB Cluster cannot properly propagate certain write operations to tables that do not have primary keys defined. Undesirable operations include data manipulation statements that perform writing to table (especially `DELETE`).

Note

This type of validation is not applicable to temporary tables, system tables(tables located in mysql database), and performance-schema.

Depending on the selected mode, the following happens:

Strict Mode	At startup	At
DISABLED	No validation is performed.	All operations are permitted.
PERMISSIVE	No validation is performed.	All operations are permitted, but a warning is logged when an undesirable operation is performed on a table without an explicit primary key defined.
ENFORCING or MASTER	No validation is performed.	Any undesirable operation performed on a table without an explicit primary key is denied and an error is logged.

Log output

Percona XtraDB Cluster does not support tables in the MySQL database as the destination for log output. By default, log entries are written to a file. This validation checks the value of the `log_output` variable.

Depending on the selected mode, the following happens:

Strict Mode	At startup	At runtime
DISABLED	No validation is performed.	You can set `log_output` to any value.
PERMISSIVE	If `log_output` is set only to `TABLE`, a warning is logged and startup continues.	It is permitted to change `log_output` to any value, but if you set it only to `TABLE`, a warning is logged. Setting it to FILE/NONE is welcome operation and doesn't result in a warning.
ENFORCING or MASTER	If `log_output` is set only to `TABLE`, an error is logged and startup is aborted.	Any attempt to change `log_output` only to `TABLE` fails and an error is logged. Setting it to FILE/NONE is welcome operation and doesn't result in a warning.

Percona and/or its affiliates © 2023 — Cookie Consent

Explicit table locking

Percona XtraDB Cluster has only experimental support for explicit table locking operations, The following undesirable operations lead to explicit table locking and are covered by this validation:

- `LOCK TABLES`
- `GET_LOCK()` and `RELEASE_LOCK()`
- `FLUSH TABLES <tables> WITH READ LOCK`
- Setting the `SERIALIZABLE` transaction level

Depending on the selected mode, the following happens:

Strict Mode	At startup	At runtime
DISABLED	No validation is performed.	All operations are permitted.
PERMISSIVE	No validation is performed.	All operations are permitted, but a warning is logged when an undesirable operation is performed.
ENFORCING	No validation is performed.	Any undesirable operation is denied and an error is logged.

Auto-increment lock mode

The lock mode for generating auto-increment values must be *interleaved* to ensure that each node generates a unique (but non-sequential) identifier.

This validation checks the value of the `innodb_autoinc_lock_mode` variable. By default, the variable is set to `1` (*consecutive* lock mode), but it should be set to `2` (*interleaved* lock mode).

Note

This validation is not performed during runtime, because the `innodb_autoinc_lock_mode` variable cannot be set dynamically.

Depending on the strict mode selected, the following happens:

Strict Mode	At startup	At runtime
DISABLED	No validation is performed.	This option is not dynamic, no validation is required during runtime.
PERMISSIVE	If `innodb_autoinc_lock_mode` is not set to `2`, a warning is logged and startup continues.	This option is not dynamic, no validation is required during runtime.
ENFORCING or MASTER	If `innodb_autoinc_lock_mode` is not set to `2`, an error is logged and startup is aborted.	This option is not dynamic, no validation is required during runtime.

Combining schema and data changes in a single statement

Percona XtraDB Cluster does not support `CREATE TABLE ... AS SELECT` (CTAS) statements, because they combine both schema and data changes.

Depending on the strict mode selected, the following happens:

Percona and/or its affiliates © 2023 — Cookie Consent

Strict Mode	At startup	At runtime
DISABLED	No validation is performed.	All operations are permitted.
PERMISSIVE	No validation is performed.	All operations are permitted, but a warning is logged when a CTAS operation is performed.
ENFORCING	No validation is performed.	Any CTAS operation is denied and an error is logged.

Note

CTAS operations for temporary tables are permitted even in strict mode.

Discarding and Importing Tablespaces

`DISCARD TABLESPACE` and `IMPORT TABLESPACE` are not replicated using `TOI (Total Order Isolation)`. This can lead to data inconsistency if executed on only one node.

Depending on the strict mode selected, the following happens:

Strict Mode	At startup	At runtime
DISABLED	No validation is performed.	All operations are permitted.
PERMISSIVE	No validation is performed.	All operations are permitted, but a warning is logged when you discard or import a tablespace.
ENFORCING	No validation is performed.	Discarding or importing a tablespace is denied and an error is logged.

XA transactions

XA transaction are not supported by PXC/Galera and with new MySQL-5.7 semantics using xa statement causes reuse of XID which is being used by Galera causing conflicts. XA statements are completely blocked irrespective of the value of `pxc-strict-mode`.

Major version check

This validation checks that the protocol version is the same as the server major version. This validation protects the cluster against writes attempted on already upgraded nodes.

The following output shows the error message:

```
    ERROR 1105 (HY000): Percona-XtraDB-Cluster prohibits use of multiple major versions
while accepting write workload with pxc_strict_mode = ENFORCING or MASTER
```

Last update: 2022-09-29

Percona and/or its affiliates © 2023 — Cookie Consent

6. PXC Security

6.1 Security Basics

By default, Percona XtraDB Cluster does not provide any protection for stored data. There are several considerations to take into account for securing Percona XtraDB Cluster:

- Securing the Network

Anyone with access to your network can connect to any Percona XtraDB Cluster node either as a client or as another node joining the cluster. You should consider restricting access using VPN and filter traffic on ports used by Percona XtraDB Cluster.

- Encrypting PXC Traffic

Unencrypted traffic can potentially be viewed by anyone monitoring your network. You should enable encryption for all nodes in the cluster to prevent this.

- Data-at-rest encryption

Percona XtraDB Cluster supports tablespace encryption to provide at-rest encryption for physical tablespace data files.

For more information, see the following blog post:

```
 * [MySQL Data at Rest Encryption](https://www.percona.com/blog/2016/04/08/mysql-data-at-rest-encryption/)
```

6.1.1 Security Modules

Most modern distributions include special security modules that control access to resources for users and applications. By default, these modules will most likely constrain communication between Percona XtraDB Cluster nodes.

The easiest solution is to disable or remove such programs, however, this is not recommended for production environments. You should instead create necessary security policies for Percona XtraDB Cluster.

SELinux

SELinux is usually enabled by default in Red Hat Enterprise Linux and derivatives (including CentOS). During installation and configuration, you can set the mode to `permissive` by running the following command:

```
setenforce 0
```

Note

This only changes the mode at runtime. To run SELinux in permissive mode after a reboot, set `SELINUX=permissive` in the `/etc/selinux/config` configuration file.

To use SELinux with Percona XtraDB Cluster, you need to create an access policy. For more information, see SELinux and MySQL.

Percona and/or its affiliates © 2023 — Cookie Consent

AppArmor

AppArmor is included in Debian and Ubuntu. During installation and configuration, you can disable AppArmor for `mysqld`:

1. Create the following symbolic link:

```
$ sudo ln -s /etc/apparmor.d/usr /etc/apparmor.d/disable/.sbin.mysqld
```

2. Restart AppArmor:

```
$ sudo service apparmor restart
```

Note

If your system uses `systemd`, run the following command instead:

```
$ sudo systemctl restart apparmor
```

To use AppArmor with Percona XtraDB Cluster, you need to create or extend the MySQL profile. For more information, see AppArmor and MySQL.

Last update: 2022-08-23

6.2 Securing the Network

By default, anyone with access to your network can connect to any Percona XtraDB Cluster node either as a client or as another node joining the cluster. This could potentially let them query your data or get a complete copy of it.

In general, it is a good idea to disable all remote connections to Percona XtraDB Cluster nodes. If you require clients or nodes from outside of your network to connect, you can set up a VPN (virtual private network) for this purpose.

6.2.1 Firewall Configuration

A firewall can let you filter Percona XtraDB Cluster traffic based on the clients and nodes that you trust.

By default, Percona XtraDB Cluster nodes use the following ports:

- 3306 is used for MySQL client connections and SST (State Snapshot Transfer) via `mysqldump`.
- 4444 is used for SST via `rsync` and Percona XtraBackup.
- 4567 is used for write-set replication traffic (over TCP) and multicast replication (over TCP and UDP).
- 4568 is used for IST (Incremental State Transfer).

Ideally you want to make sure that these ports on each node are accessed only from trusted IP addresses. You can implement packet filtering using `iptables`, `firewalld`, `pf`, or any other firewall of your choice.

Percona and/or its affiliates © 2023 — Cookie Consent

Using iptables

To restrict access to Percona XtraDB Cluster ports using `iptables`, you need to append new rules to the `INPUT` chain on the filter table. In the following example, the trusted range of IP addresses is 192.168.0.1/24. It is assumed that only Percona XtraDB Cluster nodes and clients will connect from these IPs. To enable packet filtering, run the commands as root on each Percona XtraDB Cluster node.

```
# iptables --append INPUT --in-interface eth0 \
   --protocol tcp --match tcp --dport 3306 \
   --source 192.168.0.1/24 --jump ACCEPT
# iptables --append INPUT --in-interface eth0 \
   --protocol tcp --match tcp --dport 4444 \
   --source 192.168.0.1/24 --jump ACCEPT
# iptables --append INPUT --in-interface eth0 \
   --protocol tcp --match tcp --dport 4567 \
   --source 192.168.0.1/24 --jump ACCEPT
# iptables --append INPUT --in-interface eth0 \
   --protocol tcp --match tcp --dport 4568 \
   --source 192.168.0.1/24 --jump ACCEPT
# iptables --append INPUT --in-interface eth0 \
   --protocol udp --match udp --dport 4567 \
   --source 192.168.0.1/24 --jump ACCEPT
```

Note

The last one opens port 4567 for multicast replication over UDP.

If the trusted IPs are not in sequence, you will need to run these commands for each address on each node. In this case, you can consider to open all ports between trusted hosts. This is a little bit less secure, but reduces the amount of commands. For example, if you have three Percona XtraDB Cluster nodes, you can run the following commands on each one:

```
# iptables --append INPUT --protocol tcp \
    --source 64.57.102.34 --jump ACCEPT
# iptables --append INPUT --protocol tcp \
    --source 193.166.3.20  --jump ACCEPT
# iptables --append INPUT --protocol tcp \
    --source 193.125.4.10  --jump ACCEPT
```

Running the previous commands will allow TCP connections from the IP addresses of the other Percona XtraDB Cluster nodes.

Percona and/or its affiliates © 2023 — Cookie Consent

Note

The changes that you make in `iptables` are not persistent unless you save the packet filtering state:

```
# service save iptables
```

For distributions that use `systemd`, you need to save the current packet filtering rules to the path where `iptables` reads from when it starts. This path can vary by distribution, but it is usually in the `/etc` directory. For example:

- `/etc/sysconfig/iptables`
- `/etc/iptables/iptables.rules`

Use `iptables-save` to update the file:

```
# iptables-save > /etc/sysconfig/iptables
```

Last update: 2022-08-23

6.3 Encrypting PXC Traffic

There are two kinds of traffic in Percona XtraDB Cluster:

1. Client-Server traffic (the one between client applications and cluster nodes).
2. Replication traffic, that includes SST, IST, write-set replication, and various service messages.

   Percona XtraDB Cluster supports encryption for all types of traffic. Replication traffic encryption can be configured either in automatic or in manual mode.

6.3.1 Encrypting Client-Server Communication

Percona XtraDB Cluster uses the underlying MySQL encryption mechanism to secure communication between client applications and cluster nodes.

Specify the following settings in the `my.cnf` configuration file for each node:

```
[mysqld]
ssl-ca=/etc/mysql/certs/ca.pem
ssl-cert=/etc/mysql/certs/server-cert.pem
ssl-key=/etc/mysql/certs/server-key.pem

[client]
ssl-ca=/etc/mysql/certs/ca.pem
ssl-cert=/etc/mysql/certs/client-cert.pem
ssl-key=/etc/mysql/certs/client-key.pem
```

After restart the node will use these files to encrypt communication with clients. MySQL clients require only the second part of the configuration to communicate with cluster nodes.

Starting from the version 5.7, *MySQL* generates default key and certificate files and places them in data directory. You can either use them or generate new certificates. For generation of new certificate please refer to Generating Keys and Certificates Manually section.

Percona and/or its affiliates © 2023 — Cookie Consent

6.3.2 Encrypting Replication Traffic

Replication traffic refers to the inter-node traffic which includes SST traffic, IST traffic, and replication traffic.

Traffic of each type is transferred via different channel, and so it is important to configure secure channels for all 3 variants to completely secure the replication traffic.

Starting from 5.7, PXC supports a single configuration option which helps to secure complete replication traffic, and is often referred as Automatic Configuration. User can also ignore this and configure security of each channel by specifying independent parameters.

Section below will help, covering this aspect.

6.3.3 SSL Automatic Configuration

Enabling `pxc-encrypt-cluster-traffic`

Percona XtraDB Cluster includes the `pxc-encrypt-cluster-traffic` variable that enables automatic configuration of SSL encryption there-by encrypting SST, IST, and replication traffic.

This variable is not dynamic and so cannot be changed on runtime. To enable automatic configuration of SSL encryption, set `pxc-encrypt-cluster-traffic=ON` in the the `[mysqld]` section of the `my.cnf` file, and restart the cluster (by default it is disabled there-by using non-secured channel for replication).

Note

Setting `pxc-encrypt-cluster-traffic=ON` has effect of applying the following settings in `my.cnf` configuration file:

```
[mysqld]
wsrep_provider_options="socket.ssl_key=server-key.pem;socket.ssl_cert=server-
cert.pem;socket.ssl_ca=ca.pem"

[sst]
encrypt=4
ssl-key=server-key.pem
ssl-ca=ca.pem
ssl-cert=server-cert.pem
```

For `wsrep_provider_options`, only the mentioned options are affected (`socket.ssl_key`, `socket,ssl_cert`, and `socket.ssl_ca`), the rest is not modified.

Automatic configuration of the SSL encryption needs key and certificate files. Starting from the version 5.7, *MySQL* generates default key and certificate files and places them in data directory. These auto-generated files are suitable for automatic SSL configuration, but you should use the same key and certificate files on all nodes. Also you can override auto-generated files with manually created ones, as covered by the Generating Keys and Certificates Manually section.

Necessary key and certificate files are first searched at the `ssl-ca`, `ssl-cert`, and `ssl-key` options under `[mysqld]`. If these options are not set, it then looks in the data directory for `ca.pem`, `server-cert.pem`, and `server-key.pem` files.

Note

The `[sst]` section is not searched.

If all three files are found, they are used to configure encryption. If any of the files is missing, a fatal error is generated.

Percona and/or its affiliates © 2023 — Cookie Consent

6.3.4 SSL Manual Configuration

If user wants to enable encryption for specific channel only or use different certificates or other mix-match, then user can opt for manual configuration. This helps to provide more flexibility to end-users.

To enable encryption manually, the location of the required key and certificate files shoud be specified in the Percona XtraDB Cluster configuration. If you do not have the necessary files, see Generating Keys and Certificates Manually.

Note

Encryption settings are not dynamic. To enable it on a running cluster, you need to restart the entire cluster.

There are three aspects of Percona XtraDB Cluster operation, where you can enable encryption:

- Encrypting SST Traffic

This refers to SST traffic during full data copy from one cluster node (donor) to the joining node (joiner).

- Encrypting Replication Traffic
- Encrypting IST Traffic

This refers to all internal Percona XtraDB Cluster communication, such as, write-set replication, IST, and various service messages.

Encrypting SST Traffic

This refers to full data transfer that usually occurs when a new node (JOINER) joins the cluster and receives data from an existing node (DONOR).

For more information, see State Snapshot Transfer.

Note

If `keyring_file` plugin is used, then SST encryption is mandatory: when copying encrypted data via SST, the keyring must be sent over with the files for decryption. In this case following options are to be set in `my.cnf` on all nodes:

```
early-plugin-load=keyring_file.so
keyring-file-data=/path/to/keyring/file
```

The cluster will not work if keyring configuration across nodes is different.

The following SST methods are available: `xtrabackup`, `rsync`, and `mysqldump`.

XTRABACKUP

This is the default SST method (the `wsrep_sst_method` is set to `xtrabackup-v2`), which uses Percona XtraBackup to perform non-blocking transfer of files. For more information, see Percona XtraBackup SST Configuration.

Percona and/or its affiliates © 2023 — Cookie Consent

Encryption mode for this method is selected using the `encrypt` option:

- `encrypt=0` is the default value, meaning that encryption is disabled.
- `encrypt=1`, `encrypt=2`, and `encrypt=3` have been deprecated.
- `encrypt=4` enables encryption based on key and certificate files generated with OpenSSL. For more information, see Generating Keys and Certificates Manually.

To enable encryption for SST using XtraBackup, specify the location of the keys and certificate files in the each node's configuration under `[sst]`:

```
[sst]
encrypt=4
ssl-ca=/etc/mysql/certs/ca.pem
ssl-cert=/etc/mysql/certs/server-cert.pem
ssl-key=/etc/mysql/certs/server-key.pem
```

Note

SSL clients require DH parameters to be at least 1024 bits, due to the logjam vulnerability. However, versions of `socat` earlier than 1.7.3 use 512-bit parameters. If a `dhparams.pem` file of required length is not found during SST in the data directory, it is generated with 2048 bits, which can take several minutes. To avoid this delay, create the `dhparams.pem` file manually and place it in the data directory before joining the node to the cluster:

```
openssl dhparam -out /path/to/datadir/dhparams.pem 2048
```

For more information, see this blog post.

RSYNC

This SST method does not support encryption. Avoid using this method if you need to secure traffic between DONOR and JOINER nodes. If you using keyring plugin then keyring file needs to be send over from DONOR to JOINER. Avoid using this method in such cases too.

MYSQLDUMP

This SST method dumps data from DONOR and imports it to JOINER. Encryption in this case is performed using the same certificates configured for Encrypting Client-Server Communication, because `mysqldump` connects through the database client.

Here is how to enable encryption for SST using `mysqldump` in a running cluster:

1. Create a user for SST on one of the nodes:

```
mysql> CREATE USER 'sstuser'$'%' IDENTIFIED BY PASSWORD 'sst_password';
```

Note

This user must have the same name and password on all nodes where you want to use `mysqldump` for SST.

2. Grant usage privileges to this user and require SSL:

```
mysql> GRANT USAGE ON *.* TO 'sstuser' REQUIRE SSL;
```


Percona and/or its affiliates © 2023 — Cookie Consent

3. To make sure that the SST user replicated across the cluster, run the following query on another node:

```
mysql> SELECT User, Host, ssl_type FROM mysql.user WHERE User='sstuser';
```

The example of the output is the following:

```
+----------+------+----------+
| User     | Host | ssl_type |
+----------+------+----------+
| sstuser | %    | Any      |
+----------+------+----------+
```

Note

If the `wsrep_OSU_method` is set to ROI, you need to manually create the SST user on each node in the cluster.

4. Specify corresponding certificate files in both `[mysqld]` and `[client]` sections of the configuration file on each node:

```
[mysqld]
ssl-ca=/etc/mysql/certs/ca.pem
ssl-cert=/etc/mysql/certs/server-cert.pem
ssl-key=/etc/mysql/certs/server-key.pem

[client]
ssl-ca=/etc/mysql/certs/ca.pem
ssl-cert=/etc/mysql/certs/client-cert.pem
ssl-key=/etc/mysql/certs/client-key.pem
```

For more information, see Encrypting Client-Server Communication.

5. Also specify the SST user credentials in the `wsrep_sst_auth` variable on each node:

```
[mysqld]
wsrep_sst_auth = sstuser:sst_password
```

6. Restart the cluster with the new configuration.

If you do everything correctly, `mysqldump` will connect to DONOR with the SST user, generate a dump file, and import it to JOINER node.

Encrypting Replication/IST Traffic

Replication traffic refers to the following:

- Write-set replication which is the main workload of Percona XtraDB Cluster (replicating transactions that execute on one node to all other nodes).
- Incremental State Transfer (IST) which is copying only missing transactions from DONOR to JOINER node.
- Service messages which ensure that all nodes are synchronized.

All this traffic is transferred via the same underlying communication channel (`gcomm`). Securing this channel will ensure that IST traffic, write-set replication, and service messages are encrypted. (For IST, a separate channel is configured using the same configuration parameters, so 2 sections are described together).

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

To enable encryption for all these processes, define the paths to the key, certificate and certificate authority files using the following wsrep provider options:

- `socket.ssl_ca`
- `socket.ssl_cert`
- `socket.ssl_key`

To set these options, use the `wsrep_provider_options` variable in the configuration file:

```
wsrep_provider_options="socket.ssl=yes;socket.ssl_ca=/etc/mysql/certs/
ca.pem;socket.ssl_cert=/etc/mysql/certs/server-cert.pem;socket.ssl_key=/etc/mysql/certs/
server-key.pem"
```

Note

You must use the same key and certificate files on all nodes, preferably those used for Encrypting Client-Server Communication.

Check the Upgrading Certificates section on how to upgrade existing certificates.

6.3.5 Generating Keys and Certificates Manually

As mentioned above, *MySQL* generates default key and certificate files and places them in data directory. If user wants to override these certificates, the following new sets of files can be generated:

- *Certificate Authority (CA) key and certificate* to sign the server and client certificates.
- *Server key and certificate* to secure database server activity and write-set replication traffic.
- *Client key and certificate* to secure client communication traffic.

These files should be generated using OpenSSL.

Note

The `Common Name` value used for the server and client keys and certificates must differ from that value used for the CA certificate.

Generating CA Key and Certificate

The Certificate Authority is used to verify the signature on certificates.

1. Generate the CA key file:

```
$ openssl genrsa 2048 > ca-key.pem
```

2. Generate the CA certificate file:

```
$ openssl req -new -x509 -nodes -days 3600
    -key ca-key.pem -out ca.pem
```


Percona and/or its affiliates © 2023 — Cookie Consent

Generating Server Key and Certificate

1. Generate the server key file:

```
$ openssl req -newkey rsa:2048 -days 3600 \
    -nodes -keyout server-key.pem -out server-req.pem
```

2. Remove the passphrase:

```
$ openssl rsa -in server-key.pem -out server-key.pem
```

3. Generate the server certificate file:

```
$ openssl x509 -req -in server-req.pem -days 3600 \
    -CA ca.pem -CAkey ca-key.pem -set_serial 01 \
    -out server-cert.pem
```

Generating Client Key and Certificate

1. Generate the client key file:

```
$ openssl req -newkey rsa:2048 -days 3600 \
    -nodes -keyout client-key.pem -out client-req.pem
```

2. Remove the passphrase:

```
$ openssl rsa -in client-key.pem -out client-key.pem
```

3. Generate the client certificate file:

```
$ openssl x509 -req -in client-req.pem -days 3600 \
   -CA ca.pem -CAkey ca-key.pem -set_serial 01 \
   -out client-cert.pem
```

Verifying Certificates

To verify that the server and client certificates are correctly signed by the CA certificate, run the following command:

```
$ openssl verify -CAfile ca.pem server-cert.pem client-cert.pem
```

If the verification is successful, you should see the following output:

```
server-cert.pem: OK
client-cert.pem: OK
```

Failed validation caused by matching CN

Sometimes, an SSL configuration may fail if the certificate and the CA files contain the same .

Percona LLC and/or its affiliates © 2023 — Cookie Consent

To check if this is the case run `openssl` command as follows and verify that the **CN** field differs for the *Subject* and *Issuer* lines.

```
$ openssl x509 -in server-cert.pem -text -noout
```

To obtain a more compact output run `openssl` specifying -subject and -issuer parameters:

```
$ openssl x509 -in server-cert.pem -subject -issuer -noout
```

Deploying Keys and Certificates

Use a secure method (for example, `scp` or `sftp`) to send the key and certificate files to each node. Place them under the `/etc/mysql/certs/` directory or similar location where you can find them later.

> **Note**
>
> Make sure that this directory is protected with proper permissions. Most likely, you only want to give read permissions to the user running `mysqld`.

The following files are required:

- Certificate Authority certificate file (`ca.pem`)

This file is used to verify signatures.

- Server key and certificate files (`server-key.pem` and `server-cert.pem`)

These files are used to secure database server activity and write-set replication traffic.

- Client key and certificate files (`client-key.pem` and `client-cert.pem`)

These files are required only if the node should act as a MySQL client. For example, if you are planning to perform SST using `mysqldump`.

> **Note**
>
> Upgrading Certificates subsection covers the details on upgrading certificates, if necessary.

Upgrading Certificates

The following procedure shows how to upgrade certificates used for securing replication traffic when there are two nodes in the cluster.

1. Restart the first node with the `socket.ssl_ca` option set to a combination of the the old and new certificates in a single file.

   For example, you can merge contents of `old-ca.pem` and `new-ca.pem` into `upgrade-ca.pem` as follows:

   ```
   cat old-ca.pem > upgrade-ca.pem && \
   cat new-ca.pem >> upgrade-ca.pem
   ```

   Set the `wsrep_provider_options` variable as follows:

Percona and/or its affiliates © 2023 — Cookie Consent

```
wsrep_provider_options="socket.ssl=yes;socket.ssl_ca=/etc/mysql/certs/upgrade-
ca.pem;socket.ssl_cert=/etc/mysql/certs/old-cert.pem;socket.ssl_key=/etc/mysql/certs/
old-key.pem"
```

2. Restart the second node with the `socket.ssl_ca`, `socket.ssl_cert`, and `socket.ssl_key` options set to the corresponding new certificate files.

```
wsrep_provider_options="socket.ssl=yes;socket.ssl_ca=/etc/mysql/certs/new-
ca.pem;socket.ssl_cert=/etc/mysql/certs/new-cert.pem;socket.ssl_key=/etc/mysql/certs/
new-key.pem"
```

3. Restart the first node with the new certificate files, as in the previous step.
4. You can remove the old certificate files.

Last update: 2022-08-23

Percona and/or its affiliates © 2023 — Cookie Consent

7. User's Manual

7.1 State Snapshot Transfer

State Snapshot Transfer (SST) is a full data copy from one node (donor) to the joining node (joiner). It's used when a new node joins the cluster. In order to be synchronized with the cluster, the new node has to receive data from a node that is already part of the cluster.

There are three methods of SST available in Percona XtraDB Cluster:

- `mysqldump`
- `rsync`
- `xtrabackup`

The downside of `mysqldump` and `rsync` is that the donor node becomes **READ-ONLY** while data is being copied. Xtrabackup SST, on the other hand, uses backup locks, which means the Galera provider is not paused at all as with FTWRL (Flush Tables with Read Lock) earlier. The SST method can be configured using the `wsrep_sst_method` variable.

Note

If the `gcs.sync_donor` variable is set to `Yes` (default is `No`), the whole cluster will get blocked if the donor is blocked by SST.

7.1.1 Choosing the SST Donor

If there are no nodes available that can safely perform incremental state transfer (IST), the cluster defaults to SST.

If there are nodes available that can perform IST, the cluster prefers a local node over remote nodes to serve as the donor.

If there are no local nodes available that can perform IST, the cluster chooses a remote node to serve as the donor.

If there are several local and remote nodes that can perform IST, the cluster chooses the node with the highest `seqno` to serve as the donor.

7.1.2 Using Percona Xtrabackup

The default SST method is `xtrabackup-v2` which uses *Percona XtraBackup*. This is the least blocking method that leverages backup locks. XtraBackup is run locally on the donor node, so it's important that the correct user credentials are set up on the donor node. In order for Percona XtraDB Cluster to perform SST using XtraBackup, credentials for connecting to the donor node need to be set up in the `wsrep_sst_auth` variable. Besides the credentials, the datadir needs to be specified in the server configuration file `my.cnf`, otherwise the transfer process will fail.

For more information about the required credentials, see the XtraBackup manual.

To test if the credentials will work, run **innobackupex** on the donor node with the username and password specified in the `wsrep_sst_auth` variable. For example, if the value of `wsrep_sst_auth` is `root:Passw0rd`, the **innobackupex** command should look like this:

Percona and/or its affiliates © 2023 — Cookie Consent

```
innobackupex --user=root --password=Passw0rd /tmp/
```

Detailed information on this method is provided in Percona XtraBackup SST Configuration documentation.

7.1.3 Using mysqldump

This method uses the standard `mysqldump` utility to dump all the databases from the donor node and import them to the joining node. For this method to work, the `wsrep_sst_auth` variable needs to be set up with the root credentials. This method is the slowest and it performs a global lock during SST, which blocks writes to the donor node.

The script used for this method is `/usr/bin/wsrep_sst_mysqldump` and it is included in the Percona XtraDB Cluster binary packages.

7.1.4 Using `rsync`

This method uses `rsync` to copy files from donor to the joining node. In some cases, this can be faster than using XtraBackup, but it requires a global data lock, which will block writes to the donor node. This method doesn't require root credentials to be set up in the `wsrep_sst_auth`.

The script used for this method is `/usr/bin/wsrep_sst_rsync` and it is included in the Percona XtraDB Cluster binary packages.

7.1.5 SST for tables with tablespaces that are not in the data directory

For example:

```
CREATE TABLE t1 (c1 INT PRIMARY KEY) DATA DIRECTORY = '/alternative/directory';
```

The result depends on the SST method:

- SST using `rsync`

SST will report success, however the table's data will not be copied over, since `rsync` just copies the files. You will not be able to access the table on the joiner node:

```
mysql> select * from t1;
```

The example of the output is the following:

```
ERROR 1812 (HY000): Tablespace is missing for table `sbtest`.`t1`.
```

- SST using `mysqldump`

Works as expected. If the file does not exist, it will be created. Otherwise it will attempt to use the file (if the file doesn't have the expected format, an error is returned).

- SST using Percona XtraBackup

XtraBackup will restore the table to the same location on the joiner node. If the target directory does not exist, it will be created. If the target file already exists, an error will be returned, because XtraBackup cannot clear tablespaces not in the data directory.

Percona and/or its affiliates © 2023 — Cookie Consent

7.1.6 Other Reading

- SST Methods for MySQL
- Xtrabackup SST configuration

Last update: 2022-08-16

7.2 Percona XtraBackup SST Configuration

Percona XtraBackup SST works in two stages:

- First it identifies the type of data transfer based on the presence of `xtrabackup_ist` file on the joiner node.
- Then it starts data transfer:
  - In case of SST, it empties the data directory except for some files (`galera.cache`, `sst_in_progress`, `grastate.dat`) and then proceeds with SST
  - In case of IST, it proceeds as before.

Note

As of Percona XtraDB Cluster 5.7, `xtrabackup-v2` is the only XtraBackup SST method.

7.2.1 SST Options

The following options specific to SST can be used in `my.cnf` under `[sst]`.

Note

- Non-integer options which have no default value are disabled if not set.
- `:Match: Yes` implies that option should match on donor and joiner nodes.
- SST script reads `my.cnf` when it runs on either donor or joiner node, not during `mysqld` startup.
- SST options must be specified in the main `my.cnf` file.

streamfmt

Parameter	Description
Values:	xbstream, tar
Default:	xbstream
Match:	Yes

Used to specify the Percona XtraBackup streaming format. The recommended value is `streamfmt=xbstream`. Certain features are not available with `tar`, for instance: encryption, compression, parallel streaming, streaming incremental backups. For more information about the `xbstream` format, see The xbstream Binary.

Percona and/or its affiliates © 2023 — Cookie Consent

transferfmt

Parameter	Description
Values:	`socat`, `nc`
Default:	`socat`
Match:	Yes

Used to specify the data transfer format. The recommended value is the default `transferfmt=socat` because it allows for socket options, such as transfer buffer sizes. For more information, see socat(1).

Note

Using `transferfmt=nc` does not support any of the SSL-based encryption modes (values `2`, `3`, and `4` for the `encrypt` option). Only `encrypt=1` is supported.

tca

Example: tca=/etc/ssl/certs/mycert.crt

Used to specify the full path to the certificate authority (CA) file for `socat` encryption based on OpenSSL.

tcert

Example: tcert=/etc/ssl/certs/mycert.pem

Used to specify the full path to the certificate file in PEM format for `socat` encryption based on OpenSSL.

Note

For more information about `tca` and `tcert`, see https://www.dest-unreach.org/socat/doc/socat-openssltunnel.html. The `tca` is essentially the self-signed certificate in that example, and `tcert` is the PEM file generated after concatenation of the key and the certificate generated earlier. The names of options were chosen to be compatible with `socat` parameter names as well as with MySQL's SSL authentication. For testing you can also download certificates from launchpad.

Note

Irrespective of what is shown in the example, you can use the same .crt and .pem files on all nodes and it will work, since there is no server-client paradigm here, but rather a cluster with homogeneous nodes.

tkey

Example: tkey=/etc/ssl/keys/key.pem

Used to specify the full path to the private key in PEM format for socat encryption based on OpenSSL.

encrypt

Parameter	Description
Values:	0, 1, 2, 3

Percona and/or its affiliates © 2023 — Cookie Consent

Parameter	Description
Default:	0
Match:	Yes

Used to enable and specify SST encryption mode:

- Set `encrypt=0` to disable SST encryption. This is the default value.
- Set `encrypt=1` to perform symmetric SST encryption based on XtraBackup.
- Set `encrypt=2` to perform SST encryption based on OpenSSL with `socat`. Ensure that `socat` is built with OpenSSL: `socat -V | grep OPENSSL`. This is recommended if your nodes are over WAN and security constraints are higher.
- Set `encrypt=3` to perform SST encryption based on SSL for just the key and certificate files as implemented in Galera cluster

It does not provide certificate validation. In order to work correctly, paths to the key and certificate files need to be specified as well, for example:

```
[sst]
encrypt=3
tkey=/etc/mysql/key.pem
tcert=/etc/mysql/cert.pem
```

- Set `encrypt=4` for SST encryption with SSL files generated by MySQL. This is the recommended mode.

Considering that you have all three necessary files:

```
[sst]
encrypt=4
ssl-ca=ca.pem
ssl-cert=server-cert.pem
ssl-key=server-key.pem
```

Note

All encryption modes can only be used when `wsrep_sst_method` is set to `xtrabackup-v2` (which is the default).

For more information, see Encrypting PXC Traffic.

encrypt-algo

Values: AES128, AES192, AES256

Used to specify the SST encryption algorithm. It uses the same values as the `--encryption` option for XtraBackup (see this document). The `encrypt-algo` option is considered only if `encrypt` is set to `1`.

sockopt

Used to specify key/value pairs of socket options, separated by commas, for example:

```
[sst]
sockopt="retry=2,interval=3"
```


Percona and/or its affiliates © 2023 — Cookie Consent

The previous example causes socat to try to connect three times (initial attempt and two retries with a 3-second interval between attempts).

Note

For versions of Percona XtraDB Cluster before 5.7.17-29.20, the value must begin with a comma, for example:

```
[sst]
sockopt=",cipher=AES128"
```

This option only applies when socat is used (`transferfmt=socat`). For more information about socket options, see socat (1).

Note

You can also enable SSL based compression with `sockopt`. This can be used instead of the Percona XtraBackup `compress` option.

ncsockopt

Used to specify socket options for the `netcat` transfer format (`transferfmt=nc`).

progress

Values: 1, path/to/file

Used to specify where to write SST progress. If set to `1`, it writes to MySQL `stderr`. Alternatively, you can specify the full path to a file. If this is a FIFO, it needs to exist and be open on reader end before itself, otherwise `wsrep_sst_xtrabackup` will block indefinitely.

Note

Value of 0 is not valid.

rebuild

Parameter	Description
Values:	0, 1
Default:	0

Used to enable rebuilding of index on joiner node. This is independent of compaction, though compaction enables it. Rebuild of indexes may be used as an optimization.

Note

#1192834 affects this option.

Percona and/or its affiliates © 2023 — Cookie Consent

time

Parameter	Description
Values:	0, 1
Default:	0

Enabling this option instruments key stages of backup and restore in SST.

rlimit

Example: rlimit=128k

Used to set a a ratelimit in bytes. Add a suffix (k, m, g, t) to specify units. For example, `128k` is 128 kilobytes. For more information, see pv(1).

> **Note**
>
> Rate is limited on donor node. The rationale behind this is to not allow SST to saturate the donor's regular cluster operations or to limit the rate for other purposes.

use_extra

Parameter	Description
Values:	0, 1
Default:	0

Used to force SST to use the thread pool's extra_port. Make sure that thread pool is enabled and the `extra_port` option is set in `my.cnf` before you enable this option.

cpat

Default: `'.\*\\.pem$\\|.\*init\\.ok$\\|.\*galera\\.cache$\\|.\*sst_in_progress$\\|.\*\\.sst$\\|.\*gvwstate\\.dat$\\|.\*grastate\\.dat$\\|.\*\\.err$\\|.\*\\.log$\\|.\*RPM_UPGRADE_MARKER$\\|.\*RPM_UPGRADE_HISTORY$'`

Used to define the files that need to be retained in the datadir before running SST, so that the state of the other node can be restored cleanly. For example:

```
[sst]
cpat='.*galera\.cache$\|.*sst_in_progress$\|.*grastate\.dat$\|.*\.err$\|.*\.log$
\|.*RPM_UPGRADE_MARKER$\|.*RPM_UPGRADE_HISTORY$\|.*\.xyz$'
```

> **Note**
>
> This option can only be used when `wsrep_sst_method` is set to `xtrabackup-v2` (which is the default value).

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

compressor

Parameter	Description
Default:	not set (disabled)
Example:	compressor='gzip'

decompressor

Parameter	Description
Default:	not set (disabled)
Example:	decompressor='gzip -dc'

Two previous options enable stream-based compression/decompression. When these options are set, compression/decompression is performed on stream, in contrast to performing decompression after streaming to disk, involving additional I/O. This saves a lot of I/O (up to twice less I/O on joiner node).

You can use any compression utility which works on stream: `gzip`, `pigz` (which is recommended because it is multi-threaded), etc. Compressor has to be set on donor node and decompressor on joiner node (although you can set them vice-versa for configuration homogeneity, it won't affect that particular SST). To use XtraBackup based compression as before, set `compress` under `[xtrabackup]`. Having both enabled won't cause any failure (although you will be wasting CPU cycles).

inno-backup-opts

inno-apply-opts

inno-move-opts

Parameter	Description
Default:	Empty
Type:	Quoted String

This group of options is used to pass XtraBackup options for backup, apply, and move stages. The SST script doesn't alter, tweak, or optimize these options.

Note

Although these options are related to XtraBackup SST, they cannot be specified in `my.cnf`, because they are for passing innobackupex options.

sst-initial-timeout

Parameter	Description
Default:	100
Unit:	seconds

Percona and/or its affiliates © 2023 — Cookie Consent

This option is used to configure initial timeout (in seconds) to receive the first packet via SST. This has been implemented, so that if the donor node fails somewhere in the process, the joiner node will not hang up and wait forever.

By default, the joiner node will not wait for more than 100 seconds to get a donor node. The default should be sufficient, however, it is configurable, so you can set it appropriately for your cluster. To disable initial SST timeout, set `sst-initial-timeout=0`.

Note

If you are using `wsrep_sst_donor`, and you want the joiner node to strictly wait for donors listed in the variable and not fall back (that is, without a terminating comma at the end), **and** there is a possibility of **all** nodes in that variable to be unavailable, disable initial SST timeout or set it to a higher value (maximum threshold that you want the joiner node to wait). You can also disable this option (or set it to a higher value) if you believe all other nodes in the cluster can potentially become unavailable at any point in time (mostly in small clusters) or there is a high network latency or network disturbance (which can cause donor selection to take longer than 100 seconds).

sst-idle-timeout

Parameter	Description
Version:	Introducted in `5.7.34-31.51`
Default:	120
Unit:	seconds

This option configures the time the SST operation waits on the joiner to receive more data. The size of the joiner's sst directory is checked for the amount of data received. For example, the directory has received 50MB of data. The operation checks the data size again after the 120 seconds, the default value, has elapsed. If the data size is still 50MB, this operation is aborted. If the data has increased, the operation continues.

An example of setting the option:

```
[sst]
sst-idle-timeout=0
```

tmpdir

Parameter	Description
Version:	Introducted in `5.7.17-29.20`
Default:	Empty
Unit:	/path/to/tmp/dir

This option specifies the location for storing the temporary file on a donor node where the transaction log is stored before streaming or copying it to a remote host.

Percona and/or its affiliates © 2023 — Cookie Consent

Note

Starting from Percona XtraDB Cluster `5.7.20-29.24` this option can be used on joiner node also, to specify non-default location to receive temporary SST files. This location must be large enough to hold the contents of the entire database. If tmpdir is empty then default location datadir/.sst will be used.

The `tmpdir` option can be set in the following `my.cnf` groups:

- `[sst]` is the primary location (others are ignored)
- `[xtrabackup]` is the secondary location (if not specified under `[sst]`)
- `[mysqld]` is used if it is not specified in either of the above

`wsrep_debug`

Specifies whether additional debugging output for the database server error log should be enabled. Disabled by default.

This option can be set in the following `my.cnf` groups:

- Under `[mysqld]` it enables debug logging for `mysqld` and the SST script
- Under `[sst]` it enables debug logging for the SST script only

encrypt_threads

Parameter	Description
Version:	Introducted in `5.7.19-29.22`
Default:	`4`

Specifies the number of threads that XtraBackup should use for encrypting data (when `encrypt=1`). The value is passed using the `--encrypt-threads` option in XtraBackup.

This option affects only SST with XtraBackup and should be specified under the `[sst]` group.

backup_threads

Parameter	Description
Version:	Introducted in `5.7.19-29.22`
Default:	`4`

Specifies the number of threads that XtraBackup should use to create backups. See the `--parallel` option in XtraBackup.

This option affects only SST with XtraBackup and should be specified under the `[sst]` group.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

7.2.2 XtraBackup SST Dependencies

Each suppored version of Percona XtraDB Cluster is tested against a specific version of Percona XtraBackup:

- Percona XtraDB Cluster 5.6 requires Percona XtraBackup 2.3
- Percona XtraDB Cluster 5.7 requires Percona XtraBackup 2.4
- Percona XtraDB Cluster 8.0 requires Percona XtraBackup 8.0

Other combinations are not guaranteed to work.

The following are optional dependencies of Percona XtraDB Cluster introduced by `wsrep_sst_xtrabackup-v2` (except for obvious and direct dependencies):

- `qpress` for decompression. It is an optional dependency of *Percona XtraBackup* and it is available in our software repositories.
- `my_print_defaults` to extract values from `my.cnf`. Provided by the server package.
- `openbsd-netcat` or `socat` for transfer. `socat` is a direct dependency of Percona XtraDB Cluster and it is the default.
- `xbstream` or `tar` for streaming. `xbstream` is the default.
- `pv` is required for `progress` and `rlimit`.
- `mkfifo` is required for `progress`. Provided by `coreutils`.
- `mktemp` is required. Provided by `coreutils`.
- `which` is required.

7.2.3 XtraBackup-based Encryption

This is enabled when `encrypt` is set to `1` under `[sst]` in `my.cnf`. However, due to bug #1190335, it will also be enabled when you specify any of the following options under `[xtrabackup]` in `my.cnf`:

- `encrypt`
- `encrypt-key`
- `encrypt-key-file`

There is no way to disable encryption from innobackupex if any of the above are in `my.cnf` under `[xtrabackup]`. For that reason, consider the following scenarios:

1. If you want to use XtraBackup-based encryption for SST but not otherwise, use `encrypt=1` under `[sst]` and provide the above XtraBackup encryption options under `[sst]`. Details of those options can be found here.
2. If you want to use XtraBackup-based encryption always, use `encrypt=1` under `[sst]` and have the above XtraBackup encryption options either under `[sst]` or `[xtrabackup]`.
3. If you don't want to use XtraBackup-based encryption for SST, but want it otherwise, use `encrypt=0` or `encrypt=2` and do **NOT** provide any XtraBackup encryption options under `[xtrabackup]`. You can still have them under `[sst]` though. You will need to provide those options on innobackupex command line then.
4. If you don't want to use XtraBackup-based encryption at all (or only the OpenSSL-based for SST with `encrypt=2`), don't provide any XtraBackup encryption options in `my.cnf`.

Percona and/or its affiliates © 2023 — Cookie Consent

Note

The `encrypt` option under `[sst]` is different from the one under `[xtrabackup]`. The former is for disabling/changing encryption mode, while the latter is to provide an encryption algorithm. To disambiguate, if you need to provide the latter under `[sst]` (for example, in cases 1 and 2 above), it should be specified as `encrypt-algo`.

Warning

An implication of the above is that if you specify any of the XtraBackup encryption options, and `encrypt=0` under `[sst]`, it will still be encrypted and SST will fail. Look at case 3 above for resolution.

Warning

It is insecure to use the `encrypt-key` option when performing an SST with xtrabackup-v2 and encrypt=1 (using `wsrep_sst_method='xtrabackup-v2'` under `[mysqld]` and `encrypt=1` under `[sst]`) since the key will appear on the command line, and will be visible via `ps`. Therefore it is strongly recommended to place the key into a file and use the `encrypt-key-file` option.

7.2.4 Memory Allocation

The amount of memory for XtraBackup is defined by the `--use-memory` option. You can pass it using the `inno-apply-opts` option under `[sst]` as follows:

```
[sst]
inno-apply-opts="--use-memory=500M"
```

If it is not specified, the `use-memory` option under `[xtrabackup]` will be used:

```
[xtrabackup]
use-memory=32M
```

If neither of the above are specified, the size of the InnoDB memory buffer will be used:

```
[mysqld]
innodb_buffer_pool_size=24M
```

Last update: 2022-08-23

7.3 Restarting the cluster nodes

To restart a cluster node, shut down MySQL and restarting it. The node should leave the cluster (and the total vote count for quorum should decrement).

When it rejoins, the node should synchronize using IST. If the set of changes needed for IST are not found in the `gcache` file on any other node in the entire cluster, then SST will be performed instead. Therefore, restarting cluster nodes for rolling configuration changes or software upgrades is rather simple from the cluster's perspective.

Percona and/or its affiliates © 2023 — Cookie Consent

Note

If you restart a node with an invalid configuration change that prevents MySQL from loading, Galera will drop the node's state and force an SST for that node.

Note

If MySQL fails for any reason, it will not remove its PID file (which is by design deleted only on clean shutdown). Obviously server will not restart if existing PID file is present. So in case of encountered MySQL failure for any reason with the relevant records in log, PID file should be removed manually.

Last update: 2022-07-20

7.4 Cluster Failover

Cluster membership is determined simply by which nodes are connected to the rest of the cluster; there is no configuration setting explicitly defining the list of all possible cluster nodes. Therefore, every time a node joins the cluster, the total size of the cluster is increased and when a node leaves (gracefully) the size is decreased.

The size of the cluster is used to determine the required votes to achieve quorum. A quorum vote is done when a node or nodes are suspected to no longer be part of the cluster (they do not respond). This no response timeout is the `evs.suspect_timeout` setting in the `wsrep_provider_options` (default 5 sec), and when a node goes down ungracefully, write operations will be blocked on the cluster for slightly longer than that timeout.

Once a node (or nodes) is determined to be disconnected, then the remaining nodes cast a quorum vote, and if the majority of nodes from before the disconnect are still still connected, then that partition remains up. In the case of a network partition, some nodes will be alive and active on each side of the network disconnect. In this case, only the quorum will continue. The partition(s) without quorum will change to non-primary state.

As a consequence, it's not possible to have safe automatic failover in a 2 node cluster, because failure of one node will cause the remaining node to become non-primary. Moreover, any cluster with an even number of nodes (say two nodes in two different switches) have some possibility of a *split brain* situation, when neither partition is able to retain quorum if connection between them is lost, and so they both become non-primary.

Therefore, for automatic failover, the *rule of 3s* is recommended. It applies at various levels of your infrastructure, depending on how far the cluster is spread out to avoid single points of failure. For example:

- A cluster on a single switch should have 3 nodes
- A cluster spanning switches should be spread evenly across at least 3 switches
- A cluster spanning networks should span at least 3 networks
- A cluster spanning data centers should span at least 3 data centers

These rules will prevent split brain situations and ensure automatic failover works correctly.

7.4.1 Using an arbitrator

If it is too expensive to add a third node, switch, network, or datacenter, you should use an arbitrator. An arbitrator is a voting member of the cluster that can receive and relay replication, but it does not persist any

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

data, and runs its own daemon instead of `mysqld`. Placing even a single arbitrator in a 3rd location can add split brain protection to a cluster that is spread across only two nodes/locations.

7.4.2 Recovering a Non-Primary cluster

It is important to note that the *rule of 3s* applies only to automatic failover. In the event of a 2-node cluster (or in the event of some other outage that leaves a minority of nodes active), the failure of one node will cause the other to become non-primary and refuse operations. However, you can recover the node from non-primary state using the following command:

```
SET GLOBAL wsrep_provider_options='pc.bootstrap=true';
```

This will tell the node (and all nodes still connected to its partition) that it can become a primary cluster. However, this is only safe to do when you are sure there is no other partition operating in primary as well, or else Percona XtraDB Cluster will allow those two partitions to diverge (and you will end up with two databases that are impossible to re-merge automatically).

For example, assume there are two data centers, where one is primary and one is for disaster recovery, with an even number of nodes in each. When an extra arbitrator node is run only in the primary data center, the following high availability features will be available:

- Auto-failover of any single node or nodes within the primary or secondary data center
- Failure of the secondary data center would not cause the primary to go down (because of the arbitrator)
- Failure of the primary data center would leave the secondary in a non-primary state.
- If a disaster-recovery failover has been executed, you can tell the secondary data center to bootstrap itself with a single command, but disaster-recovery failover remains in your control.

7.4.3 Other Reading

- PXC - Failure Scenarios with only 2 nodes

Last update: 2022-08-23

7.5 Monitoring the cluster

Each node can have a different view of the cluster. There is no centralized node to monitor. To track down the source of issues, you have to monitor each node independently.

Values of many variables depend on the node from which you are querying. For example, replication sent from a node and writes received by all other nodes.

Having data from all nodes can help you understand where flow messages are coming from, which node sends excessively large transactions, and so on.

7.5.1 Manual Monitoring

Manual cluster monitoring can be performed using myq-tools.

Percona and/or its affiliates © 2023 — Cookie Consent

7.5.2 Alerting

Besides standard MySQL alerting, you should use at least the following triggers specific to Percona XtraDB Cluster:

- Cluster state of each node

`wsrep_cluster_status` != Primary

- Node state

`wsrep_connected` != `ON`

`wsrep_ready` != `ON`

For additional alerting, consider the following:

- Excessive replication conflicts can be identtified using the `wsrep_local_cert_failures` and `wsrep_local_bf_aborts` variables
- Excessive flow control messages can be identified using the `wsrep_flow_control_sent` and `wsrep_flow_control_recv` variables
- Large replication queues can be identified using the `wsrep_local_recv_queue`.

7.5.3 Metrics

Cluster metrics collection for long-term graphing should be done at least for the following:

- Queue sizes:

`wsrep_local_recv_queue` and `wsrep_local_send_queue`

- Flow control:

`wsrep_flow_control_sent` and `wsrep_flow_control_recv`

- Number of transactions for a node:

`wsrep_replicated` and `wsrep_received`

- Number of transactions in bytes:

`wsrep_replicated_bytes` and `wsrep_received_bytes`

- Replication conflicts:

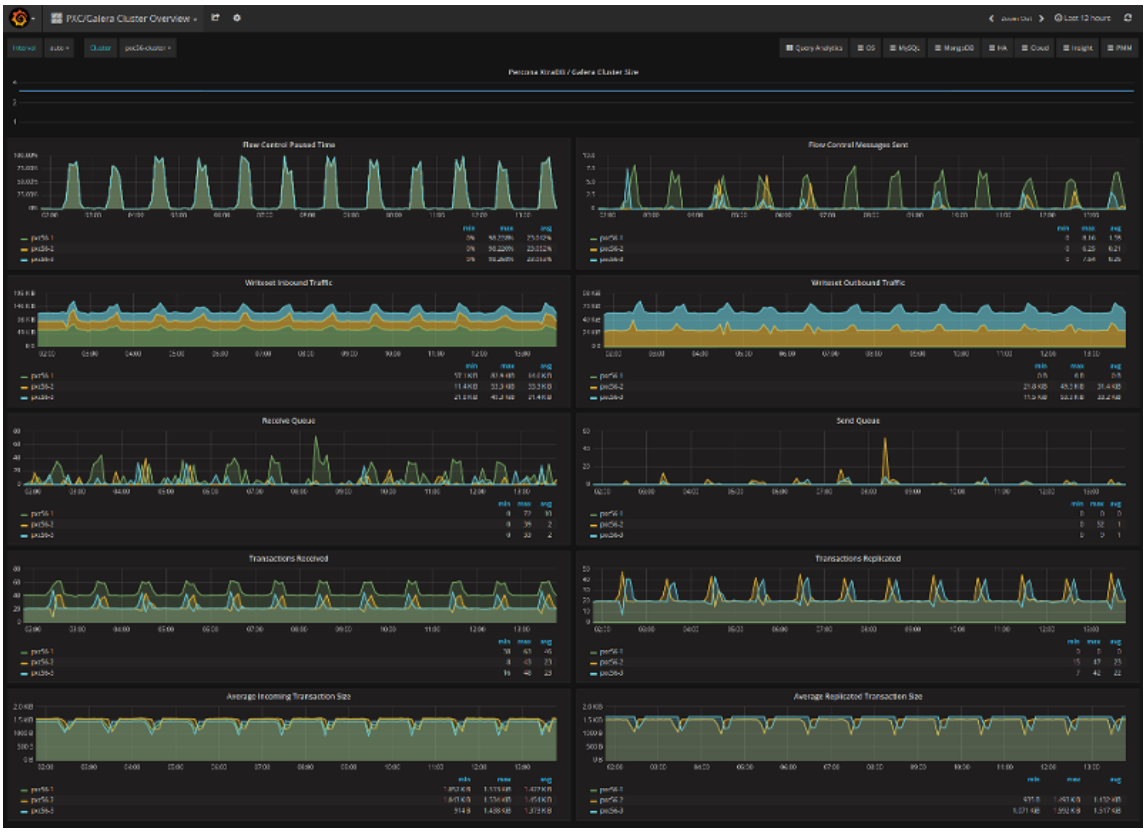
`wsrep_local_cert_failures` and `wsrep_local_bf_aborts`

7.5.4 Using Percona Monitoring and Management

Percona Monitoring and Management includes two dashboards to monitor PXC:

1. PXC/Galera Cluster Overview:

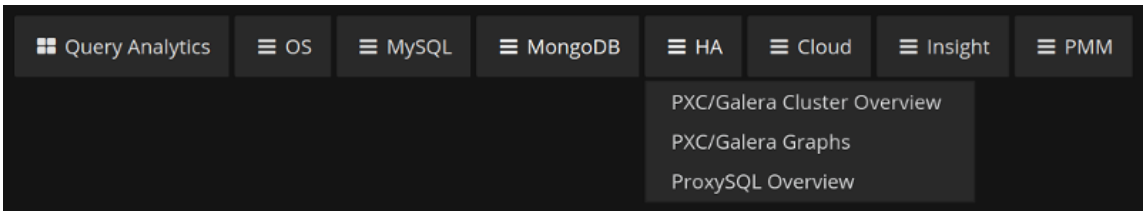
<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent



2. PXC/Galera Graphs:

Percona LLC and/or its affiliates © 2023 — Cookie Consent

These dashboards are available from the menu:



Please refer to the official documentation for details on Percona Monitoring and Management installation and setup.

7.5.5 Other Reading

- Realtime stats to pay attention to in PXC and Galera

Last update: 2022-08-23

7.6 Certification in Percona XtraDB Cluster

Percona XtraDB Cluster replicates actions executed on one node to all other nodes in the cluster, and makes it fast enough to appear as if it is synchronous (*virtually synchronous*).

Percona LLC and/or its affiliates © 2023 — Cookie Consent

The following types of actions exist:

- DDL actions are executed using *Total Order Isolation* (TOI). We can ignore *Rolling Schema Upgrades* (ROI).
- DML actions are executed using normal Galera replication protocol.

Note

This manual page assumes the reader is aware of TOI and MySQL replication protocol.

DML (`INSERT`, `UPDATE`, and `DELETE`) operations effectively change the state of the database, and all such operations are recorded in *XtraDB* by registering a unique object identifier (key) for each change (an update or a new addition).

- A transaction can change an arbitrary number of different data objects. Each such object change is recorded in *XtraDB* using an `append_key` operation. An `append_key` operation registers the key of the data object that has undergone change by the transaction. The key for rows can be represented in three parts as `db_name`, `table_name`, and `pk_columns_for_table` (if `pk` is absent, a hash of the complete row is calculated).

This ensures that there is quick and short meta information about the rows that this transaction has touched or modified. This information is passed on as part of the write-set for certification to all the nodes in the cluster while the transaction is in the commit phase.

- For a transaction to commit, it has to pass XtraDB/Galera certification, ensuring that transactions don't conflict with any other changes posted on the cluster group/channel. Certification will add the keys modified by a given transaction to its own central certification vector (CCV), represented by `cert_index_ng`. If the said key is already part of the vector, then conflict resolution checks are triggered.
- Conflict resolution traces the reference transaction (that last modified this item in the cluster group). If this reference transaction is from some other node, that suggests the same data was modified by the other node, and changes of that node have been certified by the local node that is executing the check. In such cases, the transaction that arrived later fails to certify.

Changes made to database objects are bin-logged. This is similar to how *MySQL* does it for replication with its Source-Replica ecosystem, except that a packet of changes from a given transaction is created and named as a write-set.

Once the client/user issues a `COMMIT`, Percona XtraDB Cluster will run a commit hook. Commit hooks ensure the following:

- Flush the binary logs.
- Check if the transaction needs replication (not needed for read-only transactions like `SELECT`).
- If a transaction needs replication, then it invokes a pre-commit hook in the Galera ecosystem. During this pre-commit hook, a write-set is written in the group channel by a *replicate* operation. All nodes (including the one that executed the transaction) subscribe to this group-channel and read the write-set.
- `gcs_recv_thread` is the first to receive the packet, which is then processed through different action handlers.
- Each packet read from the group-channel is assigned an `id`, which is a locally maintained counter by each node in sync with the group. When any new node joins the group/cluster, a seed-id for it is initialized to the current active id from group/cluster.

There is an inherent assumption/protocol enforcement that all nodes read the packet from a channel in the same order, and that way even though each packet doesn't carry `id` information, it is inherently established using the locally maintained `id` value.

Percona and/or its affiliates © 2023 — Cookie Consent

7.6.1 Common Situation

The following example shows what happens in a common situation. `act_id` is incremented and assigned only for totally ordered actions, and only in primary state (skip messages while in state exchange).

```
rcvd->id = ++group->act_id_;
```

Note

This is an amazing way to solve the problem of the id coordination in multi-source systems. Otherwise a node will have to first get an id from central system or through a separate agreed protocol, and then use it for the packet, thereby doubling the round-trip time.

7.6.2 Conflicts

The following happens if two nodes get ready with their packet at same time:

1. Both nodes will be allowed to put the packet on the channel. That means the channel will see packets from different nodes queued one behind another.
2. The following example shows what happens if two nodes modify same set of rows. Nodes are in sync until this point:

   ```
   create -> insert (1,2,3,4)
   ```

   - Node 1: `update i = i + 10;`
   - Node 2: `update i = i + 100;`

   Let's associate transaction ID (`trx-id`) for an update transaction that is executed on Node 1 and Node 2 in parallel. Although the real algorithm is more involved (with `uuid` + `seqno`), it is conceptually the same, so we are using `trx_id`.

   - Node 1: `update action: trx-id=n1x`
   - Node 2: `update action: trx-id=n2x`

   Both node packets are added to the channel, but the transactions are conflicting. The protocol says: FIRST WRITE WINS.

   So in this case, whoever is first to write to the channel will get certified. Let's say Node 2 is first to write the packet, and then Node 1 makes changes immediately after it.

   Note

   Each node subscribes to all packages, including its own package.

   - Node 2 will see its own packet and will process it. Then it will see the packet from Node 1, try to certify it, and fail.
   - Node 1 will see the packet from Node 2 and will process it.

Percona and/or its affiliates © 2023 — Cookie Consent

> **Note**
>
> InnoDB allows isolation, so Node 1 can process packets from Node 2 independent of Node 1 transaction changes

Then Node 1 will see its own packet, try to certify it, and fail.

> **Note**
>
> Even though the packet originated from Node 1, it will undergo certification to catch cases like these.

7.6.3 Resolving Certification Conflicts

The certification protocol can be described using the previous example. The central certification vector (CCV) is updated to reflect reference transaction.

- Node 2 sees its own packet for certification, adds it to its local CCV and performs certification checks. Once these checks pass, it updates the reference transaction by setting it to `n2x`.

Node 2 then gets the packet from Node 1 for certification. The packet key is already present in CCV, with the reference transaction set it to `n2x`, whereas write-set proposes setting it to `n1x`. This causes a conflict, which in turn causes the transaction from Node 1 to fail the certification test.

- Node 1 sees the packet from Node 2 for certification, which is then processed, the local CCV is updated, and the reference transaction is set to `n2x`.

Using the same case as explained above, Node 1 certification also rejects the packet from Node 1.

This suggests that the node doesn't need to wait for certification to complete, but just needs to ensure that the packet is written to the channel. The applier transaction will always win and the local conflicting transaction will be rolled back.

The following example shows what happens if one of the nodes has local changes that are not synced with the group:

```
create (id primary key) -> insert (1), (2), (3), (4);
```

The example of the output is the following:

```
node-1: wsrep_on=0; insert (5); wsrep_on=1
node-2: insert(5).
```

The `insert(5)` statement will generate a write-set that will then be replicated to Node 1. Node 1 will try to apply it but will fail with `duplicate-key-error`, because 5 already exist.

XtraDB will flag this as an error, which would eventually cause Node 1 to shutdown.

7.6.4 Incrementing GTID

GTID is incremented only when the transaction passes certification, and is ready for commit. That way errant packets don't cause GTID to increment.

Also, group packet `id` is not confused with GTID. Without errant packets, it may seem that these two counters are the same, but they are not related.

Percona and/or its affiliates © 2023 — Cookie Consent

Last update: 2022-07-20

7.7 Percona XtraDB Cluster threading model

Percona XtraDB Cluster creates a set of threads to service its operations, which are not related to existing *MySQL* threads. There are three main groups of threads:

7.7.1 Applier threads

Applier threads apply write-sets that the node receives from other nodes. Write messages are directed through `gcv_recv_thread`.

The number of applier threads is controlled using the `wsrep_slave_threads` variable. The default value is `1`, which means at least one wsrep applier thread exists to process the request.

Applier threads wait for an event, and once it gets the event, it applies it using normal replica apply routine path, and relays the log info apply path with wsrep-customization. These threads are similar to replica worker threads (but not exactly the same).

Coordination is achieved using *Apply and Commit Monitor*. A transaction passes through two important states: `APPLY` and `COMMIT`. Every transaction registers itself with an apply monitor, where its apply order is defined. So all transactions with apply order sequence number (`seqno`) of less than this transaction's sequence number, are applied before applying this transaction. The same is done for commit as well (`last_left >= trx_.depends_seqno()`).

7.7.2 Rollback thread

There is only one rollback thread to perform rollbacks in case of conflicts.

- Transactions executed in parallel can conflict and may need to roll back.
- Applier transactions always take priority over local transactions. This is natural, as applier transactions have been accepted by the cluster, and some of the nodes may have already applied them. Local conflicting transactions still have a window to rollback.

All the transactions that need to be rolled back are added to the rollback queue, and the rollback thread is notified. The rollback thread then iterates over the queue and performs rollback operations.

If a transaction is active on a node, and a node receives a transaction write-set from the cluster group that conflicts with the local active transaction, then such local transactions are always treated as a victim transaction to roll back.

Transactions can be in a commit state or an execution stage when the conflict arises. Local transactions in the execution stage are forcibly killed so that the waiting applier transaction is allowed to proceed. Local transactions in the commit stage fail with a certification error.

Percona LLC and/or its affiliates © 2023 — Cookie Consent

7.7.3 Other threads

Service thread

This thread is created during boot-up and used to perform auxiliary services. It has two main functions:

- It releases the GCache buffer after the cached write-set is purged up to the said level.
- It notifies the cluster group that the respective node has committed a transaction up to this level. Each node maintains some basic status info about other nodes in the cluster. On receiving the message, the information is updated in this local metadata.

Receiving thread

The `gcs_recv_thread` thread is the first one to see all the messages received in a group.

It will try to assign actions against each message it receives. It adds these messages to a central FIFO queue, which are then processed by the Applier threads. Messages can include different operations like state change, configuration update, flow-control, and so on.

One important action is processing a write-set, which actually is applying transactions to database objects.

Gcomm connection thread

The gcomm connection thread `GCommConn::run_fn` is used to co-ordinate the low-level group communication activity. Think of it as a black box meant for communication.

Action-based threads

Besides the above, some threads are created on a needed basis. SST creates threads for donor and joiner (which eventually forks out a child process to host the needed SST script), IST creates receiver and async sender threads, PageStore creates a background thread for removing the files that were created.

If the checksum is enabled and the replicated write-set is big enough, the checksum is done as part of a separate thread.

Last update: 2022-08-16

7.8 Understanding GCache and Record-Set Cache

In Percona XtraDB Cluster, there is a concept of GCache and Record-Set cache (which can also be called transaction write-set cache). The use of these two caches is often confusing if you are running long transactions, because both of them result in the creation of disk-level files. This manual describes what their main differences are.

7.8.1 Record-Set Cache

When you run a long-running transaction on any particular node, it will try to append a key for each row that it tries to modify (the key is a unique identifier for the row `{db,table,pk.columns}`). This information is cached in out-write-set, which is then sent to the group for certification.

Keys are cached in HeapStore (which has `page-size=64K` and `total-size=4MB`). If the transaction data-size outgrows this limit, then the storage is switched from Heap to Page (which has `page-size=64MB` and `total-limit=free-space-on-disk`).

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

All these limits are non-configurable, but having a memory-page size greater than 4MB per transaction can cause things to stall due to memory pressure, so this limit is reasonable. This is another limitation to address when Galera supports large transaction.

The same long-running transaction will also generate binlog data that also appends to out-write-set on commit (`HeapStore->FileStore`). This data can be significant, as it is a binlog image of rows inserted/updated/deleted by the transaction. The `wsrep_max_ws_size` variable controls the size of this part of the write-set. The threshold doesn't consider size allocated for caching-keys and the header.

If `FileStore` is used, it creates a file on the disk (with names like `xxxx_keys` and `xxxx_data`) to store the cache data. These files are kept until a transaction is committed, so the lifetime of the transaction is linked.

When the node is done with the transaction and is about to commit, it will generate the final-write-set using the two files (if the data size grew enough to use `FileStore`) plus `HEADER`, and will publish it for certification to cluster.

The native node executing the transaction will also act as subscription node, and will receive its own write-set through the cluster publish mechanism. This time, the native node will try to cache write-set into its GCache. How much data GCache retains is controlled by the GCache configuration.

7.8.2 GCache

GCache holds the write-set published on the cluster for replication. The lifetime of write-set in GCache is not transaction-linked.

When a `JOINER` node needs an IST, it will be serviced through this GCache (if possible).

GCache will also create the files to disk. You can read more about it here.

At any given point in time, the native node has two copies of the write-set: one in GCache and another in Record-Set Cache.

For example, lets say you `INSERT/UPDATE` 2 million rows in a table with the following schema.

```
(int, char(100), char(100) with pk (int, char(100))
```

It will create write-set key/data files in the background similar to the following:

```
-rw------- 1 xxx xxx 67108864 Apr 11 12:26 0x00000707_data.000000
-rw------- 1 xxx xxx 67108864 Apr 11 12:26 0x00000707_data.000001
-rw------- 1 xxx xxx 67108864 Apr 11 12:26 0x00000707_data.000002
-rw------- 1 xxx xxx 67108864 Apr 11 12:26 0x00000707_keys.000000
```

Last update: 2022-07-20

7.9 Perfomance Schema Instrumentation

To improve monitoring *Percona XtraDB Cluster* has implemented an infrastructure to expose Galera instruments (mutexes, cond-variables, files, threads) as a part of `PERFORMANCE_SCHEMA`.

Although mutexes and condition variables from `wsrep` were already part of `PERFORMANCE_SCHEMA` threads were not.

Mutexes, condition variables, threads, and files from Galera library also were not part of the `PERFORMANCE_SCHEMA`.

Percona LLC and/or its affiliates © 2023 — Cookie Consent

You can see the complete list of available instruments by running:

```
mysql> SELECT * FROM performance_schema.setup_instruments WHERE name LIKE '%galera%' OR
name LIKE '%wsrep%';
```

The example of the output is the following:

```
+------------------------------------------------------------+---------+-------+
| NAME                                                       | ENABLED | TIMED |
+------------------------------------------------------------+---------+-------+
| wait/synch/mutex/sql/LOCK_wsrep_ready                      | NO      | NO    |
| wait/synch/mutex/sql/LOCK_wsrep_sst                        | NO      | NO    |
| wait/synch/mutex/sql/LOCK_wsrep_sst_init                   | NO      | NO    |
...
| stage/wsrep/wsrep: in rollback thread                      | NO      | NO    |
| stage/wsrep/wsrep: aborter idle                            | NO      | NO    |
| stage/wsrep/wsrep: aborter active                          | NO      | NO    |
+------------------------------------------------------------+---------+-------+
73 rows in set (0.00 sec)
```

Some of the most important are:

- Two main actions that Galera does are `REPLICATION` and `ROLLBACK`. Mutexes, condition variables, and threads related to this are part of `PERFORMANCE_SCHEMA`.
- Galera internally uses monitor mechanism to enforce ordering of events. These monitor control events apply and are mainly responsible for the wait between different action. All such monitor mutexes and condition variables are covered as part of this implementation.
- There are lot of other miscellaneous action related to receiving of package and servicing messages. Mutexes and condition variables needed for them are now visible too. Threads that manage receiving and servicing are also being instrumented.

This feature has exposed all the important mutexes, condition variables that lead to lock/threads/files as part of this process.

Besides exposing file it also tracks write/read bytes like stats for file. These stats are not exposed for Galera files as Galera uses `mmap`.

Also, there are some threads that are short-lived and created only when needed especially for SST/IST purpose. They are also tracked but come into `PERFORMANCE_SCHEMA` tables only if/when they are created.

`Stage Info` from Galera specific function which server updates to track state of running thread is also visible in `PERFORMANCE_SCHEMA`.

7.9.1 What is not exposed ?

Galera uses customer data-structure in some cases (like STL structures). Mutexes used for protecting these structures which are not part of mainline Galera logic or doesn't fall in big-picture are not tracked. Same goes with threads that are `gcomm` library specific.

Galera maintains a process vector inside each monitor for its internal graph creation. This process vector is 65K in size and there are two such vectors per monitor. That is 128K * 3 = 384K condition variables. These are not tracked to avoid hogging `PERFORMANCE_SCHEMA` limits and sidelining of the main crucial information.

Percona and/or its affiliates © 2023 — Cookie Consent

7.9.2 Using `pxc_cluster_view`

The `pxc_cluster_view` - provides a unified view of the cluster. The table is in the Performance_Schema database.

```
DESCRIBE pxc_cluster_view;
```

This table has the following definition:

```
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| HOST_NAME   | char(64)     | NO   |     | NULL    |       |
| UUID        | char(36)     | NO   |     | NULL    |       |
| STATUS      | char(64)     | NO   |     | NULL    |       |
| LOCAL_INDEX | int unsigned | NO   |     | NULL    |       |
| SEGMENT     | int unsigned | NO   |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

To view the table, run the following query:

```
SELECT * FROM pxc_cluster_view;
```

The example of the output is the following:

```
+-----------+--------------------------------------+--------+-------------+---------+
| HOST_NAME | UUID                                 | STATUS | LOCAL_INDEX | SEGMENT |
+-----------+--------------------------------------+--------+-------------+---------+
| node1     | 22b9d47e-c215-11eb-81f7-7ed65a9d253b | SYNCED |           0 |       0 |
| node3     | 29c51cf5-c216-11eb-9101-1ba3a28e377a | SYNCED |           1 |       0 |
| node2     | 982cdb03-c215-11eb-9865-0ae076a59c5c | SYNCED |           2 |       0 |
+-----------+--------------------------------------+--------+-------------+---------+
3 rows in set (0.00 sec)
```

Last update: 2022-07-20

7.10 Data at Rest Encryption

This feature is considered **tech preview** quality.

7.10.1 Introduction

The “Data-at-rest” enables data at rest encryption of the InnoDB (file-per-table) tablespace by encrypting the physical database files. The data is automatically encrypted prior to writing to storage and automatically decrypted when read. If unauthorized users access the data files, they cannot read the contents. Data-in-transit can be encrypted using an SSL connection (details are available in the encrypt traffic documentation). Data-at-rest encryption is supported in Percona XtraDB Cluster for file-per-table tablespace and temporary files.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Note

The Percona Server for MySQL 5.7 data at rest encryption is similar to the MySQL 5.7 data-at-rest encryption. First, review the available encryption features for Percona Server for MySQL 5.7. Percona Server for MySQL 8.0 provides encryption features and options not available in the 5.7 version.

Feature	Status	GA Version
keyring_plugin	Generally Available, supported	`5.7.21-21`
File-Per_Table Tablespace	Generally Available, supported	`5.7.21-21`
Temporary Files	Generally Available, supported	`5.7.22-22`

About the keyring_file

The `keyring_file` stores an encryption key in a physical file. Specify the location of the file with the `keyring_file_data` parameter during startup. The following subsections cover some of the essential procedures for the `keyring_file` plugin.

CONFIGURATION

Percona XtraDB Cluster inherits the Percona Server for MySQL behavior to configure the `keyring_file` plugin. Install the plugin and add the following options in the configuration file:

```
[mysqld]
early-plugin-load=keyring_file.so
keyring_file_data=<PATH>/keyring
```

The `keyring_file` must be loaded using the `--early-plugin-load` option.

A `SHOW PLUGINS` statement can be used to check if the plugin has been successfully loaded.

Note

PXC recommends the same configuration on all cluster nodes, and all nodes should have the keyring configured. A JOINER node cannot join the cluster if there is a mismatch in the keyring configuration.

If the user has bootstrapped node with keyring enabled, then upcoming cluster nodes inherit the keyring (the encrypted key) from the DONOR node, in Percona XtraDB Cluster prior to 5.7.22, or generate the keyring, implemented in Percona XtraDB Cluster 5.7.22.

USAGE

The operations for a keyring are transactional. During write operations, the `keyring_file` plugin creates a backup file to ensure the operation can be rolled back if needed.

Prior to Percona XtraDB Cluster `5.7.22-29.26` the DONOR node had to send the keyring to the JOINER, because Percona XtraBackup backs up encrypted tablespaces. The JOINER must have the encryption key used by the DONOR to encrypt the tables to read these encrypted tablespaces. This restriction has been relaxed in Percona XtraDB Cluster 5.7.22 and now Percona XtraBackup re-encrypts the data using a transition-key and the JOINER re-encrypts the table using a generated master-key.

A keyring is sent from the DONOR to the JOINER as part of SST process (prior to Percona XtraDB Cluster 5.7.22) or is generated on the JOINER.

Percona and/or its affiliates © 2023 — Cookie Consent

The SST process can be done using xtrabackup (the recommended way), mysqldump or the rsync tool. In *xtrabackup* case, the keyring is sent before the data backup/streaming begins. The mysqldump utility uses a logical backup so it does not need to send keyring. The rsync tool syncs the keys when the data directories are synced.

Warning

The rsync tool does not provide a secure channel. A keyring sent using the rsync SST could be vulnerable to attack.

The recommended SST process uses xtrabackup. The user can configure a secure channel and the keyring is fully secured. In fact, xtrabackup does not allow a user to send the keyring if the SST channel is not secured.

To maintain data consistency, Percona XtraDB Cluster does not allow a combination of nodes with encryption and nodes without encryption. For example, a user creates node-1 with encryption (keyring) enabled and node-2 with encryption (keyring) disabled. A table created with encryption on node-1 fails on node-2, causing data inconsistency.

With Percona XtraDB Cluster `5.7.22-29.26`, a node will fail to start if it fails to load keyring plugin.

Note

If you do not specify the keyring parameters, the node does not know that it must load keyring. A JOINER node may start but eventually shutdown when a DML-level inconsistency with encrypted tablespace is detected.

If a node does not have an encrypted tablespace, the keyring is not generated and the keyring file is empty. The keyring is generated only when node uses an encrypted tablespace.

A user can rotate the key when needed. The `ALTER INSTANCE ROTATE INNODB MASTER KEY` statement is local to the node and is not replicated on cluster.

Starting from Percona XtraDB Cluster 5.7.22, the JOINER generates its keyring. In Percona XtraDB Cluster before 5.7.22 when JOINER joined the cluster, its keyring was the same as DONOR's keyring. The user could rotate the key if different keys for each node is part of the user's requirements (internal rules). Using different keys for each node is not necessary from the technical side, since all cluster nodes can continue operating with the same MASTER-key.

COMPATIBILITY

The keyring and the Percona XtraDB Cluster SST process is backward compatible. A higher-version JOINER can join from lower-version DONOR, but not vice-versa. More details are covered in the Upgrade and compatibility issues section.

Note

Percona XtraDB Cluster 5.6 does not have encrypted tablespaces. No major upgrade scenario for data-at-rest encryption is possible.

7.10.2 Configuring PXC to use keyring_vault plugin

keyring_vault

The `keyring_vault` plugin is supported starting from PXC 5.7.22. This plugin allows storing the master-key in vault-server.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Warning

The rsync tool does not support the `keyring_vault`. Any rysnc-SST on a joiner is aborted if the `keyring_vault` is configured.

CONFIGURATION

Configuration options are the same as upstream.

The `my.cnf` configuration file should contain the following options:

```
[mysqld]
early-plugin-load="keyring_vault=keyring_vault.so"
keyring_vault_config="<PATH>/keyring_vault_n1.conf"
```

Also `keyring_vault_n1.conf` file contents should be :

```
vault_url = http://127.0.0.1:8200
secret_mount_point = secret1
token = e0345eb4-35dd-3ddd-3b1e-e42bb9f2525d
vault_ca = /data/keyring_vault_confs/vault_ca.crt
```

The detailed description of these options can be found in the upstream documentation.

Vault-server is an external server, so make sure a PXC node can reach the server.

Note

Percona XtraDB Cluster recommends using the same keyring_plugin on all cluster nodes. Mixing keyring plugins is recommended only while transitioning from `keyring_file` to `keyring_vault` or vice-versa.

All nodes do not need to refer to same vault server. Whatever vault server is used, it should be accessible from the respective node. Also there is no restriction for all nodes to use the same mount point.

If the node is not able to reach/connect to the vault server, an error is notified during the server boot, and node refuses to start:

```
2018-05-29T03:54:33.859613Z 0 [Warning] Plugin keyring_vault reported:
'There is no vault_ca specified in keyring_vault's configuration file.
Please make sure that Vault's CA certificate is trusted by the machine
from which you intend to connect to Vault.'
2018-05-29T03:54:33.977145Z 0 [ERROR] Plugin keyring_vault reported:
'CURL returned this error code: 7 with error message : Failed to connect
to 127.0.0.1 port 8200: Connection refused'
```

If some nodes of the cluster are unable to connect to vault-server, this relates only to these specific nodes: e.g., if node-1 can connect, and node-2 cannot connect, only node-2 refuses to start. Also, if the server has pre-existing encrypted object and on reboot, the server fails to connect to vault-server, the object is not accessible.

In case when vault-server is accessible but authentication credential is incorrect, the consequences are the same, and the corresponding error looks like the following:

```
2018-05-29T03:58:54.461911Z 0 [Warning] Plugin keyring_vault reported:
'There is no vault_ca specified in keyring_vault's configuration file.
Please make sure that Vault's CA certificate is trusted by the machine
```


<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

```
from which you intend to connect to Vault.'
2018-05-29T03:58:54.577477Z 0 [ERROR] Plugin keyring_vault reported:
'Could not retrieve list of keys from Vault. Vault has returned the
following error(s): ["permission denied"]'
```

In case of an accessible vault-server with the wrong mount point, there is no error during server boot, but the node still refuses to start:

```
mysql> CREATE TABLE t1 (c1 INT, PRIMARY KEY pk(c1)) ENCRYPTION='Y';
```

The example of the output is the following:

```
ERROR 3185 (HY000): Can't find master key from keyring, please check keyring plugin is
loaded.

2018-05-29T04:01:33.774684Z 5 [ERROR] Plugin keyring_vault reported:
'Could not write key to Vault. Vault has returned the following error(s):
["no handler for route
'secret1/NDhfSU5OT0RCS2V5LTkzNzVmZWQ0LTVjZTQtMTFlOC05YTc3LTM0MDI4NmI4ODhiZS0xMF8='"]'
2018-05-29T04:01:33.774786Z 5 [ERROR] Plugin keyring_vault reported:
'Could not flush keys to keyring'
```

7.10.3 Mixing keyring plugins

With Percona XtraBackup introducing transition-key logic, it is now possible to mix and match keyring plugins. For example, the user has node-1 configured to use `keyring_file` plugin and node-2 configured to use `keyring_vault`.

Note

Percona recommends the same configuration for all the nodes of the cluster. A mix and match (in keyring plugins) is recommended only during transition from one type of keying to another.

Upgrade and compatibility issues

Percona XtraDB Cluster server before `5.7.22` only supported the `keyring_file` and the dependent Percona XtraBackup did not have the concept of transition-key. This makes the mix and match of old Percona XtraDB Cluster server (pre-5.7.21) using `keyring_file` with new Percona XtraDB Cluster server (post-5.7.22) using `keyring_vault` not possible. A user should first upgrade Percona XtraDB Cluster server to version 5.7.22 or newer using `keyring_file` plugin and then let it act as a DONOR to a new booting `keyring_vault` running the JOINER.

If all the nodes use Percona XtraDB Cluster 5.7.22, then the user can configure some nodes to use the `keyring_file` and other to use the `keyring_vault`, but this setup is not recommended and should be used during only during the transition to vault.

If all the nodes are using Percona XtraDB Cluster 5.7.21 and the user would like to use `keyring_vault` plugin, all the nodes should be upgraded to use Percona XtraDB Cluster 5.7.22 (that is where vault plugin support was introduced in PXC) or newer. Once all nodes are configured to use Percona XtraDB Cluster 5.7.22, users can switch one node to use `vault-plugin`.

Note

MySQL 5.7.21 supports migration between keystores. Migration requires a restart.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

7.10.4 InnoDB tablespace encryption

Percona XtraDB Cluster supports tablespace encryption for the file-per-table tablespace. File-per-tablespace encryption is a table or tablespace-specific feature and is enabled through DDL:

```
CREATE TABLE t1 (c1 INT, PRIMARY KEY pk(c1)) ENCRYPTION='Y';
CREATE TABLESPACE foo ADD DATAFILE 'foo.ibd' ENCRYPTION='Y';
```

The PXC cluster replicates the DDL statements and creates the encrypted table or tablespace on all cluster nodes.

This feature requires a keyring plugin to be loaded before it can be used. Percona XtraDB Cluster supports two types of keyring plugin: `keyring_file` and `keyring_vault`.

7.10.5 Temporary file encryption

Percona Server for MySQL 5.7.22 added support for encrypting temporary file storage enabled using `encrypt-tmp-files`. This storage or files are local to the node and has no direct effect on Percona XtraDB Cluster replication. Percona XtraDB Cluster recommends enabling it on all the cluster nodes, though the action is not mandatory. The parameter is the same as in Percona Server:

```
[mysqld]
encrypt-tmp-files=ON
```

7.10.6 Migrating Keys Between Keyring Keystores

Percona XtraDB Cluster supports key migration between keystores. The migration can be performed offline or online.

Offline Migration

In offline migration, the node to migrate is shutdown and the migration server takes care of migrating keys for the said server to a new keystore.

Following example illustrates this scenario:

1. Three Percona XtraDB Cluster nodes n1, n2, n3 - all using `keyring_file`, and n2 should be migrated to use `keyring_vault`
2. The user shuts down n2 node.
3. The user starts the Migration Server (`mysqld` with a special option).
4. The Migration Server copies keys from n2 keyring file and adds them to the vault server.
5. The user starts n2 node with the vault parameter, and keys should be available.

Here is how the migration server output should look like:

```
/dev/shm/pxc57/bin/mysqld --defaults-file=/dev/shm/pxc57/copy_mig.cnf \
--keyring-migration-source=keyring_file.so \
--keyring_file_data=/dev/shm/pxc57/node2/keyring \
--keyring-migration-destination=keyring_vault.so \
--keyring_vault_config=/dev/shm/pxc57/vault/keyring_vault.cnf &

2018-05-30T03:44:11.803459Z 0 [Warning] TIMESTAMP with implicit DEFAULT
value is deprecated. Please use
--explicit_defaults_for_timestamp server option (see documentation for
```


<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

```
more details).
2018-05-30T03:44:11.803534Z 0 [Note] --secure-file-priv is set to NULL.
Operations related to importing and
exporting data are disabled
2018-05-30T03:44:11.803550Z 0 [Warning] WSREP: Node is not a cluster node.
Disabling pxc_strict_mode
2018-05-30T03:44:11.803564Z 0 [Note] /dev/shm/pxc57/bin/mysqld
(mysqld 5.7.21-21-29.26-debug) starting as process
5710 ...
2018-05-30T03:44:11.805917Z 0 [Warning] Can't create test file /dev/shm/pxc57/copy_mig/
qaserver-06.lower-test
2018-05-30T03:44:11.805932Z 0 [Warning] Can't create test file /dev/shm/pxc57/copy_mig/
qaserver-06.lower-test
2018-05-30T03:44:11.945989Z 0 [Note] Keyring migration successful.
2018-05-30T03:44:11.946015Z 0 [Note] Binlog end
2018-05-30T03:44:11.946047Z 0 [Note] Shutting down plugin 'keyring_vault'
2018-05-30T03:44:11.946166Z 0 [Note] Shutting down plugin 'keyring_file'
2018-05-30T03:44:11.947334Z 0 [Note] /dev/shm/pxc57/bin/mysqld: Shutdown complete
```

The destination keystore receives additional migrated keys (pre-existing keys in destination keystore are not touched or removed) on successful migration. The source keystore continues to retain the keys as migration performs copy operation and not move operation.

If the migration fails, then the destination keystore is left untouched.

Online Migration

In online migration, node to migrate is kept running, and the migration server takes care of migrating keys for the said server to a new keystore by connecting to the node.

The following example illustrates this scenario:

1. Three Percona XtraDB Cluster nodes n1, n2, n3 - all using `keyring_file`, and n3 should be migrated to use `keyring_vault`
2. User starts the Migration Server (`mysqld` with a special option).
3. Migration Server copies keys from the n3 keyring file and adds them to the vault server.
4. The user restarts n3 node with the vault parameter, and keys should be available.

Here is how the migration server output should look like:

```
/dev/shm/pxc57/bin/mysqld --defaults-file=/dev/shm/pxc57/copy_mig.cnf \
--keyring-migration-source=keyring_vault.so \
--keyring_vault_config=/dev/shm/pxc57/keyring_vault3.cnf \
--keyring-migration-destination=keyring_file.so \
--keyring_file_data=/dev/shm/pxc57/node3/keyring \
--keyring-migration-host=localhost \
--keyring-migration-user=root \
--keyring-migration-port=16300 \
--keyring-migration-password='' &

2018-05-29T14:07:32.789673Z 0 [Warning] TIMESTAMP with implicit DEFAULT value is
deprecated. Please use
--explicit_defaults_for_timestamp server option (see documentation for more details).
2018-05-29T14:07:32.789748Z 0 [Note] --secure-file-priv is set to NULL. Operations related
to importing and
exporting data are disabled
2018-05-29T14:07:32.789766Z 0 [Warning] WSREP: Node is not a cluster node. Disabling
pxc_strict_mode
2018-05-29T14:07:32.789780Z 0 [Note] /dev/shm/pxc57/bin/mysqld (mysqld 5.7.21-21-29.26-
```


<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

```
debug) starting as process
4936 ...
2018-05-29T14:07:32.792036Z 0 [Warning] Can't create test file /dev/shm/pxc57/copy_mig/
qaserver-06.lower-test
2018-05-29T14:07:32.792052Z 0 [Warning] Can't create test file /dev/shm/pxc57/copy_mig/
qaserver-06.lower-test
2018-05-29T14:07:32.927612Z 0 [Note] Keyring migration successful.
2018-05-29T14:07:32.927636Z 0 [Note] Binlog end
2018-05-29T14:07:32.927671Z 0 [Note] Shutting down plugin 'keyring_vault'
2018-05-29T14:07:32.927793Z 0 [Note] Shutting down plugin 'keyring_file'
2018-05-29T14:07:32.928864Z 0 [Note] /dev/shm/pxc57/bin/mysqld: Shutdown complete
```

On a successful migration, the destination keystore has the additional migrated keys (the pre-existing keys in the destination keystore are not touched or removed). The source keystore continues to retain the keys as the migration performs copy operation and not move operation.

If the migration fails, then the destination keystore is left untouched.

Migration server options

- `--keyring-migration-source` : The source keyring plugin that manages the keys to be migrated.
- `--keyring-migration-destination` : The destination keyring plugin to which the migrated keys are to be copied

> **Note**
>
> For offline migration, no additional key migration options are needed.

- `--keyring-migration-host` : The host where the running server is located. This host is always the local host.
- `--keyring-migration-user` , `--keyring-migration-password` : The username and password for the account used to connect to the running server.
- `--keyring-migration-port` : Used for TCP/IP connections, the running server's port number used to connect.
- `--keyring-migration-socket` : Used for Unix socket file or Windows named pipe connections, the running server socket or named pipe used to connect.

Prerequisite for migration:

Make sure to pass required keyring options and other configuration parameters for the two keyring plugins. For example, if `keyring_file` is one of the plugins, you must set the `keyring_file_data` system variable if the keyring data file location is not the default location.

Other non-keyring options may be required as well. One way to specify these options is by using `--defaults-file` to name an option file that contains the required options.

```
[mysqld]
basedir=/dev/shm/pxc57
datadir=/dev/shm/pxc57/copy_mig
log-error=/dev/shm/pxc57/logs/copy_mig.err
socket=/tmp/copy_mig.sock
port=16400
```

Last update: 2022-07-20

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

7.11 Configure Systemd TimeoutStartSec

A node may fail to load with `$ systemctl start mysql.service` if the node has a large buffer pool or must synchronize a large amount of data.

A resolution is to either increase the setting for the `mysql` service `TimeoutStartSec` option to a higher number or disable the option.

Check the current timeout with the following:

```
$ systemctl show mysql.service -p TimeoutStartUSec
```

Expected output

```
TimeoutStartUSec=10min
```

The following command opens an editor and allows you to edit the value:

```
$ sudo systemctl edit mysql.service
```

In the text editor, add the following to overwrite the `TimeoutStartSec` configuration:

```
[Service]
TimeoutStartSec=900s
```

After systemctl is reloaded, the option value changes to 15 minutes.

Expected output

```
TimeoutStartUSec=15min
```

To disable the option, change the value to `0`:

```
[Service]
TimeoutStartSec=0
```

After systemctl is reloaded, the option is disabled.

Expected output

```
TimeoutStartUSec=infinity
```

For the edited change to take effect, run the following command:

```
$ sudo systemctl daemon-reload
```

After `systemctl` has been reloaded, verify your change with `systemctl show mysql.service -p TimeoutStartUSec`.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Last update: 2023-01-02

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

8. Flexibility

8.1 Binlogging and replication improvements

Due to continuous development, Percona Server incorporated a number of improvements related to replication and binary logs handling. This resulted in replication specifics, which distinguishes it from *MySQL*.

8.1.1 Temporary tables and mixed logging format

Summary of the fix:

As soon as some statement involving temporary table was met when using mixed binlog format, *MySQL* was switching to row-based logging of all statements the end of the session or until all temporary tables used in this session are dropped. It is inconvenient in case of long lasting connections, including replication-related ones. Percona Server fixes the situation by switching between statement-based and row-based logging as and when necessary.

Version Specific Information

- The fix has been ported form *Percona Server for MySQL* 5.6 in `5.7.10-1`.

Details:

Mixed binary logging format supported by Percona Server means that server runs in statement-based logging by default, but switches to row-based logging when replication would be unpredictable - in the case of a nondeterministic SQL statement that may cause data divergence if reproduced on a replica server. The switch is done upon any condition from the long list, and one of these conditions is the use of temporary tables.

Temporary tables are **never** logged using row-based format, but any statement, that touches a temporary table, is logged in row mode. This way all the side effects that temporary tables may produce on non-temporary ones are intercepted.

There is no need to use row logging format for any other statements solely because of the temp table presence. However *MySQL* was undertaking such an excessive precaution: once some statement with temporary table had appeared and the row-based logging was used, *MySQL* logged unconditionally all subsequent statements in row format.

Percona Server have implemented more accurate behavior: instead of switching to row-based logging until the last temporary table is closed, the usual rules of row vs statement format apply, and presence of currently opened temporary tables is no longer considered. This change was introduced with the fix of a bug #151 (upstream #72475).

8.1.2 Temporary table drops and binloging on GTID-enabled server

Summary of the fix:

MySQL logs DROP statements for all temporary tables irrelative of the logging mode under which these tables were created. This produces binlog writes and errand GTIDs on replicas with row and mixed logging. Percona Server fixes this by tracking the binlog format at temporary table create time and using it to decide whether a DROP should be logged or not.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Version Specific Information

- The fix has been ported form *Percona Server for MySQL* 5.6 in `5.7.17-11`.

Details:

Even with read_only mode enabled, the server permits some operations, including ones with temporary tables. With the previous fix, temporary table operations are not binlogged in row or mixed mode. But *MySQL* doesn't track what was the logging mode when temporary table was created, and therefore unconditionally logs `DROP` statements for all temporary tables. These `DROP` statements receive `IF EXISTS` addition, which is intended to make them harmless.

Percona Server have fixed this with the bug fixes #964, upstream #83003, and upstream #85258. Moreover, after all the binlogging fixes discussed so far nothing involving temporary tables is logged to binary log in row or mixed format, and so there is no need to consider `CREATE/DROP TEMPORARY TABLE` unsafe for use in stored functions, triggers, and multi-statement transactions in row/mixed format. Therefore an additional fix was introduced to mark creation and drop of temporary tables as unsafe inside transactions in statement-based replication only (bug fixed #1816, upstream #89467)).

8.1.3 Safety of statements with a `LIMIT` clause

Summary of the fix:

MySQL considers all `UPDATE/DELETE/INSERT ... SELECT` statements with `LIMIT` clause to be unsafe, no matter wether they are really producing non-deterministic result or not, and switches from statement-based logging to row-based one. Percona Server is more accurate, it acknowledges such instructions as safe when they include `ORDER BY PK` or `WHERE` condition. This fix has been ported from the upstream bug report #42415 (#44).

Version Specific Information

- The fix has been ported form *Percona Server for MySQL* 5.6 in `5.7.10.1`.

8.1.4 Performance improvement on relay log position update

Summary of the fix:

MySQL always updated relay log position in multi-source replications setups regardless of whether the committed transaction has already been executed or not. Percona Server omitts relay log position updates for the already logged GTIDs.

Version Specific Information

- The fix has been implemented in *Percona Server for MySQL* `5.7.18-14`.

Details

Particularly, such unconditional relay log position updates caused additional fsync operations in case of `relay-log-info-repository=TABLE`, and with the higher number of channels transmitting such duplicate (already executed) transactions the situation became proportionally worse. Bug fixed #1786 (upstream #85141).

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

8.1.5 Performance improvement on source and connection status updates

Summary of the fix:

Replica nodes configured to update source status and connection information only on log file rotation did not experience the expected reduction in load. *MySQL* was additionaly updating this information in case of multi-source replication when replica had to skip the already executed GTID event.

Version Specific Information

- The fix has been implemented in *Percona Server for MySQL* `5.7.20-19`.

Details

The configuration with `master_info_repository=TABLE` and `sync_master_info=0` makes replica to update source status and connection information in this table on log file rotation and not after each sync_master_info event, but it didn't work on multi-source replication setups. Heartbeats sent to the replica to skip GTID events which it had already executed previously, were evaluated as relay log rotation events and reacted with `mysql.slave_master_info` table sync. This inaccuracy could produce huge (up to 5 times on some setups) increase in write load on the replica, before this problem was fixed in Percona Server. Bug fixed #1812 (upstream #85158).

Last update: 2022-07-20

8.2 InnoDB Full-Text Search improvements

8.2.1 Ignoring Stopword list

By default all Full-Text Search indexes check the stopwords list, to see if any indexed elements *contain* one of the words on that list.

Using this list for n-gram indexes isn't always suitable, as an example, any item that contains `a` or `i` will be ignored. Another word that can't be searched is `east`, this one will find no matches because `a` is on the FTS stopword list.

To resolve this issue, in Percona Server for MySQL `5.7.20-18` a new `innodb_ft_ignore_stopwords` variable has been implemented which controls whether *InnoDB* Full-Text Search should ignore stopword list.

Although this variable is introduced to resolve n-gram issues, it affects all Full-Text Search indexes as well.

Being a stopword doesn't just mean to be a one of the predefined words from the list. Tokens shorter than innodb_ft_min_token_size or longer than innodb_ft_max_token_size are also considered stopwords. Therefore, when `innodb_ft_ignore_stopwords` is set to `ON` even for non-ngram FTS, `innodb_ft_min_token_size` / `innodb_ft_max_token_size` will be ignored meaning that in this case very short and very long words will also be indexed.

System Variables

`innodb_ft_ignore_stopwords`

Option	Description
Command Line:	Yes
Config File:	Yes

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Option	Description
Scope:	Session, Global
Dynamic:	Yes
Data type:	Boolean
Default Value:	`OFF`

When enabled, this variable will instruct *InnoDB* Full Text Search parser to ignore the stopword list when building/updating an FTS index.

8.2.2 Punctuation Marks in Full-Text Search

By default, full text search is unable to find words with various punctuation characters in boolean search mode, although those characters are indexed with ngram parser. A new variable `ft_query_extra_word_chars` was introduced in Percona Server `5.7.21-20` to solve this issue.

When it's enabled, all the non-whitespace symbols are considered to be word symbols by FTS query parser, except for the boolean search syntax symbols (which are specified by ft_boolean_syntax variable). The latter ones are also considered to be word symbols inside double quotes. This only applies for the query tokenizer, and the indexing tokenizer is not changed in any way. Because of this, the double quote symbol itself is never considered a word symbol, as no existing indexing tokenizer does so, thus searching for it would never return documents.

System Variables

`ft_query_extra_word_chars`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Session, Global
Dynamic:	Yes
Data type:	Boolean
Default Value:	`OFF`

When enabled, this variable will make all non-whitespace symbols (including punctuation marks) to be treated as word symbols in full-text search queries.

Last update: 2022-07-20

8.3 Multiple page asynchronous I/O requests

I/O unit size in *InnoDB* is only one page, even if doing read ahead. 16KB I/O unit size is too small for sequential reads, and much less efficient than larger I/O unit size.

InnoDB uses Linux asynchronous I/O (`aio`) by default. By submitting multiple consecutive 16KB read requests at once, Linux internally can merge requests and reads can be done more efficiently.

Percona and/or its affiliates © 2023 — Cookie Consent

On a HDD RAID 1+0 environment, more than 1000MB/s disk reads can be achieved by submitting 64 consecutive pages requests at once, while only 160MB/s disk reads is shown by submitting single page request.

With this feature *InnoDB* submits multiple page I/O requests.

8.3.1 Version Specific Information

The feauture has been ported from the *Facebook MySQL* patch in `5.7.20-18`.

8.3.2 Status Variables

`Innodb_buffered_aio_submitted`

Option	Description
Data type:	Numeric
Scope:	Global

This variable has been implemented in `5.7.20-18`. The variable shows the number of submitted buffered asynchronous I/O requests.

8.3.3 Other Reading

- Making full table scan 10x faster in InnoDB
- Bug #68659 InnoDB Linux native aio should submit more i/o requests at once

---

Last update: 2022-07-20

Percona and/or its affiliates © 2023 — Cookie Consent

9. Diagnostics

9.1 InnoDB Page Fragmentation Counters

InnoDB page fragmentation is caused by random insertion or deletion from a secondary index. This means that the physical ordering of the index pages on the disk is not same as the index ordering of the records on the pages. As a consequence this means that some pages take a lot more space and that queries which require a full table scan can take a long time to finish.

To provide more information about the *InnoDB* page fragmentation Percona Server now provides the following counters as status variables: `Innodb_scan_pages_contiguous`, `Innodb_scan_pages_disjointed`, `Innodb_scan_data_size`, `Innodb_scan_deleted_recs_size`, and `Innodb_scan_pages_total_seek_distance`.

9.1.1 Version Specific Information

The feature has been implemented in `5.7.20-18`.

9.1.2 Status Variables

`Innodb_scan_pages_contiguous`

Option	Description
Data type:	Numeric
Scope:	Session

This variable shows the number of contiguous page reads inside a query.

`Innodb_scan_pages_disjointed`

Option	Description
Data type:	Numeric
Scope:	Session

This variable shows the number of disjointed page reads inside a query.

`Innodb_scan_data_size`

Option	Description
Data type:	Numeric
Scope:	Session

This variable shows the size of data in all *InnoDB* pages read inside a query (in bytes) - calculated as the sum of `page_get_data_size(page)` for every page scanned.

Percona and/or its affiliates © 2023 — Cookie Consent

`Innodb_scan_deleted_recs_size`

Option	Description
Data type:	Numeric
Scope:	Session

This variable shows the size of deleted records (marked as `deleted` in `page_delete_rec_list_end()`) in all *InnoDB* pages read inside a query (in bytes) - calculated as the sum of `page_header_get_field(page, PAGE_GARBAGE)` for every page scanned.

`Innodb_scan_pages_total_seek_distance`

Option	Description
Data type:	Numeric
Scope:	Session

This variable shows the total seek distance when moving between pages.

9.1.3 Related Reading

- InnoDB: look after fragmentation
- Defragmenting a Table

Last update: 2022-07-20

9.2 Using libcoredumper

This feature is Tech Preview quality.

This feature was implemented in Percona Server 5.7.31-34 and has been tested against the supported operating systems for this version. The tool is experimental and may not be tested against newer operating systems in the future.

The documented moment of a computer when either the computer or an application crashed is a core dump file. Developers examine the dump as one of the tasks when searching for the cause of a failure.

The `libcoredumper` is a free and Open Source fork of `google-coredumper`, enhanced to work on newer Linux versions, and GCC and CLANG.

You should test before putting this tool into production.

Enabling the `libcoredumper`

Enable core dumps for troubleshooting purposes.

To enable the `libcoredumper`, add the `coredumper` variable to the `mysqld` section of `my.cnf`. This variable is independent of the older `core-file` variable.

The variable can have the following possible values:

Percona and/or its affiliates © 2023 — Cookie Consent

Value	Description
Blank	The core dump is saved under MySQL datadir and named `core`.
A path ending with /	The core dump is saved under the specified directory and named `core`.
Full path with a filename	The core dump is saved under the specified directory and with the specified filename

Restart the server.

Verifying the `libcoredumper` is Active

MySQL writes to the log when generating a core file and delegates the core dump operation to the Linux kernel. An example of the log message is the following:

```
Writing a core file
```

MySQL using the `libcoredumper` to generate the file creates the following message in the log:

```
Writing a core file using lib coredumper
```

Every core file adds a crash timestamp instead of a PID for the following reasons:

- Lets you correlate the core file with a crash. MySQL prints a UTC timestamp on the crash log.

```
10:02:09 UTC - mysqld got signal 11;
```

- Lets you keep multiple core files.

Note

For example, operators and containers run as PID 1. If the process ID identified the core file, each container crash generates a core dump that overwrites the previous core file.

Disabling the libcoredumper

You can disable the libcoredumper. A core file may contain sensitive data and takes disk space.

To disable the `libcoredumper` you must do the following:

- In the `mysqld` section of my.cnf, remove the `libcoredumper` variable.
- Restart the server.

Last update: 2022-07-20

9.3 Stack Trace

Developers use the stack trace in the debug process, either an interactive investigation or during the post-mortem. No configuration is required to generate a stack trace.

Implemented in Percona Server for MySQL 5.7.31-34, the stack trace adds the following:

Percona and/or its affiliates © 2023 — Cookie Consent

Name	Description
Prints binary BuildID	The Strip utility removes unneeded sections and debugging information to reduce the size. This method is standard with containers where the size of the image is essential. The BuildID lets you resolve the stack trace when the Strip utility removes the binary symbols table.
Print the server version information	The version information establishes the starting point for analysis. Some applications, such as MySQL, only print this information to a log on startup, and when the crash occurs, the size of the log may be large, rotated, or truncated.

Last update: 2022-07-20

Percona and/or its affiliates © 2023 — Cookie Consent

10. How-tos

10.1 Upgrading Percona XtraDB Cluster

This guide describes the procedure for upgrading Percona XtraDB Cluster without downtime (*rolling upgrade*) to the latest 5.7 version. A "rolling upgrade" means there is no need to take down the complete cluster during the upgrade.

Both major upgrades (from 5.6 to 5.7 version) and minor ones (from 5.7.x to 5.7.y) can be done in this way. Rolling upgrades to 5.7 from versions older than 5.6 are not supported. Therefore if you are running Percona XtraDB Cluster version 5.5, it is recommended to shut down all nodes, then remove and re-create the cluster from scratch. Alternatively, you can perform a rolling upgrade from PXC 5.5 to 5.6, and then follow the current procedure to upgrade from 5.6 to 5.7.

The following documents contain details about relevant changes in the 5.7 series of MySQL and Percona Server. Make sure you deal with any incompatible features and variables mentioned in these documents when upgrading to Percona XtraDB Cluster 5.7.

- Changed in Percona Server 5.7
- Upgrading MySQL
- Upgrading from MySQL 5.6 to 5.7

10.1.1 Major upgrade

To upgrade the cluster, follow these steps for each node:

1. Make sure that all nodes are synchronized.
2. Stop the `mysql` service:

```
$ sudo service mysql stop
```

3. Remove existing Percona XtraDB Cluster and Percona XtraBackup packages, then install Percona XtraDB Cluster version 5.7 packages. For more information, see Installing Percona XtraDB Cluster.

   For example, if you have Percona software repositories configured, you might use the following commands:

   - On CentOS or RHEL:

```
$ sudo yum remove percona-xtrabackup* Percona-XtraDB-Cluster*
$ sudo yum install Percona-XtraDB-Cluster-57
```

   - On Debian or Ubuntu:

```
$ sudo apt remove percona-xtrabackup* percona-xtradb-cluster*
$ sudo apt install percona-xtradb-cluster-57
```

4. In case of Debian or Ubuntu, the `mysql` service starts automatically after install. Stop the service:

```
$ sudo service mysql stop
```

5. Back up `grastate.dat`, so that you can restore it if it is corrupted or zeroed out due to network issue.

Percona and/or its affiliates © 2023 — Cookie Consent

6. Start the node outside the cluster (in standalone mode) by setting the `wsrep_provider` variable to `none`.

   For example:

   ```
   sudo mysqld --skip-grant-tables --user=mysql --wsrep-provider='none'
   ```

   Note

   As of Percona XtraDB Cluster 5.7.6, the `--skip-grant-tables` option is not required.

   Note

   To prevent any users from accessing this node while performing work on it, you may add –skip-networking to the startup options and use a local socket to connect, or alternatively you may want to divert any incoming traffic from your application to other operational nodes.

7. Open another session and run `mysql_upgrade`.
8. When the upgrade is done, stop the `mysqld` process. You can either run `sudo kill` on the `mysqld` process ID, or `sudo mysqladmin shutdown` with the MySQL root user credentials.

   Note

   On CentOS, the `my.cnf` configuration file is renamed to `my.cnf.rpmsave`. Make sure to rename it back before joining the upgraded node back to the cluster.

9. Now you can join the upgraded node back to the cluster.

   In most cases, starting the `mysql` service should run the node with your previous configuration:

   ```
   $ sudo service mysql start
   ```

   For more information, see Adding Nodes to Cluster.

   Note

   As of version 5.7, Percona XtraDB Cluster runs with PXC Strict Mode enabled by default. This will deny any unsupported operations and may halt the server upon encountering a failed validation.

   If you are not sure, it is recommended to first start the node with the `pxc_strict_mode` variable set to `PERMISSIVE` in the in the *MySQL* configuration file, `my.cnf`.

   After you check the log for any experimental or unsupported features and fix any encountered incompatibilities, you can set the variable back to `ENFORCING` at run time:

   ```
   mysql> SET pxc_strict_mode=ENFORCING;
   ```

   Also switch back to `ENFORCING` may be done by restarting the node with updated `my.cnf`.

10. Repeat this procedure for the next node in the cluster until you upgrade all nodes.

It is important that on rejoining, the node should synchronize using IST. For this, it is best not to leave the cluster node being upgraded offline for an extended period. More on this below.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

When performing any upgrade (major or minor), SST could be initiated by the joiner node after the upgrade if the server was offline for some time. After SST completes, the data directory structure needs to be upgraded (using mysql_upgrade) once more time to ensure compatibility with the newer version of binaries.

Note

In case of SST synchronization, the error log contains statements like "Check if state gap can be serviced using IST … State gap can't be serviced using IST. Switching to SST" instead of "Receiving IST: …" lines appropriate to IST synchronization.

10.1.2 Minor upgrade

To upgrade the cluster, follow these steps for each node:

1. Make sure that all nodes are synchronized.
2. Stop the `mysql` service:

```
$ sudo service mysql stop
```

3. Upgrade Percona XtraDB Cluster and Percona XtraBackup packages. For more information, see Installing Percona XtraDB Cluster.

   For example, if you have Percona software repositories configured, you might use the following commands:

   - On CentOS or RHEL:

```
$ sudo yum update Percona-XtraDB-Cluster-57
```

   - On Debian or Ubuntu:

```
$ sudo apt install --only-upgrade percona-xtradb-cluster-57
```

4. In case of Debian or Ubuntu, the `mysql` service starts automatically after install.

   Stop the service:

```
$ sudo service mysql stop
```

5. Back up `grastate.dat`, so that you can restore it if it is corrupted or zeroed out due to network issue.
6. Start the node outside the cluster (in standalone mode) by setting the `wsrep_provider` variable to `none`.

   For example:

```
sudo mysqld --skip-grant-tables --user=mysql --wsrep-provider='none'
```

Note

As of Percona XtraDB Cluster 5.7.6, the `--skip-grant-tables` option is not required.

Percona and/or its affiliates © 2023 — Cookie Consent

Note

To prevent any users from accessing this node while performing work on it, you may add –skip-networking to the startup options and use a local socket to connect, or alternatively you may want to divert any incoming traffic from your application to other operational nodes.

7. Open another session and run `mysql_upgrade`.
8. When the upgrade is done, stop the `mysqld` process. You can either run `sudo kill` on the `mysqld` process ID, or `sudo mysqladmin shutdown` with the MySQL root user credentials.

Note

On CentOS, the `my.cnf` configuration file is renamed to `my.cnf.rpmsave`. Make sure to rename it back before joining the upgraded node back to the cluster.

9. Now you can join the upgraded node back to the cluster.

   In most cases, starting the `mysql` service should run the node with your previous configuration:

   ```
   $ sudo service mysql start
   ```

   For more information, see Adding Nodes to Cluster.

Note

As of version 5.7, Percona XtraDB Cluster runs with PXC Strict Mode enabled by default. This will deny any unsupported operations and may halt the server upon encountering a failed validation.

If you are not sure, it is recommended to first start the node with the `pxc_strict_mode` variable set to `PERMISSIVE` in the in the *MySQL* configuration file, `my.cnf`.

After you check the log for any experimental or unsupported features and fix any encountered incompatibilities, you can set the variable back to `ENFORCING` at run time:

```
mysql> SET pxc_strict_mode=ENFORCING;
```

Also switch back to `ENFORCING` may be done by restarting the node with updated `my.cnf`.

10. Repeat this procedure for the next node in the cluster until you upgrade all nodes.

10.1.3 Dealing with IST/SST synchronization while upgrading

It is important that on rejoining, the node should synchronize using IST. For this, it is best not to leave the cluster node being upgraded offline for an extended period. More on this below.

When performing any upgrade (major or minor), SST could be initiated by the joiner node after the upgrade if the server was offline for some time. After SST completes, the data directory structure needs to be upgraded (using mysql_upgrade) once more time to ensure compatibility with the newer version of binaries.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Note

In case of SST synchronization, the error log contains statements like "Check if state gap can be serviced using IST … State gap can't be serviced using IST. Switching to SST" instead of "Receiving IST: …" lines appropriate to IST synchronization.

The following additional steps should be made to upgrade the data directory structure after SST (after the normal major or minor upgrade steps):

1. Shutdown the node that rejoined the cluster using SST:

```
$ sudo service mysql stop
```

2. Restart the node in standalone mode by setting the `wsrep_provider` variable to `none`, for example:

```
sudo mysqld --skip-grant-tables --user=mysql --wsrep-provider='none'
```

3. Run `mysql-upgrade`
4. Restart the node in cluster mode (e.g., by executing `sudo service mysql start` and make sure the cluster joins back using IST.

Last update: 2022-08-16

10.2 Crash Recovery

Unlike the standard MySQL replication, a Percona XtraDB Cluster cluster acts like one logical entity, which controls the status and consistency of each node as well as the status of the whole cluster. This allows maintaining the data integrity more efficiently than with traditional asynchronous replication without losing safe writes on multiple nodes at the same time.

However, there are scenarios where the database service can stop with no node being able to serve requests.

10.2.1 Scenario: Node A is gracefully stopped

In a three node cluster (node A, Node B, node C), one node (node A, for example) is gracefully stopped: for the purpose of maintenance, configuration change, etc.

In this case, the other nodes receive a "good bye" message from the stopped node and the cluster size is reduced; some properties like quorum calculation or auto increment are automatically changed. As soon as node A is started again, it joins the cluster based on its `wsrep_cluster_address` variable in `my.cnf`.

If the writeset cache (`gcache.size`) on nodes B and/or C still has all the transactions executed while node A was down, joining is possible via IST. If IST is impossible due to missing transactions in donor's gcache, the fallback decision is made by the donor and SST is started automatically.

10.2.2 Scenario: Two nodes are gracefully stopped

Similar to Scenario: Node A is gracefully stopped, the cluster size is reduced to 1 — even the single remaining node C forms the primary component and is able to serve client requests. To get the nodes back into the cluster, you just need to start them.

Percona and/or its affiliates © 2023 — Cookie Consent

However, when a new node joins the cluster, node C will be switched to the “Donor/Desynced” state as it has to provide the state transfer at least to the first joining node. It is still possible to read/write to it during that process, but it may be much slower, which depends on how large amount of data should be sent during the state transfer. Also, some load balancers may consider the donor node as not operational and remove it from the pool. So, it is best to avoid the situation when only one node is up.

If you restart node A and then node B, you may want to make sure note B does not use node A as the state transfer donor: node A may not have all the needed writesets in its gcache. Specify node C node as the donor in your configuration file and start the mysql service:

```
$ systemctl start mysql
```

See also

Galera Documentation: wsrep_sst_donor option

10.2.3 Scenario: All three nodes are gracefully stopped

The cluster is completely stopped and the problem is to initialize it again. It is important that a PXC node writes its last executed position to the `grastate.dat` file.

By comparing the seqno number in this file, you can see which is the most advanced node (most likely the last stopped). The cluster must be bootstrapped using this node, otherwise nodes that had a more advanced position will have to perform the full SST to join the cluster initialized from the less advanced one. As a result, some transactions will be lost). To bootstrap the first node, invoke the startup script like this on Debian or Ubuntu:

```
$ /etc/init.d/mysql bootstrap-pxc
```

If you are using RedHat or CentOS, use the following script:

```
$ systemctl start mysql@bootstrap.service
```

Note

Even though you bootstrap from the most advanced node, the other nodes have a lower sequence number. They will still have to join via the full SST because the *Galera Cache* is not retained on restart.

For this reason, it is recommended to stop writes to the cluster *before* its full shutdown, so that all nodes can stop at the same position. See also `pc.recovery`.

10.2.4 Scenario: One node disappears from the cluster

This is the case when one node becomes unavailable due to power outage, hardware failure, kernel panic, mysqld crash, **kill -9** on mysqld pid, etc.

Two remaining nodes notice the connection to node A is down and start trying to re-connect to it. After several timeouts, node A is removed from the cluster. The quorum is saved (2 out of 3 nodes are up), so no service disruption happens. After it is restarted, node A joins automatically (as described in Scenario: Node A is gracefully stopped).

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

10.2.5 Scenario: Two nodes disappear from the cluster

Two nodes are not available and the remaining node (node C) is not able to form the quorum alone. The cluster has to switch to a non-primary mode, where MySQL refuses to serve any SQL queries. In this state, the **mysqld** process on node C is still running and can be connected to but any statement related to data fails with an error

```
> SELECT * FROM test.sbtest1;
```

```
ERROR 1047 (08S01): WSREP has not yet prepared node for application use
```

Reads are possible until node C decides that it cannot access node A and node B. New writes are forbidden.

As soon as the other nodes become available, the cluster is formed again automatically. If node B and node C were just network-severed from node A, but they can still reach each other, they will keep functioning as they still form the quorum.

If node A and node B crashed, you need to enable the primary component on node C manually, before you can bring up node A and node B. The command to do this is:

```
> SET GLOBAL wsrep_provider_options='pc.bootstrap=true';
```

This approach only works if the other nodes are down before doing that! Otherwise, you end up with two clusters having different data.

See also

Adding Nodes to Cluster

10.2.6 Scenario: All nodes went down without a proper shutdown procedure

This scenario is possible in case of a datacenter power failure or when hitting a MySQL or Galera bug. Also, it may happen as a result of data consistency being compromised where the cluster detects that each node has different data. The `grastate.dat` file is not updated and does not contain a valid sequence number (seqno). It may look like this:

```
$ cat /var/lib/mysql/grastate.dat
# GALERA saved state
version: 2.1
uuid: 220dcdcb-1629-11e4-add3-aec059ad3734
seqno: -1
safe_to_bootstrap: 0
```

In this case, you cannot be sure that all nodes are consistent with each other. We cannot use safe_to_bootstrap variable to determine the node that has the last transaction committed as it is set to **0** for each node. An attempt to bootstrap from such a node will fail unless you start `mysqld` with the `--wsrep-recover` parameter:

```
$ mysqld --wsrep-recover
```

Search the output for the line that reports the recovered position after the node UUID (**1122** in this case):

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

```
...
... [Note] WSREP: Recovered position: 220dcdcb-1629-11e4-add3-aec059ad3734:1122
...
```

The node where the recovered position is marked by the greatest number is the best bootstrap candidate. In its `grastate.dat` file, set the safe_to_bootstrap variable to **1**. Then, bootstrap from this node.

Note

After a shutdown, you can boostrap from the node which is marked as safe in the `grastate.dat` file.

```
...
safe_to_bootstrap: 1
...
```

See also

Galera Documentation Introducing the Safe-To-Bootstrap feature in Galera Cluster

In recent Galera versions, the option `pc.recovery` (enabled by default) saves the cluster state into a file named `gvwstate.dat` on each member node. As the name of this option suggests (pc – primary component), it saves only a cluster being in the PRIMARY state. An example content of : file may look like this:

```
cat /var/lib/mysql/gvwstate.dat
my_uuid: 76de8ad9-2aac-11e4-8089-d27fd06893b9
#vwbeg
view_id: 3 6c821ecc-2aac-11e4-85a5-56fe513c651f 3
bootstrap: 0
member: 6c821ecc-2aac-11e4-85a5-56fe513c651f 0
member: 6d80ec1b-2aac-11e4-8d1e-b2b2f6caf018 0
member: 76de8ad9-2aac-11e4-8089-d27fd06893b9 0
#vwend
```

We can see a three node cluster with all members being up. Thanks to this new feature, the nodes will try to restore the primary component once all the members start to see each other. This makes the PXC cluster automatically recover from being powered down without any manual intervention! In the logs we will see:

10.2.7 Scenario: The cluster loses its primary state due to split brain

For the purpose of this example, let's assume we have a cluster that consists of an even number of nodes: six, for example. Three of them are in one location while the other three are in another location and they lose network connectivity. It is best practice to avoid such topology: if you cannot have an odd number of real nodes, you can use an additional arbitrator (garbd) node or set a higher pc.weight to some nodes. But when the split brain happens any way, none of the separated groups can maintain the quorum: all nodes must stop serving requests and both parts of the cluster will be continuously trying to re-connect.

If you want to restore the service even before the network link is restored, you can make one of the groups primary again using the same command as described in Scenario: Two nodes disappear from the cluster

```
> SET GLOBAL wsrep_provider_options='pc.bootstrap=true';
```


<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

After this, you are able to work on the manually restored part of the cluster, and the other half should be able to automatically re-join using IST as soon as the network link is restored.

> **Warning**
>
> If you set the bootstrap option on both the separated parts, you will end up with two living cluster instances, with data likely diverging away from each other. Restoring a network link in this case will not make them re-join until the nodes are restarted and members specified in configuration file are connected again.
>
> Then, as the Galera replication model truly cares about data consistency: once the inconsistency is detected, nodes that cannot execute row change statement due to a data difference – an emergency shutdown will be performed and the only way to bring the nodes back to the cluster is via the full SST

Based on material from Percona Database Performance Blog

This article is based on the blog post Galera replication - how to recover a PXC cluster by *Przemysław Malkowski*: https://www.percona.com/blog/2014/09/01/galera-replication-how-to-recover-a-pxc-cluster/

Last update: 2022-07-20

10.3 Configuring Percona XtraDB Cluster on CentOS

This tutorial describes how to install and configure three Percona XtraDB Cluster nodes on CentOS 6.8 servers, using the packages from Percona repositories.

- Node 1
  - Host name: `percona1`
  - IP address: `192.168.70.71`
- Node 2
  - Host name: `percona2`
  - IP address: `192.168.70.72`
- Node 3
  - Host name: `percona3`
  - IP address: `192.168.70.73`

10.3.1 Prerequisites

The procedure described in this tutorial requires the following:

- All three nodes have CentOS 6.8 installed.
- The firewall on all nodes is configured to allow connecting to ports 3306, 4444, 4567 and 4568.
- SELinux on all nodes is disabled.

10.3.2 Step 1. Installing PXC

Install Percona XtraDB Cluster on all three nodes as described in Installing Percona XtraDB Cluster on Red Hat Enterprise Linux and CentOS.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

10.3.3 Step 2. Configuring the first node

Individual nodes should be configured to be able to bootstrap the cluster. For more information about bootstrapping the cluster, see Bootstrapping the First Node.

1. Make sure that the configuration file `/etc/my.cnf` on the first node (`percona1`) contains the following:

```
[mysqld]

datadir=/var/lib/mysql
user=mysql

# Path to Galera library
wsrep_provider=/usr/lib64/libgalera_smm.so

# Cluster connection URL contains the IPs of node#1, node#2 and node#3
wsrep_cluster_address=gcomm://192.168.70.71,192.168.70.72,192.168.70.73

# In order for Galera to work correctly binlog format should be ROW
binlog_format=ROW

# MyISAM storage engine has only experimental support
default_storage_engine=InnoDB

# This InnoDB autoincrement locking mode is a requirement for Galera
innodb_autoinc_lock_mode=2

# Node 1 address
wsrep_node_address=192.168.70.71

# SST method
wsrep_sst_method=xtrabackup-v2

# Cluster name
wsrep_cluster_name=my_centos_cluster

# Authentication for SST method
wsrep_sst_auth="sstuser:s3cret"
```

2. Start the first node with the following command:

```
[root@percona1 ~]# /etc/init.d/mysql bootstrap-pxc
```

Note

In case you're running CentOS 7, the bootstrap service should be used instead:

```
[root@percona1 ~]# systemctl start mysql@bootstrap.service
```

The previous command will start the cluster with initial `wsrep_cluster_address` variable set to `gcomm://` . If the node or *MySQL* are restarted later, there will be no need to change the configuration file.

3. After the first node has been started, cluster status can be checked with the following command:

```
mysql> show status like 'wsrep%'
```


<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

The following output shows that the cluster has been successfully bootstrapped:

```
+----------------------------+--------------------------------------+
| Variable_name              | Value                                |
+----------------------------+--------------------------------------+
| wsrep_local_state_uuid     | c2883338-834d-11e2-0800-03c9c68e41ec |
...
| wsrep_local_state          | 4                                    |
| wsrep_local_state_comment  | Synced                               |
...
| wsrep_cluster_size         | 1                                    |
| wsrep_cluster_status       | Primary                              |
| wsrep_connected            | ON                                   |
...
| wsrep_ready                | ON                                   |
+----------------------------+--------------------------------------+
40 rows in set (0.01 sec)
```

Note

It is not recommended to leave an empty password for the root account. Password can be changed as follows:

```
mysql@percona1> UPDATE mysql.user SET password=PASSWORD("Passw0rd") where user='root';
mysql@percona1> FLUSH PRIVILEGES;
```

To perform State Snapshot Transfer using *XtraBackup*, set up a new user with proper privileges:

```
mysql@percona1> CREATE USER 'sstuser'@'localhost' IDENTIFIED BY 's3cret';
mysql@percona1> GRANT PROCESS, RELOAD, LOCK TABLES, REPLICATION CLIENT ON *.* TO
'sstuser'@'localhost';
mysql@percona1> FLUSH PRIVILEGES;
```

Note

MySQL root account can also be used for performing SST, but it is more secure to use a different (non-root) user for this.

10.3.4 Step 3. Configuring the second node

1. Make sure that the configuration file `/etc/my.cnf` on the second node (`percona2`) contains the following:

```
[mysqld]

datadir=/var/lib/mysql
user=mysql

# Path to Galera library
wsrep_provider=/usr/lib64/libgalera_smm.so

# Cluster connection URL contains IPs of node#1, node#2 and node#3
wsrep_cluster_address=gcomm://192.168.70.71,192.168.70.72,192.168.70.73
```


<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

```
# In order for Galera to work correctly binlog format should be ROW
binlog_format=ROW

# MyISAM storage engine has only experimental support
default_storage_engine=InnoDB

# This InnoDB autoincrement locking mode is a requirement for Galera
innodb_autoinc_lock_mode=2

# Node 2 address
wsrep_node_address=192.168.70.72

# Cluster name
wsrep_cluster_name=my_centos_cluster

# SST method
wsrep_sst_method=xtrabackup-v2

# Authentication for SST method
wsrep_sst_auth="sstuser:s3cret"
```

2. Start the second node with the following command:

```
[root@percona2 ~]# /etc/init.d/mysql start
```

3. After the server has been started, it should receive SST automatically. This means that the second node won't have empty root password anymore. In order to connect to the cluster and check the status, the root password from the first node should be used. Cluster status can be checked on both nodes. The following is an example of status from the second node (`percona2`):

```
mysql> show status like 'wsrep%'
```

The following output shows that the new node has been successfully added to the cluster:

```
+----------------------------+--------------------------------------+
| Variable_name              | Value                                |
+----------------------------+--------------------------------------+
| wsrep_local_state_uuid     | c2883338-834d-11e2-0800-03c9c68e41ec |
...
| wsrep_local_state          | 4                                    |
| wsrep_local_state_comment  | Synced                               |
...
| wsrep_cluster_size         | 2                                    |
| wsrep_cluster_status       | Primary                              |
| wsrep_connected            | ON                                   |
...
| wsrep_ready                | ON                                   |
+----------------------------+--------------------------------------+
40 rows in set (0.01 sec)
```

10.3.5 Step 4. Configuring the third node

1. Make sure that the MySQL configuration file `/etc/my.cnf` on the third node (`percona3`) contains the following:

```
[mysqld]
```


<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

```
datadir=/var/lib/mysql
user=mysql

# Path to Galera library
wsrep_provider=/usr/lib64/libgalera_smm.so

# Cluster connection URL contains IPs of node#1, node#2 and node#3
wsrep_cluster_address=gcomm://192.168.70.71,192.168.70.72,192.168.70.73

# In order for Galera to work correctly binlog format should be ROW
binlog_format=ROW

# MyISAM storage engine has only experimental support
default_storage_engine=InnoDB

# This InnoDB autoincrement locking mode is a requirement for Galera
innodb_autoinc_lock_mode=2

# Node #3 address
wsrep_node_address=192.168.70.73

# Cluster name
wsrep_cluster_name=my_centos_cluster

# SST method
wsrep_sst_method=xtrabackup-v2

# Authentication for SST method
wsrep_sst_auth="sstuser:s3cret"
```

2. Start the third node with the following command:

```
[root@percona3 ~]# /etc/init.d/mysql start
```

3. After the server has been started, it should receive SST automatically. Cluster status can be checked on all three nodes. The following is an example of status from the third node (`percona3`):

```
mysql> show status like 'wsrep%'
```

The following output confirms that the third node has joined the cluster:

```
+----------------------------+--------------------------------------+
| Variable_name              | Value                                |
+----------------------------+--------------------------------------+
| wsrep_local_state_uuid     | c2883338-834d-11e2-0800-03c9c68e41ec |
...
| wsrep_local_state          | 4                                    |
| wsrep_local_state_comment  | Synced                               |
...
| wsrep_cluster_size         | 3                                    |
| wsrep_cluster_status       | Primary                              |
| wsrep_connected            | ON                                   |
...
| wsrep_ready                | ON                                   |
+----------------------------+--------------------------------------+
40 rows in set (0.01 sec)
```


<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

10.3.6 Testing replication

To test replication, lets create a new database on second node, create a table for that database on the third node, and add some records to the table on the first node.

1. Create a new database on the second node:

```
mysql@percona2> CREATE DATABASE percona;
```

The following output confirms that a new database has been created:

```
Query OK, 1 row affected (0.01 sec)
```

2. Switch to a newly created database:

```
mysql@percona3> USE percona;
```

The following output confirms that a database has been changed:

```
Database changed
```

3. Create a table on the third node:

```
mysql@percona3> CREATE TABLE example (node_id INT PRIMARY KEY, node_name VARCHAR(30));
```

The following output confirms that a table has been created:

```
Query OK, 0 rows affected (0.05 sec)
```

4. Insert records on the first node:

```
mysql@percona1> INSERT INTO percona.example VALUES (1, 'percona1');
```

The following output confirms that the records have been inserted:

```
Query OK, 1 row affected (0.02 sec)
```

5. Retrieve all the rows from that table on the second node:

```
mysql@percona2> SELECT * FROM percona.example;
```

The following output confirms that all the rows have been retrieved:

```
+---------+-----------+
| node_id | node_name |
+---------+-----------+
|       1 | percona1  |
+---------+-----------+
1 row in set (0.00 sec)
```

This simple procedure should ensure that all nodes in the cluster are synchronized and working as intended.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Last update: 2022-07-20

10.4 Configuring Percona XtraDB Cluster on Ubuntu

This tutorial describes how to install and configure three Percona XtraDB Cluster nodes on Ubuntu 12.04.2 LTS servers, using the packages from Percona repositories.

- Node 1
- Host name: `pxc1`
- IP address: `192.168.70.61`
- Node 2
- Host name: `pxc2`
- IP address: `192.168.70.62`
- Node 3
- Host name: `pxc3`
- IP address: `192.168.70.63`

10.4.1 Prerequisites

The procedure described in this tutorial requires he following:

- All three nodes have Ubuntu 12.04.2 LTS installed.
- Firewall on all nodes is configured to allow connecting to ports 3306, 4444, 4567 and 4568.
- AppArmor profile for *MySQL* is disabled.

10.4.2 Step 1. Installing PXC

Install Percona XtraDB Cluster on all three nodes as described in Installing Percona XtraDB Cluster on Debian or Ubuntu.

Note

Debian/Ubuntu installation prompts for root password. For this tutorial, set it to `Passw0rd`. After the packages have been installed, `mysqld` will start automatically. Stop `mysqld` on all three nodes using `/etc/init.d/mysql stop`.

10.4.3 Step 2. Configuring the first node

Individual nodes should be configured to be able to bootstrap the cluster. For more information about bootstrapping the cluster, see Bootstrapping the First Node.

- Make sure that the configuration file `/etc/mysql/my.cnf` for the first node (`pxc1`) contains the following:

```
[mysqld]

datadir=/var/lib/mysql
user=mysql

# Path to Galera library
wsrep_provider=/usr/lib/libgalera_smm.so
```


Percona and/or its affiliates © 2023 — Cookie Consent

```
# Cluster connection URL contains the IPs of node#1, node#2 and node#3
wsrep_cluster_address=gcomm://192.168.70.61,192.168.70.62,192.168.70.63

# In order for Galera to work correctly, the binary log format should be ROW
binlog_format=ROW

# MyISAM storage engine has only experimental support
default_storage_engine=InnoDB

# This InnoDB autoincrement locking mode is a requirement for Galera
innodb_autoinc_lock_mode=2

# Node #1 address
wsrep_node_address=192.168.70.61

# SST method
wsrep_sst_method=xtrabackup-v2

# Cluster name
wsrep_cluster_name=my_ubuntu_cluster

# Authentication for SST method
wsrep_sst_auth="sstuser:s3cretPass"
```

- Start the first node with the following command:

```
[root@pxc1 ~]# /etc/init.d/mysql bootstrap-pxc
```

This command will start the first node and bootstrap the cluster.

- After the first node has been started, cluster status can be checked with the following command:

```
mysql> show status like 'wsrep%';
```

This output shows that the cluster has been successfully bootstrapped.

```
+----------------------------+--------------------------------------+
| Variable_name              | Value                                |
+----------------------------+--------------------------------------+
| wsrep_local_state_uuid     | b598af3e-ace3-11e2-0800-3e90eb9cd5d3 |
...
| wsrep_local_state          | 4                                    |
| wsrep_local_state_comment  | Synced                               |
...
| wsrep_cluster_size         | 1                                    |
| wsrep_cluster_status       | Primary                              |
| wsrep_connected            | ON                                   |
...
| wsrep_ready                | ON                                   |
+----------------------------+--------------------------------------+
40 rows in set (0.01 sec)
```

To perform State Snapshot Transfer using *XtraBackup*, set up a new user with proper privileges:

```
mysql@pxc1> CREATE USER 'sstuser'@'localhost' IDENTIFIED BY 's3cretPass';
mysql@pxc1> GRANT PROCESS, RELOAD, LOCK TABLES, REPLICATION CLIENT ON *.* TO
'sstuser'@'localhost';
mysql@pxc1> FLUSH PRIVILEGES;
```


<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Note

MySQL root account can also be used for performing SST, but it is more secure to use a different (non-root) user for this.

10.4.4 Step 3. Configuring the second node

- Make sure that the configuration file `/etc/mysql/my.cnf` on the second node (`pxc2`) contains the following:

```
[mysqld]

datadir=/var/lib/mysql
user=mysql

# Path to Galera library
wsrep_provider=/usr/lib/libgalera_smm.so

# Cluster connection URL contains IPs of node#1, node#2 and node#3
wsrep_cluster_address=gcomm://192.168.70.61,192.168.70.62,192.168.70.63

# In order for Galera to work correctly binlog format should be ROW
binlog_format=ROW

# MyISAM storage engine has only experimental support
default_storage_engine=InnoDB

# This InnoDB autoincrement locking mode is a requirement for Galera
innodb_autoinc_lock_mode=2

# Node #2 address
wsrep_node_address=192.168.70.62

# Cluster name
wsrep_cluster_name=my_ubuntu_cluster

# SST method
wsrep_sst_method=xtrabackup-v2

# Authentication for SST method
wsrep_sst_auth="sstuser:s3cretPass"
```

- Start the second node with the following command:

```
[root@pxc2 ~]# /etc/init.d/mysql start
```

- After the server has been started, it should receive SST automatically. Cluster status can now be checked on both nodes. The following is an example of status from the second node (`pxc2`):

```
mysql> show status like 'wsrep%';
```

This output shows that the new node has been successfully added to the cluster.

```
+----------------------------+--------------------------------------+
| Variable_name              | Value                                |
+----------------------------+--------------------------------------+
| wsrep_local_state_uuid     | b598af3e-ace3-11e2-0800-3e90eb9cd5d3 |
...
```


<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

```
| wsrep_local_state          | 4                                    |
| wsrep_local_state_comment  | Synced                               |
...
| wsrep_cluster_size         | 2                                    |
| wsrep_cluster_status       | Primary                              |
| wsrep_connected            | ON                                   |
...
| wsrep_ready                | ON                                   |
+----------------------------+--------------------------------------+
40 rows in set (0.01 sec)
```

10.4.5 Step 4. Configuring the third node

- Make sure that the MySQL configuration file `/etc/mysql/my.cnf` on the third node (`pxc3`) contains the following:

```
[mysqld]

datadir=/var/lib/mysql
user=mysql

# Path to Galera library
wsrep_provider=/usr/lib/libgalera_smm.so

# Cluster connection URL contains IPs of node#1, node#2 and node#3
wsrep_cluster_address=gcomm://192.168.70.61,192.168.70.62,192.168.70.63

# In order for Galera to work correctly binlog format should be ROW
binlog_format=ROW

# MyISAM storage engine has only experimental support
default_storage_engine=InnoDB

# This InnoDB autoincrement locking mode is a requirement for Galera
innodb_autoinc_lock_mode=2

# Node #3 address
wsrep_node_address=192.168.70.63

# Cluster name
wsrep_cluster_name=my_ubuntu_cluster

# SST method
wsrep_sst_method=xtrabackup-v2

#Authentication for SST method
wsrep_sst_auth="sstuser:s3cretPass"
```

- Start the third node with the following command:

```
[root@pxc3 ~]# /etc/init.d/mysql start
```

- After the server has been started, it should receive SST automatically. Cluster status can be checked on all nodes. The following is an example of status from the third node (`pxc3`):

```
mysql> show status like 'wsrep%';
```

This output confirms that the third node has joined the cluster.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

```
+----------------------------+--------------------------------------+
| Variable_name              | Value                                |
+----------------------------+--------------------------------------+
| wsrep_local_state_uuid     | b598af3e-ace3-11e2-0800-3e90eb9cd5d3 |
...
| wsrep_local_state          | 4                                    |
| wsrep_local_state_comment  | Synced                               |
...
| wsrep_cluster_size         | 3                                    |
| wsrep_cluster_status       | Primary                              |
| wsrep_connected            | ON                                   |
...
| wsrep_ready                | ON                                   |
+----------------------------+--------------------------------------+
40 rows in set (0.01 sec)
```

This output confirms that the third node has joined the cluster.

10.4.6 Testing replication

To test replication, lets create a new database on the second node, create a table for that database on the third node, and add some records to the table on the first node.

- Create a new database on the second node:

```
mysql@pxc2> CREATE DATABASE percona;
```

The following output confirms that a new database has been created:

```
Query OK, 1 row affected (0.01 sec)
```

- Switch to a newly created database:

```
mysql@percona3> USE percona;
```

The following output confirms that a database has been changed:

```
Database changed
```

- Create a table on the third node:

```
mysql@pxc3> CREATE TABLE example (node_id INT PRIMARY KEY, node_name VARCHAR(30));
```

The following output confirms that a table has been created:

```
Query OK, 0 rows affected (0.05 sec)
```

- Insert records on the first node:

```
mysql@percona1> INSERT INTO percona.example VALUES (1, 'percona1');
```

The following output confirms that the records have been inserted:

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

```
Query OK, 1 row affected (0.02 sec)
```

- Retrieve all the rows from that table on the second node:

```
mysql@percona2> SELECT * FROM percona.example;
```

The following output confirms that all the rows have been retrieved:

```
+---------+-----------+
| node_id | node_name |
+---------+-----------+
|       1 | percona1  |
+---------+-----------+
1 row in set (0.00 sec)
```

This simple procedure should ensure that all nodes in the cluster are synchronized and working as intended.

Last update: 2022-07-20

10.5 Setting up Galera Arbitrator

Galera Arbitrator is a member of *Percona XtraDB Cluster* that is used for voting in case you have a small number of servers (usually two) and don't want to add any more resources. Galera Arbitrator does not need a dedicated server. It can be installed on a machine running some other application. Just make sure it has good network connectivity.

Galera Arbitrator is a member of the cluster that participates in the voting, but not in actual replication (although it receives the same data as other nodes). Also, it is not included in flow control calculations.

This document will show how to add Galera Arbitrator node to an existing cluster.

Note

For more information on how to set up a cluster you can read in the Configuring Percona XtraDB Cluster on Ubuntu or Configuring Percona XtraDB Cluster on CentOS manuals.

10.5.1 Installation

Galera Arbitrator can be installed from Percona's repository by running:

```
root@ubuntu:~# apt install percona-xtradb-cluster-garbd-5.7
```

on Debian/Ubuntu distributions, or:

```
[root@centos ~]# yum install Percona-XtraDB-Cluster-garbd-57
```

on CentOS/RHEL distributions.

10.5.2 Configuration

To configure *Galera Arbitrator* on *Ubuntu/Debian* you need to edit the `/etc/default/garbd` file. On *CentOS/RHEL* configuration can be found in `/etc/sysconfig/garb` file.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Configuration file should look like this after installation:

```
# Copyright (C) 2012 Codership Oy
# This config file is to be sourced by garb service script.

# REMOVE THIS AFTER CONFIGURATION

# A comma-separated list of node addresses (address[:port]) in the cluster
# GALERA_NODES=""

# Galera cluster name, should be the same as on the rest of the nodes.
# GALERA_GROUP=""

# Optional Galera internal options string (e.g. SSL settings)
# see https://galeracluster.com/documentation-webpages/galeraparameters.html
# GALERA_OPTIONS=""

# Log file for garbd. Optional, by default logs to syslog
# Deprecated for CentOS7, use journalctl to query the log for garbd
# LOG_FILE=""
```

To set it up you'll need to add the information about the cluster you've set up. This example is using cluster information from the Configuring Percona XtraDB Cluster on Ubuntu.

```
# Copyright (C) 2012 Codership Oy
# This config file is to be sourced by garb service script.

# A comma-separated list of node addresses (address[:port]) in the cluster
GALERA_NODES="192.168.70.61:4567, 192.168.70.62:4567, 192.168.70.63:4567"

# Galera cluster name, should be the same as on the rest of the nodes.
GALERA_GROUP="my_ubuntu_cluster"

# Optional Galera internal options string (e.g. SSL settings)
# see https://galeracluster.com/documentation-webpages/galeraparameters.html
# GALERA_OPTIONS=""

# Log file for garbd. Optional, by default logs to syslog
# Deprecated for CentOS7, use journalctl to query the log for garbd
# LOG_FILE=""
```

Note

Please note that you need to remove the `# REMOVE THIS AFTER CONFIGURATION` line before you can start the service.

You can now start the *Galera Arbitrator* daemon (`garbd`) by running:

- On Debian or Ubuntu:

```
root@server:~# service garbd start
```

```
[ ok ] Starting /usr/bin/garbd: :.
```

- On Red Hat Enterprise Linux or CentOS:

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

```
root@server:~# service garb start
```

```
[ ok ] Starting /usr/bin/garbd: :.
```

You can additionally check the `arbitrator` status by running:

- On Debian or Ubuntu:

```
root@server:~# service garbd status
```

```
[ ok ] garb is running.
```

- On Red Hat Enterprise Linux or CentOS:

```
root@server:~# service garb status
```

```
[ ok ] garb is running.
```

Last update: 2022-08-23

10.6 How to set up a three-node cluster on a single box

This tutorial describes how to set up a 3-node cluster on a single physical box.

For the purposes of this tutorial, assume the following:

- The local IP address is `192.168.2.21`.
- Percona XtraDB Cluster is extracted from binary tarball into `/usr/local/Percona-XtraDB-Cluster-5.7.11-rel4beta-25.14.2.beta.Linux.x86_64`

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

To set up the cluster:

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

• Create three MySQL configuration files for the corresponding nodes:

• `/etc/my.4000.cnf`

```
[mysqld]
port = 4000
socket=/tmp/mysql.4000.sock
datadir=/data/bench/d1
basedir=/usr/local/Percona-XtraDB-Cluster-5.7.11-rel4beta-25.14.2.beta.Linux.x86_64
user=mysql
log_error=error.log
binlog_format=ROW
wsrep_cluster_address='gcomm://192.168.2.21:5030,192.168.2.21:6030'
wsrep_provider=/usr/local/Percona-XtraDB-Cluster-5.7.11-
rel4beta-25.14.2.beta.Linux.x86_64/lib/libgalera_smm.so
wsrep_sst_receive_address=192.168.2.21:4020
wsrep_node_incoming_address=192.168.2.21
wsrep_slave_threads=2
wsrep_cluster_name=trimethylxanthine
wsrep_provider_options = "gmcast.listen_addr=tcp://192.168.2.21:4030;"
wsrep_sst_method=rsync
wsrep_node_name=node4000
innodb_autoinc_lock_mode=2
```

• `/etc/my.5000.cnf`

```
[mysqld]
port = 5000
socket=/tmp/mysql.5000.sock
datadir=/data/bench/d2
basedir=/usr/local/Percona-XtraDB-Cluster-5.7.11-rel4beta-25.14.2.beta.Linux.x86_64
user=mysql
log_error=error.log
binlog_format=ROW
wsrep_cluster_address='gcomm://192.168.2.21:4030,192.168.2.21:6030'
wsrep_provider=/usr/local/Percona-XtraDB-Cluster-5.7.11-
rel4beta-25.14.2.beta.Linux.x86_64/lib/libgalera_smm.so
wsrep_sst_receive_address=192.168.2.21:5020
wsrep_node_incoming_address=192.168.2.21
wsrep_slave_threads=2
wsrep_cluster_name=trimethylxanthine
wsrep_provider_options = "gmcast.listen_addr=tcp://192.168.2.21:5030;"
wsrep_sst_method=rsync
wsrep_node_name=node5000
innodb_autoinc_lock_mode=2
```

• `/etc/my.6000.cnf`

```
[mysqld]
port = 6000
socket=/tmp/mysql.6000.sock
datadir=/data/bench/d3
basedir=/usr/local/Percona-XtraDB-Cluster-5.7.11-rel4beta-25.14.2.beta.Linux.x86_64
user=mysql
log_error=error.log
binlog_format=ROW
wsrep_cluster_address='gcomm://192.168.2.21:4030,192.168.2.21:5030'
wsrep_provider=/usr/local/Percona-XtraDB-Cluster-5.7.11-
rel4beta-25.14.2.beta.Linux.x86_64/lib/libgalera_smm.so
wsrep_sst_receive_address=192.168.2.21:6020
wsrep_node_incoming_address=192.168.2.21
```


<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

```
wsrep_slave_threads=2
wsrep_cluster_name=trimethylxanthine
wsrep_provider_options = "gmcast.listen_addr=tcp://192.168.2.21:6030;"
wsrep_sst_method=rsync
wsrep_node_name=node6000
innodb_autoinc_lock_mode=2
```

- Create three data directories for the nodes:
- `/data/bench/d1`
- `/data/bench/d2`
- `/data/bench/d3`
- Start the first node using the following command (from the Percona XtraDB Cluster install directory):

```
$ bin/mysqld_safe --defaults-file=/etc/my.4000.cnf --wsrep-new-cluster
```

If the node starts correctly, you should see the following output:

```
111215 19:01:49 [Note] WSREP: Shifting JOINED -> SYNCED (TO: 0)
111215 19:01:49 [Note] WSREP: New cluster view: global state:
4c286ccc-2792-11e1-0800-94bd91e32efa:0, view# 1: Primary, number of nodes: 1, my index: 0,
protocol version 1
```

To check the ports, run the following command:

```
$ netstat -anp | grep mysqld
```

The example of the output is the following:

```
tcp        0      0 192.168.2.21:4030           0.0.0.0:*                   LISTEN
21895/mysqld
tcp        0      0 0.0.0.0:4000                0.0.0.0:*                   LISTEN
21895/mysqld
```

- Start the second and third nodes:

```
bin/mysqld_safe --defaults-file=/etc/my.5000.cnf
bin/mysqld_safe --defaults-file=/etc/my.6000.cnf
```

If the nodes start and join the cluster successful, you should see the following output:

```
111215 19:22:26 [Note] WSREP: Shifting JOINER -> JOINED (TO: 2)
111215 19:22:26 [Note] WSREP: Shifting JOINED -> SYNCED (TO: 2)
111215 19:22:26 [Note] WSREP: Synchronized with group, ready for connections
```

To check the cluster size, run the following command:

```
$ mysql -h127.0.0.1 -P6000 -e "show global status like 'wsrep_cluster_size';"
```

The example of the output is the following:

```
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
```


<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

```
| wsrep_cluster_size | 3     |
+--------------------+-------+
```

After that you can connect to any node and perform queries, which will be automatically synchronized with other nodes. For example, to create a database on the second node, you can run the following command:

```
$ mysql -h127.0.0.1 -P5000 -e "CREATE DATABASE hello_peter"
```

Last update: 2022-07-20

10.7 How to set up a three-node cluster in EC2 environment

This manual assumes you are running three EC2 instances with Red Hat Enterprise Linux 7 64-bit.

- `node1`: `10.93.46.58`
- `node2`: `10.93.46.59`
- `node3`: `10.93.46.60`

Recommendations on launching EC2 instances

1. Select instance types that support Enhanced Networking functionality. Good network performance critical for synchronous replication used in Percona XtraDB Cluster.
2. When adding instance storage volumes, choose the ones with good I/O performance:
   - instances with NVMe are preferred
   - GP2 SSD are preferred to GP3 SSD volume types due to I/O latency
   - over sized GP2 SSD are preferred to IO1 volume types due to cost
3. Attach Elastic network interfaces with static IPs or assign Elastic IP addresses to your instances. Thereby private IP addresses are preserved on instances in case of reboot or restart. This is required as each Percona XtraDB Cluster member includes the `wsrep_cluster_address` option in its configuration which points to other cluster members.
4. Launch instances in different availability zones to avoid cluster downtime in case one of the zones experiences power loss or network connectivity issues.

> **See also**
>
> Amazon EC2 Documentation: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html

To set up Percona XtraDB Cluster:

1. Remove any Percona XtraDB Cluster 5.5, Percona Server 5.5, and Percona Server 5.6 packages.
2. Install Percona XtraDB Cluster as described in Installing Percona XtraDB Cluster on Red Hat Enterprise Linux and CentOS.
3. Create data directories:

```
mkdir -p /mnt/data
mysql_install_db --datadir=/mnt/data --user=mysql
```


<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

4. Stop the firewall service:

```
service iptables stop
```

> **Note**
>
> Alternatively, you can keep the firewall running, but open ports 3306, 4444, 4567, 4568. For example to open port 4567 on 192.168.0.1:
>
> ```
> iptables -A INPUT -i eth0 -p tcp -m tcp --source 192.168.0.1/24 --dport 4567 -j ACCEPT
> ```

5. Create `/etc/my.cnf` files:

   Contents of the configuration file on the first node:

```
[mysqld]
datadir=/mnt/data
user=mysql

binlog_format=ROW

wsrep_provider=/usr/lib64/libgalera_smm.so
wsrep_cluster_address=gcomm://10.93.46.58,10.93.46.59,10.93.46.60

wsrep_slave_threads=2
wsrep_cluster_name=trimethylxanthine
wsrep_sst_method=rsync
wsrep_node_name=node1

innodb_autoinc_lock_mode=2
```

   For the second and third nodes change the following lines:

```
wsrep_node_name=node2

wsrep_node_name=node3
```

6. Start and bootstrap Percona XtraDB Cluster on the first node:

```
[root@node1 ~]# /etc/init.d/mysql bootstrap-pxc
```

   You should see the following output:

```
2014-01-30 11:52:35 23280 [Note] /usr/sbin/mysqld: ready for connections.
Version: '5.6.15-56'  socket: '/var/lib/mysql/mysql.sock'  port: 3306  Percona XtraDB
Cluster (GPL), Release 25.3, Revision 706, wsrep_25.3.r4034
```

7. Start the second and third nodes:

```
[root@node2 ~]# /etc/init.d/mysql start
```

   You should see the following output:

```
2014-01-30 09:52:42 26104 [Note] WSREP: Flow-control interval: [28, 28]
2014-01-30 09:52:42 26104 [Note] WSREP: Restored state OPEN -> JOINED (2)
```


<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

```
2014-01-30 09:52:42 26104 [Note] WSREP: Member 2 (percona1) synced with group.
2014-01-30 09:52:42 26104 [Note] WSREP: Shifting JOINED -> SYNCED (TO: 2)
2014-01-30 09:52:42 26104 [Note] WSREP: New cluster view: global state:
4827a206-876b-11e3-911c-3e6a77d54953:2, view# 7: Primary, number of nodes: 3, my
index: 2, protocol version 2
2014-01-30 09:52:42 26104 [Note] WSREP: SST complete, seqno: 2
2014-01-30 09:52:42 26104 [Note] Plugin 'FEDERATED' is disabled.
2014-01-30 09:52:42 26104 [Note] InnoDB: The InnoDB memory heap is disabled
2014-01-30 09:52:42 26104 [Note] InnoDB: Mutexes and rw_locks use GCC atomic builtins
2014-01-30 09:52:42 26104 [Note] InnoDB: Compressed tables use zlib 1.2.3
2014-01-30 09:52:42 26104 [Note] InnoDB: Using Linux native AIO
2014-01-30 09:52:42 26104 [Note] InnoDB: Not using CPU crc32 instructions
2014-01-30 09:52:42 26104 [Note] InnoDB: Initializing buffer pool, size = 128.0M
2014-01-30 09:52:42 26104 [Note] InnoDB: Completed initialization of buffer pool
2014-01-30 09:52:43 26104 [Note] InnoDB: Highest supported file format is Barracuda.
2014-01-30 09:52:43 26104 [Note] InnoDB: 128 rollback segment(s) are active.
2014-01-30 09:52:43 26104 [Note] InnoDB: Waiting for purge to start
2014-01-30 09:52:43 26104 [Note] InnoDB:  Percona XtraDB (https://www.percona.com)
5.6.15-rel62.0 started; log sequence number 1626341
2014-01-30 09:52:43 26104 [Note] RSA private key file not found: /var/lib/mysql//
private_key.pem. Some authentication plugins will not work.
2014-01-30 09:52:43 26104 [Note] RSA public key file not found: /var/lib/mysql//
public_key.pem. Some authentication plugins will not work.
2014-01-30 09:52:43 26104 [Note] Server hostname (bind-address): '*'; port: 3306
2014-01-30 09:52:43 26104 [Note] IPv6 is available.
2014-01-30 09:52:43 26104 [Note]   - '::' resolves to '::';
2014-01-30 09:52:43 26104 [Note] Server socket created on IP: '::'.
2014-01-30 09:52:43 26104 [Note] Event Scheduler: Loaded 0 events
2014-01-30 09:52:43 26104 [Note] /usr/sbin/mysqld: ready for connections.
Version: '5.6.15-56'  socket: '/var/lib/mysql/mysql.sock'  port: 3306  Percona XtraDB
Cluster (GPL), Release 25.3, Revision 706, wsrep_25.3.r4034
2014-01-30 09:52:43 26104 [Note] WSREP: inited wsrep sidno 1
2014-01-30 09:52:43 26104 [Note] WSREP: wsrep_notify_cmd is not defined, skipping
notification.
2014-01-30 09:52:43 26104 [Note] WSREP: REPL Protocols: 5 (3, 1)
2014-01-30 09:52:43 26104 [Note] WSREP: Assign initial position for certification: 2,
protocol version: 3
2014-01-30 09:52:43 26104 [Note] WSREP: Service thread queue flushed.
2014-01-30 09:52:43 26104 [Note] WSREP: Synchronized with group, ready for connections
```

When all nodes are in SYNCED state, your cluster is ready.

8. You can try connecting to MySQL on any node and create a database:

```
$ mysql -uroot
> CREATE DATABASE hello_tom;
```

The new database will be propagated to all nodes.

Last update: 2022-08-23

10.8 Load balancing with HAProxy

This manual describes how to configure HAProxy to work with Percona XtraDB Cluster.

The following is an example of the configuration file for HAProxy:

```
# this config requires haproxy-1.4.20
```


<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

```
global
        log 127.0.0.1   local0
        log 127.0.0.1   local1 notice
        maxconn 4096
        uid 99
        gid 99
        daemon
        #debug
        #quiet

defaults
        log     global
        mode    http
        option  tcplog
        option  dontlognull
        retries 3
        redispatch
        maxconn 2000
        contimeout      5000
        clitimeout      50000
        srvtimeout      50000

listen mysql-cluster 0.0.0.0:3306
    mode tcp
    balance roundrobin
    option mysql-check user root

    server db01 10.4.29.100:3306 check
    server db02 10.4.29.99:3306 check
    server db03 10.4.29.98:3306 check
```

With this configuration, HAProxy will balance the load between three nodes. In this case, it only checks if `mysqld` listens on port 3306, but it doesn't take into an account the state of the node. So it could be sending queries to the node that has `mysqld` running even if it's in `JOINING` or `DISCONNECTED` state.

To check the current status of a node we need a more complex check. This idea was taken from codership-team google groups.

To implement this setup, you will need two scripts:

- **clustercheck** (located in `/usr/local/bin`) and a config for `xinetd`
- **mysqlchk** (located in `/etc/xinetd.d`) on each node

Both scripts are available in binaries and source distributions of Percona XtraDB Cluster.

Change the `/etc/services` file by adding the following line on each node:

```
mysqlchk        9200/tcp                # mysqlchk
```

The following is an example of the HAProxy configuration file in this case:

```
# this config needs haproxy-1.4.20

global
        log 127.0.0.1   local0
        log 127.0.0.1   local1 notice
        maxconn 4096
        uid 99
        gid 99
        #daemon
```


<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

```
        debug
        #quiet

defaults
        log     global
        mode    http
        option  tcplog
        option  dontlognull
        retries 3
        redispatch
        maxconn 2000
        contimeout      5000
        clitimeout      50000
        srvtimeout      50000

listen mysql-cluster 0.0.0.0:3306
    mode tcp
    balance roundrobin
    option  httpchk

    server db01 10.4.29.100:3306 check port 9200 inter 12000 rise 3 fall 3
    server db02 10.4.29.99:3306 check port 9200 inter 12000 rise 3 fall 3
    server db03 10.4.29.98:3306 check port 9200 inter 12000 rise 3 fall 3
```

Last update: 2022-07-20

10.9 Load balancing with ProxySQL

ProxySQL is a high-performance SQL proxy. ProxySQL runs as a daemon watched by a monitoring process. The process monitors the daemon and restarts it in case of a crash to minimize downtime.

The daemon accepts incoming traffic from *MySQL* clients and forwards it to backend *MySQL* servers.

The proxy is designed to run continuously without needing to be restarted. Most configuration can be done at runtime using queries similar to SQL statements. These include runtime parameters, server grouping, and traffic-related settings.

Note

For more information about ProxySQL, see ProxySQL documentation.

ProxySQL is available from the Percona software repositories in two versions. ProxySQL v1 does not natively support Percona XtraDB Cluster and requires custom bash scripts to keep track of the status of Percona XtraDB Cluster nodes using the ProxySQL scheduler.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

ProxySQL v2 natively supports Percona XtraDB Cluster. With this version, `proxysql-admin` tool does not require custom scripts to keep track of Percona XtraDB Cluster status.

- Using ProxySQL v1 with `proxysql-admin`
- Installing ProxySQL v1
- Automatic Configuration
  - Preparing Configuration File
  - Enabling ProxySQL
  - Disabling ProxySQL
  - Additional Options
  - ProxySQL Status script
- The **proxysql-admin** Tool with ProxySQL 2.0.x

10.9.1 Manual Configuration

This tutorial describes how to configure ProxySQL with three Percona XtraDB Cluster nodes.

Node	Host Name	IP address
Node 1	pxc1	192.168.70.61
Node 2	pxc2	192.168.70.62
Node 3	pxc3	192.168.70.63
Node 4	proxysql	192.168.70.64

ProxySQL can be configured either using the `/etc/proxysql.cnf` file or through the admin interface. Using the admin interface is preferable, because it allows you to change the configuration dynamically (without having to restart the proxy).

To connect to the ProxySQL admin interface, you need a `mysql` client. You can either connect to the admin interface from Percona XtraDB Cluster nodes that already have the `mysql` client installed (Node 1, Node 2, Node 3) or install the client on Node 4 and connect locally. For this tutorial, install Percona XtraDB Cluster on Node 4:

- On Debian or Ubuntu:

```
root@proxysql:~# apt install percona-xtradb-cluster-client-5.7
```

- On Red Hat Enterprise Linux or CentOS:

```
[root@proxysql ~]# yum install Percona-XtraDB-Cluster-client-57
```

To connect to the admin interface, use the credentials, host name and port specified in the global variables.

> **Warning**
>
> Do not use default credentials in production!

The following example shows how to connect to the ProxySQL admin interface with default credentials:

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

```
root@proxysql:~# mysql -u admin -padmin -h 127.0.0.1 -P 6032
```

The example of the output is the following:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.1.30 (ProxySQL Admin Module)

Copyright (c) 2009-2016 Percona LLC and/or its affiliates
Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql@proxysql>
```

To see the ProxySQL databases and tables use the following commands:

```
mysql@proxysql> SHOW DATABASES;
```

The example of the output is the following:

```
+-----+---------+-------------------------------+
| seq | name    | file                          |
+-----+---------+-------------------------------+
| 0   | main    |                               |
| 2   | disk    | /var/lib/proxysql/proxysql.db |
| 3   | stats   |                               |
| 4   | monitor |                               |
+-----+---------+-------------------------------+
4 rows in set (0.00 sec)
```

```
mysql@proxysql> SHOW TABLES;
```

The example of the output is the following:

```
+---------------------------------------+
| tables                                |
+---------------------------------------+
| global_variables                      |
| mysql_collations                      |
| mysql_query_rules                     |
| mysql_replication_hostgroups          |
| mysql_servers                         |
| mysql_users                           |
| runtime_global_variables              |
| runtime_mysql_query_rules             |
| runtime_mysql_replication_hostgroups  |
| runtime_mysql_servers                 |
| runtime_scheduler                     |
| scheduler                             |
+---------------------------------------+
12 rows in set (0.00 sec)
```


<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

For more information about admin databases and tables, see Admin Tables

Note

ProxySQL has 3 areas where the configuration can reside:

- MEMORY (your current working place)
- RUNTIME (the production settings)
- DISK (durable configuration, saved inside an SQLITE database)

When you change a parameter, you change it in MEMORY area. That is done by design to allow you to test the changes before pushing to production (RUNTIME), or save them to disk.

Adding cluster nodes to ProxySQL

To configure the backend Percona XtraDB Cluster nodes in ProxySQL, insert corresponding records into the `mysql_servers` table.

Note

ProxySQL uses the concept of *hostgroups* to group cluster nodes. This enables you to balance the load in a cluster by routing different types of traffic to different groups. There are many ways you can configure hostgroups (for example source and replicas, read and write load, etc.) and a every node can be a member of multiple hostgroups.

This example adds three Percona XtraDB Cluster nodes to the default hostgroup (`0`), which receives both write and read traffic:

```
mysql@proxysql> INSERT INTO mysql_servers(hostgroup_id, hostname, port) VALUES
(0,'192.168.70.61',3306);
mysql@proxysql> INSERT INTO mysql_servers(hostgroup_id, hostname, port) VALUES
(0,'192.168.70.62',3306);
mysql@proxysql> INSERT INTO mysql_servers(hostgroup_id, hostname, port) VALUES
(0,'192.168.70.63',3306);
```

To see the nodes:

```
mysql@proxysql> SELECT * FROM mysql_servers;
```

The example of the output is the following:

```
+--------------+---------------+------+--------+--------+-------------+-----------------
+---------------------+---------+----------------+---------+
| hostgroup_id | hostname      | port | status | weight | compression | max_connections |
max_replication_lag | use_ssl | max_latency_ms | comment |
+--------------+---------------+------+--------+--------+-------------+-----------------
+---------------------+---------+----------------+---------+
| 0            | 192.168.70.61 | 3306 | ONLINE | 1      | 0           | 1000            |
0                   | 0       | 0              |         |
| 0            | 192.168.70.62 | 3306 | ONLINE | 1      | 0           | 1000            |
0                   | 0       | 0              |         |
| 0            | 192.168.70.63 | 3306 | ONLINE | 1      | 0           | 1000            |
0                   | 0       | 0              |         |
+--------------+---------------+------+--------+--------+-------------+-----------------
```


<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

```
+--------------------+---------+---------------+---------+
3 rows in set (0.00 sec)
```

Creating ProxySQL Monitoring User

To enable monitoring of Percona XtraDB Cluster nodes in ProxySQL, create a user with `USAGE` privilege on any node in the cluster and configure the user in ProxySQL.

The following example shows how to add a monitoring user on Node 2:

```
mysql@pxc2> CREATE USER 'proxysql'@'%' IDENTIFIED BY 'ProxySQLPa55';
mysql@pxc2> GRANT USAGE ON *.* TO 'proxysql'@'%';
```

The following example shows how to configure this user on the ProxySQL node:

```
mysql@proxysql> UPDATE global_variables SET variable_value='proxysql'
             WHERE variable_name='mysql-monitor_username';
mysql@proxysql> UPDATE global_variables SET variable_value='ProxySQLPa55'
             WHERE variable_name='mysql-monitor_password';
```

To load this configuration at runtime, issue a `LOAD` command. To save these changes to disk (ensuring that they persist after ProxySQL shuts down), issue a `SAVE` command.

```
mysql@proxysql> LOAD MYSQL VARIABLES TO RUNTIME;
mysql@proxysql> SAVE MYSQL VARIABLES TO DISK;
```

To ensure that monitoring is enabled, check the monitoring logs:

```
mysql@proxysql> SELECT * FROM monitor.mysql_server_connect_log ORDER BY time_start_us DESC
LIMIT 6;
```

The example of the output is the following:

```
+---------------+------+------------------+----------------------+---------------+
| hostname      | port | time_start_us    | connect_success_time | connect_error |
+---------------+------+------------------+----------------------+---------------+
| 192.168.70.61 | 3306 | 1469635762434625 | 1695                 | NULL          |
| 192.168.70.62 | 3306 | 1469635762434625 | 1779                 | NULL          |
| 192.168.70.63 | 3306 | 1469635762434625 | 1627                 | NULL          |
| 192.168.70.61 | 3306 | 1469635642434517 | 1557                 | NULL          |
| 192.168.70.62 | 3306 | 1469635642434517 | 2737                 | NULL          |
| 192.168.70.63 | 3306 | 1469635642434517 | 1447                 | NULL          |
+---------------+------+------------------+----------------------+---------------+
6 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM monitor.mysql_server_ping_log ORDER BY time_start_us DESC LIMIT 6;
```

The example of the output is the following:

```
+---------------+------+------------------+-------------------+------------+
| hostname      | port | time_start_us    | ping_success_time | ping_error |
+---------------+------+------------------+-------------------+------------+
| 192.168.70.61 | 3306 | 1469635762416190 | 948               | NULL       |
| 192.168.70.62 | 3306 | 1469635762416190 | 803               | NULL       |
| 192.168.70.63 | 3306 | 1469635762416190 | 711               | NULL       |
| 192.168.70.61 | 3306 | 1469635702416062 | 783               | NULL       |
```


<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

```
| 192.168.70.62 | 3306 | 1469635702416062 | 631               | NULL       |
| 192.168.70.63 | 3306 | 1469635702416062 | 542               | NULL       |
+---------------+------+------------------+-------------------+------------+
6 rows in set (0.00 sec)
```

The previous examples show that ProxySQL is able to connect and ping the nodes you added.

To enable monitoring of these nodes, load them at runtime:

```
mysql@proxysql> LOAD MYSQL SERVERS TO RUNTIME;
```

Creating ProxySQL Client User

ProxySQL must have users that can access backend nodes to manage connections.

To add a user, insert credentials into `mysql_users` table:

```
mysql@proxysql> INSERT INTO mysql_users (username,password) VALUES ('sbuser','sbpass');
```

The example of the output is the following:

```
Query OK, 1 row affected (0.00 sec)
```

Note

ProxySQL currently doesn't encrypt passwords.

Load the user into runtime space and save these changes to disk (ensuring that they persist after ProxySQL shuts down):

```
mysql@proxysql> LOAD MYSQL USERS TO RUNTIME;
mysql@proxysql> SAVE MYSQL USERS TO DISK;
```

To confirm that the user has been set up correctly, you can try to log in:

```
root@proxysql:~# mysql -u sbuser -psbpass -h 127.0.0.1 -P 6033
```

The example of the output is the following:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1491
Server version: 5.1.30 (ProxySQL)

Copyright (c) 2009-2016 Percona LLC and/or its affiliates
Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```


Percona LLC and/or its affiliates © 2023 — Cookie Consent

To provide read/write access to the cluster for ProxySQL, add this user on one of the Percona XtraDB Cluster nodes:

```
mysql@pxc3> CREATE USER 'sbuser'@'192.168.70.64' IDENTIFIED BY 'sbpass';
```

```
Query OK, 0 rows affected (0.01 sec)

```sql
mysql@pxc3> GRANT ALL ON *.* TO 'sbuser'@'192.168.70.64';
```

```
Query OK, 0 rows affected (0.00 sec)
```

Adding Galera Support in ProxySQL v1

ProxySQL v2 supports monitoring the status Percona XtraDB Cluster nodes. ProxySQL v1 cannot detect a node which is not in `Synced` state. To monitor the status of Percona XtraDB Cluster nodes in ProxySQL v1, use the `proxysql_galera_checker` script. The script is located here: `/usr/bin/proxysql_galera_checker`.

To use this script, load it into ProxySQL v1 Scheduler.

The following example shows how you can load the script for default ProxySQL v1 configuration:

```
INSERT INTO scheduler (active,interval_ms,filename,arg1,comment)
VALUES (1,10000,'/usr/bin/proxysql_galera_checker','--config-file=/etc/proxysql-admin.cnf
--write-hg=10 --read-hg=11 --writer-count=1 --mode=singlewrite
--priority=192.168.100.20:3306,192.168.100.40:3306,192.168.100.10:3306,192.168.100.30:3306
--log=/var/lib/proxysql/cluster_one_proxysql_galera_check.log','cluster_one');
```

This scheduler script accepts the following options in the `arg1` argument:

Option	Name	Required	Description
`--config-file`	Configuration File	Yes	Specify `proxysql-admin` configuration file.
`--write-hg`	`HOSTGROUP WRITERS`	No	Specify ProxySQL write hostgroup.
`--read-hg`	`HOSTGROUP READERS`	No	Specify ProxySQL read hostgroup.
`--writer-count`	`NUMBER WRITERS`	No	Specify write nodes count. `0` for `loadbal` mode and `1` for `singlewrite` mode.
`--mode`	`MODE`	No	Specify ProxySQL read/write configuration mode.
`--priority`	`WRITER PRIORITY`	No	Specify write nodes priority.
`--log`	`LOG FILE`	No	Specify `proxysql_galera_checker` log file.

Note

Specify cluster name in comment column.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent
```

To load the scheduler changes into the runtime space:

```
mysql@proxysql> LOAD SCHEDULER TO RUNTIME;
```

To make sure that the script has been loaded, check the `runtime_scheduler` table:

```
mysql@proxysql> SELECT * FROM scheduler\G;
```

The example of the output is the following:

```
*************************** 1. row ***************************
         id: 1
     active: 1
interval_ms: 10000
   filename: /bin/proxysql_galera_checker
       arg1: --config-file=/etc/proxysql-admin.cnf --write-hg=10 --read-hg=11
             --writer-count=1 --mode=singlewrite
             --
priority=192.168.100.20:3306,192.168.100.40:3306,192.168.100.10:3306,192.168.100.30:3306
             --log=/var/lib/proxysql/cluster_one_proxysql_galera_check.log
       arg2: NULL
       arg3: NULL
       arg4: NULL
       arg5: NULL
    comment: cluster_one
1 row in set (0.00 sec)
```

To check the status of available nodes, run the following command:

```
mysql@proxysql> SELECT hostgroup_id,hostname,port,status FROM mysql_servers;
```

The example of the output is the following:

```
+--------------+---------------+------+--------+
| hostgroup_id | hostname      | port | status |
+--------------+---------------+------+--------+
| 0            | 192.168.70.61 | 3306 | ONLINE |
| 0            | 192.168.70.62 | 3306 | ONLINE |
| 0            | 192.168.70.63 | 3306 | ONLINE |
+--------------+---------------+------+--------+
3 rows in set (0.00 sec)
```

Note

Each node can have the following status:

- `ONLINE`: backend node is fully operational.
- `SHUNNED`: backend node is temporarily taken out of use, because either too many connection errors hapenned in a short time, or replication lag exceeded the allowed threshold.
- `OFFLINE_SOFT`: new incoming connections aren't accepted, while existing connections are kept until they become inactive. In other words, connections are kept in use until the current transaction is completed. This allows to gracefully detach a backend node.
- `OFFLINE_HARD`: existing connections are dropped, and new incoming connections aren't accepted. This is equivalent to deleting the node from a hostgroup, or temporarily taking it out of the hostgroup for maintenance.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Testing Cluster with sysbench

You can install `sysbench` from Percona software repositories:

- For Debian or Ubuntu:

```
root@proxysql:~# apt install sysbench
```

- For Red Hat Enterprise Linux or CentOS

```
[root@proxysql ~]# yum install sysbench
```

Note

`sysbench` requires ProxySQL client user credentials that you creted in Creating ProxySQL Client User.

- Create the database that will be used for testing on one of the Percona XtraDB Cluster nodes:

```
mysql@pxc1> CREATE DATABASE sbtest;
```

- Populate the table with data for the benchmark on the ProxySQL node:

```
root@proxysql:~# sysbench --report-interval=5 --num-threads=4 \
--num-requests=0 --max-time=20 \
--test=/usr/share/doc/sysbench/tests/db/oltp.lua \
--mysql-user='sbuser' --mysql-password='sbpass' \
--oltp-table-size=10000 --mysql-host=127.0.0.1 --mysql-port=6033 \
prepare
```

- Run the benchmark on the ProxySQL node:

```
root@proxysql:~# sysbench --report-interval=5 --num-threads=4 \
  --num-requests=0 --max-time=20 \
  --test=/usr/share/doc/sysbench/tests/db/oltp.lua \
  --mysql-user='sbuser' --mysql-password='sbpass' \
  --oltp-table-size=10000 --mysql-host=127.0.0.1 --mysql-port=6033 \
  run
```

ProxySQL stores collected data in the `stats` schema:

```
mysql@proxysql> SHOW TABLES FROM stats;
```

The example of the output is the following:

```
+--------------------------------+
| tables                         |
+--------------------------------+
| stats_mysql_query_rules        |
| stats_mysql_commands_counters  |
| stats_mysql_processlist        |
| stats_mysql_connection_pool    |
| stats_mysql_query_digest       |
| stats_mysql_query_digest_reset |
| stats_mysql_global             |
+--------------------------------+
```


<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

For example, to see the number of commands that run on the cluster:

```
mysql@proxysql> SELECT * FROM stats_mysql_commands_counters;
```

The example of the output is the following:

```
+-------------------+--------------+-----------+-----------+-----------+---------+---------
+----------+----------+-----------+-----------+--------+--------+---------+----------+
| Command           | Total_Time_us | Total_cnt | cnt_100us | cnt_500us | cnt_1ms | cnt_5ms
| cnt_10ms | cnt_50ms | cnt_100ms | cnt_500ms | cnt_1s | cnt_5s | cnt_10s | cnt_INFs |
+-------------------+--------------+-----------+-----------+-----------+---------+---------
+----------+----------+-----------+-----------+--------+--------+---------+----------+
| ALTER_TABLE       | 0             | 0         | 0         | 0         | 0       | 0
| 0        | 0        | 0         | 0         | 0      | 0      | 0       | 0        |
| ANALYZE_TABLE     | 0             | 0         | 0         | 0         | 0       | 0
| 0        | 0        | 0         | 0         | 0      | 0      | 0       | 0        |
| BEGIN             | 2212625       | 3686      | 55        | 2162      | 899     | 569
| 1        | 0        | 0         | 0         | 0      | 0      | 0       | 0        |
| CHANGE_MASTER     | 0             | 0         | 0         | 0         | 0       | 0
| 0        | 0        | 0         | 0         | 0      | 0      | 0       | 0        |
| COMMIT            | 21522591      | 3628      | 0         | 0         | 0       | 1765
| 1590     | 272      | 1         | 0         | 0      | 0      | 0       | 0        |
| CREATE_DATABASE   | 0             | 0         | 0         | 0         | 0       | 0
| 0        | 0        | 0         | 0         | 0      | 0      | 0       | 0        |
| CREATE_INDEX      | 0             | 0         | 0         | 0         | 0       | 0
| 0        | 0        | 0         | 0         | 0      | 0      | 0       | 0        |
...
| DELETE            | 2904130       | 3670      | 35        | 1546      | 1346    | 723
| 19       | 1        | 0         | 0         | 0      | 0      | 0       | 0        |
| DESCRIBE          | 0             | 0         | 0         | 0         | 0       | 0
| 0        | 0        | 0         | 0         | 0      | 0      | 0       | 0        |
...
| INSERT            | 19531649      | 3660      | 39        | 1588      | 1292    | 723
| 12       | 2        | 0         | 1         | 0      | 1      | 2       | 0        |
...
| SELECT            | 35049794      | 51605     | 501       | 26180     | 16606   | 8241
| 70       | 3        | 4         | 0         | 0      | 0      | 0       | 0        |
| SELECT_FOR_UPDATE | 0             | 0         | 0         | 0         | 0       | 0
| 0        | 0        | 0         | 0         | 0      | 0      | 0       | 0        |
...
| UPDATE            | 6402302       | 7367      | 75        | 2503      | 3020    | 1743
| 23       | 3        | 0         | 0         | 0      | 0      | 0       | 0        |
| USE               | 0             | 0         | 0         | 0         | 0       | 0
| 0        | 0        | 0         | 0         | 0      | 0      | 0       | 0        |
| SHOW              | 19691         | 2         | 0         | 0         | 0       | 0
| 1        | 1        | 0         | 0         | 0      | 0      | 0       | 0        |
| UNKNOWN           | 0             | 0         | 0         | 0         | 0       | 0
| 0        | 0        | 0         | 0         | 0      | 0      | 0       | 0        |
+-------------------+--------------+-----------+-----------+-----------+---------+---------
+----------+----------+-----------+-----------+--------+--------+---------+----------+
45 rows in set (0.00 sec)
```

Automatic Fail-over

ProxySQL will automatically detect if a node is not available or not synced with the cluster.

You can check the status of all available nodes by running:

```
mysql@proxysql> SELECT hostgroup_id,hostname,port,status FROM mysql_servers;
```


<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

The example of the output is the following:

```
+--------------+---------------+------+--------+
| hostgroup_id | hostname      | port | status |
+--------------+---------------+------+--------+
| 0            | 192.168.70.61 | 3306 | ONLINE |
| 0            | 192.168.70.62 | 3306 | ONLINE |
| 0            | 192.168.70.63 | 3306 | ONLINE |
+--------------+---------------+------+--------+
3 rows in set (0.00 sec)
```

To test problem detection and fail-over mechanism, shut down Node 3:

```
root@pxc3:~# service mysql stop
```

ProxySQL will detect that the node is down and update its status to `OFFLINE_SOFT`:

```
mysql@proxysql> SELECT hostgroup_id,hostname,port,status FROM mysql_servers;
```

The example of the output is the following:

```
+--------------+---------------+------+--------------+
| hostgroup_id | hostname      | port | status       |
+--------------+---------------+------+--------------+
| 0            | 192.168.70.61 | 3306 | ONLINE       |
| 0            | 192.168.70.62 | 3306 | ONLINE       |
| 0            | 192.168.70.63 | 3306 | OFFLINE_SOFT |
+--------------+---------------+------+--------------+
3 rows in set (0.00 sec)
```

Now start Node 3 again:

```
root@pxc3:~# service mysql start
```

The script will detect the change and mark the node as `ONLINE`:

```
mysql@proxysql> SELECT hostgroup_id,hostname,port,status FROM mysql_servers;
```

The example of the output is the following:

```
+--------------+---------------+------+--------+
| hostgroup_id | hostname      | port | status |
+--------------+---------------+------+--------+
| 0            | 192.168.70.61 | 3306 | ONLINE |
| 0            | 192.168.70.62 | 3306 | ONLINE |
| 0            | 192.168.70.63 | 3306 | ONLINE |
+--------------+---------------+------+--------+
3 rows in set (0.00 sec)
```

10.9.2 Assisted Maintenance Mode

Usually, to take a node down for maintenance, you need to identify that node, update its status in ProxySQL to `OFFLINE_SOFT`, wait for ProxySQL to divert traffic from this node, and then initiate the shutdown or perform maintenance tasks. Percona XtraDB Cluster includes a special *maintenance mode* for nodes that enables

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

you to take a node down without adjusting ProxySQL manually. This mode is controlled using the `pxc_maint_mode` variable, which is monitored by ProxySQL and can be set to one of the following values:

- `DISABLED`: This is the default state that tells ProxySQL to route traffic to the node as usual.
- `SHUTDOWN`: This state is set automatically when you initiate node shutdown.

You may need to shut down a node when upgrading the OS, adding resources, changing hardware parts, relocating the server, etc.

When you initiate node shutdown, Percona XtraDB Cluster does not send the signal immediately. Instead, it changes the state to `pxc_maint_mode=SHUTDOWN` and waits for a predefined period, which is determined by the value of the `pxc_maint_transition_period`. After detecting that the maintenance mode is set to `SHUTDOWN`, ProxySQL changes the status of this node to `OFFLINE_SOFT`, which stops creating new connections for the node. After the transition period ends, any long-running transactions that are still active are aborted.

- `MAINTENANCE`: You can change to this state if you need to perform maintenace on a node without shutting it down.

You may need to isolate the node for some time, so that it does not receive traffic from ProxySQL while you resize the buffer pool, truncate the undo log, defragment, or check disks, etc.

To do this, manually set `pxc_maint_mode=MAINTENANCE`. Control is not returned to the user for the predefined period set by `pxc-maint_transaction_period`, 10 seconds by default. ProxySQL marks the node as OFFLINE, and avoids opening new connections for any DML transactions. ProxySQL does not terminate existing connections.

Once control is returned, you can perform maintenance activity.

Note

Any data changes will still be replicated across the cluster.

After you finish maintenance, set the mode back to `DISABLED`. When ProxySQL detects this, it starts routing traffic to the node again.

Last update: 2023-01-27

10.10 Setting up PXC reference architecture with HAProxy

This manual describes how to set up Percona XtraDB Cluster in a virtualized test sandbox.

The procedure assumes Amazon EC2 micro instances running CentOS 6. However, it should apply to any virtualization technology (for example, VirtualBox) with any Linux distribution.

This manual requires three virtual machines for Percona XtraDB Cluster nodes, and one for HAProxy client, which redirects requests to the nodes. Running HAProxy on an application server, instead of having it as a dedicated entity, removes the unnecessary extra network roundtrip, because the load balancing layer in Percona XtraDB Cluster scales well with application servers.

1. Install Percona XtraDB Cluster on the three cluster nodes, as described in Installing Percona XtraDB Cluster on Red Hat Enterprise Linux and CentOS.
2. Install HAProxy and `sysbench` on the client node:

```
yum -y install haproxy sysbench
```


<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

3. Make sure that the `my.cnf` configuration file on the first node contains the following:

```
[mysqld]
server_id=1
binlog_format=ROW
log_bin=mysql-bin
wsrep_cluster_address=gcomm://
wsrep_provider=/usr/lib/libgalera_smm.so
datadir=/var/lib/mysql

wsrep_slave_threads=2
wsrep_cluster_name=pxctest
wsrep_sst_method=xtrabackup
wsrep_node_name=ip-10-112-39-98

log_slave_updates

innodb_autoinc_lock_mode=2
innodb_buffer_pool_size=400M
innodb_log_file_size=64M
```

4. Start the first node
5. Adjust the `my.cnf` configuration files on the second and third nodes to contain the same configuration settings, except the following:

   • Second node:

```
server_id=2
wsrep_cluster_address=gcomm://10.116.39.76
wsrep_node_name=ip-10-244-33-92
```

   • Third node:

```
server_id=3
wsrep_cluster_address=gcomm://10.116.39.76
wsrep_node_name=ip-10-194-10-179
```

Note

- `server_id` can be any unique number
- `wsrep_cluster_address` is the IP address of the first node
- `wsrep_node_name` can be any unique name, for example, the output of the `hostname` command

1. Start the second and third nodes.

   When a new node joins the cluster, SST is performed by taking a backup using XtraBackup, then copying it to the new node with `netcat`. After a successful SST, you should see the following in the error log:

```
120619 13:20:17 [Note] WSREP: State transfer required:
     Group state: 77c9da88-b965-11e1-0800-ea53b7b12451:97
     Local state: 00000000-0000-0000-0000-000000000000:-1
120619 13:20:17 [Note] WSREP: New cluster view: global state: 77c9da88-b965-11e1-0800-
ea53b7b12451:97, view# 18: Primary, number of nodes: 3, my index: 0, protocol version
2
```


<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

```
120619 13:20:17 [Warning] WSREP: Gap in state sequence. Need state transfer.
120619 13:20:19 [Note] WSREP: Running: 'wsrep_sst_xtrabackup 'joiner'
'10.195.206.117' '' '/var/lib/mysql/' '/etc/my.cnf' '20758' 2>sst.err'
120619 13:20:19 [Note] WSREP: Prepared |SST| request: xtrabackup|10.195.206.117:4444/
xtrabackup_sst
120619 13:20:19 [Note] WSREP: wsrep_notify_cmd is not defined, skipping notification.
120619 13:20:19 [Note] WSREP: Assign initial position for certification: 97, protocol
version: 2
120619 13:20:19 [Warning] WSREP: Failed to prepare for incremental state transfer:
Local state UUID (00000000-0000-0000-0000-000000000000) does not match group state
UUID (77c9da88-b965-11e1-0800-ea53b7b12451): 1 (Operation not permitted)
     at galera/src/replicator_str.cpp:prepare_for_IST():439. IST will be unavailable.
120619 13:20:19 [Note] WSREP: Node 0 (ip-10-244-33-92) requested state transfer from
'*any*'. Selected 1 (ip-10-112-39-98)(SYNCED) as donor.
120619 13:20:19 [Note] WSREP: Shifting PRIMARY -> JOINER (TO: 102)
120619 13:20:19 [Note] WSREP: Requesting state transfer: success, donor: 1
120619 13:20:59 [Note] WSREP: 1 (ip-10-112-39-98): State transfer to 0
(ip-10-244-33-92) complete.
120619 13:20:59 [Note] WSREP: Member 1 (ip-10-112-39-98) synced with group.
120619 13:21:17 [Note] WSREP: |SST| complete, seqno: 105
120619 13:21:17 [Note] Plugin 'FEDERATED' is disabled.
120619 13:21:17 InnoDB: The InnoDB memory heap is disabled
120619 13:21:17 InnoDB: Mutexes and rw_locks use GCC atomic builtins
120619 13:21:17 InnoDB: Compressed tables use zlib 1.2.3
120619 13:21:17 InnoDB: Using Linux native AIO
120619 13:21:17 InnoDB: Initializing buffer pool, size = 400.0M
120619 13:21:17 InnoDB: Completed initialization of buffer pool
120619 13:21:18 InnoDB: highest supported file format is Barracuda.
120619 13:21:18  InnoDB: Waiting for the background threads to start
120619 13:21:19 Percona XtraDB (https://www.percona.com) 1.1.8-rel25.3 started; log
sequence number 246661644
120619 13:21:19 [Note] Recovering after a crash using mysql-bin
120619 13:21:19 [Note] Starting crash recovery...
120619 13:21:19 [Note] Crash recovery finished.
120619 13:21:19 [Note] Server hostname (bind-address): '(null)'; port: 3306
120619 13:21:19 [Note]   - '(null)' resolves to '0.0.0.0';
120619 13:21:19 [Note]   - '(null)' resolves to '::';
120619 13:21:19 [Note] Server socket created on IP: '0.0.0.0'.
120619 13:21:19 [Note] Event Scheduler: Loaded 0 events
120619 13:21:19 [Note] WSREP: Signalling provider to continue.
120619 13:21:19 [Note] WSREP: Received |SST|: 77c9da88-b965-11e1-0800-ea53b7b12451:105
120619 13:21:19 [Note] WSREP: |SST| received: 77c9da88-b965-11e1-0800-ea53b7b12451:105
120619 13:21:19 [Note] WSREP: 0 (ip-10-244-33-92): State transfer from 1
(ip-10-112-39-98) complete.
120619 13:21:19 [Note] WSREP: Shifting JOINER -> JOINED (TO: 105)
120619 13:21:19 [Note] /usr/sbin/mysqld: ready for connections.
Version: '5.5.24-log'  socket: '/var/lib/mysql/mysql.sock'  port: 3306  Percona
XtraDB Cluster (GPL), wsrep_23.6.r340
120619 13:21:19 [Note] WSREP: Member 0 (ip-10-244-33-92) synced with group.
120619 13:21:19 [Note] WSREP: Shifting JOINED -> SYNCED (TO: 105)
120619 13:21:20 [Note] WSREP: Synchronized with group, ready for connections
```

For debugging information about the SST, you can check the `sst.err` file and the error log.

After SST finishes, you can check the cluster size as follows:

```
mysql> show global status like 'wsrep_cluster_size';
```

The example of the output is the following:

```
+--------------------+-------+
| Variable_name      | Value |
```


<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

```
+--------------------+-------+
| wsrep_cluster_size | 3     |
+--------------------+-------+
1 row in set (0.00 sec)
```

2. When all cluster nodes are started, configure HAProxy on the client node. This will enable the application to connect to localhost as if it were a single MySQL server, instead of a Percona XtraDB Cluster node.

   You can configure HAProxy to connect and write to all cluster nodes or to one node at a time. The former method can lead to rollbacks due to conflicting writes when optimistic locking at commit time is triggered, while the latter method avoids rollbacks.

   However, most good applications should be able to handle rollbacks, so either method is fine in this case.

   To configure HAProxy, add the following to `/etc/haproxy/haproxy.cfg`:

```
global
log 127.0.0.1 local0
log 127.0.0.1 local1 notice
maxconn 4096
chroot /usr/share/haproxy
user haproxy
group haproxy
daemon

defaults
log global
mode http
option tcplog
option dontlognull
retries 3
option redispatch
maxconn 2000
contimeout 5000
clitimeout 50000
srvtimeout 50000

frontend pxc-front
bind *:3307
mode tcp
default_backend pxc-back

frontend stats-front
bind *:80
mode http
default_backend stats-back

frontend pxc-onenode-front
bind *:3306
mode tcp
default_backend pxc-onenode-back

backend pxc-back
mode tcp
balance leastconn
option httpchk
server c1 10.116.39.76:3306 check port 9200 inter 12000 rise 3 fall 3
server c2 10.195.206.117:3306 check port 9200 inter 12000 rise 3 fall 3
server c3 10.202.23.92:3306 check port 9200 inter 12000 rise 3 fall 3

backend stats-back
```


<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

```
mode http
balance roundrobin
stats uri /haproxy/stats
stats auth pxcstats:secret

backend pxc-onenode-back
mode tcp
balance leastconn
option httpchk
server c1 10.116.39.76:3306 check port 9200 inter 12000 rise 3 fall 3
server c2 10.195.206.117:3306 check port 9200 inter 12000 rise 3 fall 3 backup
server c3 10.202.23.92:3306 check port 9200 inter 12000 rise 3 fall 3 backup
```

In this configuration, three frontend-backend pairs are defined:

- The `stats` pair is for HAProxy statistics page (port 80).

You can access it at `/haproxy/stats` using the credential specified in the `stats auth` parameter.

- The `pxc` pair is for connecting to all three nodes (port 3307).

In this case, the *leastconn* load balancing method is used, instead of round-robin, which means connection is made to the backend with the least connections established.

- The `pxc-onenode` pair is for connecting to one node at a time (port 3306) to avoid rollbacks because of optimistic locking.

If the node goes offline, HAProxy will connect to another one.

Percona LLC and/or its affiliates © 2023 — Cookie Consent

Note

MySQL is checked via `httpchk`. MySQL will not serve these requests by default. You have to set up the `clustercheck` utility, which is distributed with Percona XtraDB Cluster. This will enable HAProxy to check MySQL via HTTP.

The `clustercheck` script is a simple shell script that accepts HTTP requests and checks the node via the `wsrep_local_state` variable. If the node's status is fine, it will send a response with HTTP code `200 OK`. Otherwise, it sends `503`.

To create the `clustercheck` user, run the following:

```
mysql> grant process on *.* to 'clustercheckuser'@'localhost' identified by
'clustercheckpassword!';
```

```
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> flush privileges;
```

```
Query OK, 0 rows affected (0.00 sec)
```

If you want to use a different user name or password, you have to modify them in the `clustercheck` script.

If you run the script on a running node, you should see the following:

```
# clustercheck
HTTP/1.1 200 OK

Content-Type: Content-Type: text/plain
```

You can use `xinetd` to daemonize the script. If xinetd is not installed, you can install it with `yum`:

```
# yum -y install xinetd
```

The service is configured in `/etc/xinetd.d/mysqlchk`:

```
# default: on
# description: mysqlchk
service mysqlchk
{
# this is a config for xinetd, place it in /etc/xinetd.d/
disable = no
flags = REUSE
socket_type = stream
port = 9200
wait = no
user = nobody
server = /usr/bin/clustercheck
log_on_failure += USERID
only_from = 0.0.0.0/0
# recommended to put the IPs that need
# to connect exclusively (security purposes)
per_source = UNLIMITED
}
```

Add the new service to `/etc/services`:

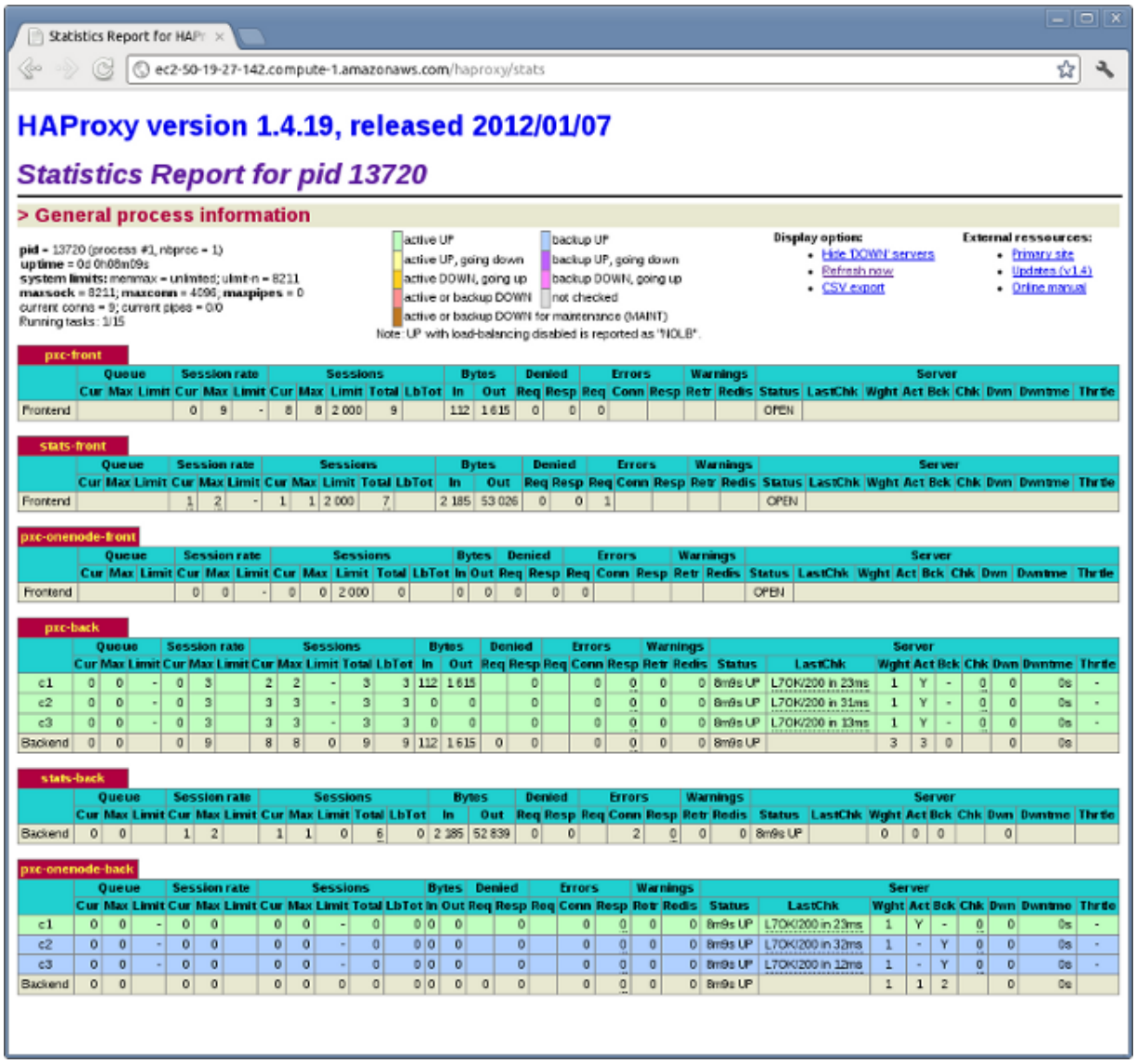
```
mysqlchk 9200/tcp # mysqlchk
```

Clustercheck will now listen on port 9200 after `xinetd` restarts and HAProxy is ready to check MySQL via HTTP:

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

```
# service xinetd restart
```

If you did everything correctly, the statistics page for HAProxy should look like this:



10.10.1 Testing the cluster with sysbench

After you set up Percona XtraDB Cluster in a sand box, you can test it using sysbench. This example shows how to do it with `sysbench` from the EPEL repository.

1. Create a database and a user for `sysbench`:

```
mysql> create database sbtest;
```

```
Query OK, 1 row affected (0.01 sec)
```

```
mysql> grant all on sbtest.* to 'sbtest'@'%' identified by 'sbpass';
```


<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

```
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> flush privileges;
```

```
Query OK, 0 rows affected (0.00 sec)
```

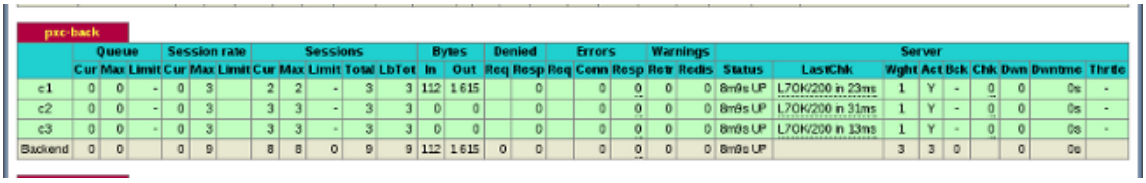
2. Populate the table with data for the benchmark:

```
sysbench --test=oltp --db-driver=mysql --mysql-engine-trx=yes --mysql-table-
engine=innodb --mysql-host=127.0.0.1 --mysql-port=3307 --mysql-user=sbtest --mysql-
password=sbpass --oltp-table-size=10000 prepare
```

3. Run the benchmark on port 3307:

```
sysbench --test=oltp --db-driver=mysql --mysql-engine-trx=yes --mysql-table-
engine=innodb --mysql-host=127.0.0.1 --mysql-port=3307 --mysql-user=sbtest --mysql-
password=sbpass --oltp-table-size=10000 --num-threads=8 run
```

You should see the following in HAProxy statistics for `pxc-back`:



pxc-back

	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
c1	0	0	-	0	3		2	2	-	3	3	112	1615		0		0	0	0	0	8m9s UP	L7OK/200 in 23ms	1	Y	-	0	0	0s	-
c2	0	0	-	0	3		3	3	-	3	3	0	0		0		0	0	0	0	8m9s UP	L7OK/200 in 31ms	1	Y	-	0	0	0s	-
c3	0	0	-	0	3		3	3	-	3	3	0	0		0		0	0	0	0	8m9s UP	L7OK/200 in 13ms	1	Y	-	0	0	0s	-
Backend	0	0		0	9		8	8	0	9	9	112	1615	0	0		0	0	0	0	8m9s UP		3	3	0		0	0s	

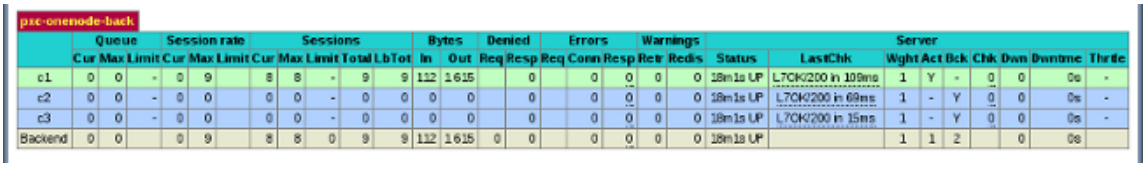
Note the `Cur` column under `Session`:

- `c1` has 2 threads connected
- `c2` and `c3` have 3 threads connected

4. Run the same benchmark on port 3306:

```
sysbench --test=oltp --db-driver=mysql --mysql-engine-trx=yes --mysql-table-
engine=innodb --mysql-host=127.0.0.1 --mysql-port=3306 --mysql-user=sbtest --mysql-
password=sbpass --oltp-table-size=10000 --num-threads=8 run
```

You should see the following in HAProxy statistics for `pxc-onenode-back`:



pxc-onenode-back

	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
c1	0	0	-	0	9		8	8	-	9	9	112	1615		0		0	0	0	0	18m1s UP	L7OK/200 in 109ms	1	Y	-	0	0	0s	-
c2	0	0	-	0	0		0	0	-	0	0	0	0		0		0	0	0	0	18m1s UP	L7OK/200 in 69ms	1	-	Y	0	0	0s	-
c3	0	0	-	0	0		0	0	-	0	0	0	0		0		0	0	0	0	18m1s UP	L7OK/200 in 15ms	1	-	Y	0	0	0s	-
Backend	0	0		0	9		8	8	0	9	9	112	1615	0	0		0	0	0	0	18m1s UP		1	1	2		0	0s	

All 8 threads are connected to the `c1` server. `c2` and `c3` are acting as backup nodes.

If you are using HAProxy for *MySQL* you can break the privilege system's host part, because *MySQL* will think that the connections are always coming from the load balancer. You can work this around using T-Proxy patches and some iptables magic for the backwards connections. However in the setup described in this how-to this is not an issue, since each application server has it's own HAProxy instance, each application server connects to 127.0.0.1, so MySQL will see that connections are coming from the application servers. Just like in the normal case.

Last update: 2022-08-23

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

11. References

11.1 Index of wsrep status variables

`wsrep_apply_oooe`

This variable shows parallelization efficiency, how often writests have been applied out of order.

`wsrep_apply_oool`

This variable shows how often a writeset with a higher sequence number was applied before one with a lower sequence number.

`wsrep_apply_window`

Average distance between highest and lowest concurrently applied sequence numbers.

`wsrep_causal_reads`

Shows the number of writesets processed while the variable `wsrep_causal_reads` was set to `ON`.

`wsrep_cert_bucket_count`

This variable, shows the number of cells in the certification index hash-table.

`wsrep_cert_deps_distance`

Average distance between highest and lowest sequence number that can be possibly applied in parallel.

`wsrep_cert_index_size`

Number of entries in the certification index.

`wsrep_cert_interval`

Average number of write-sets received while a transaction replicates.

`wsrep_cluster_conf_id`

Number of cluster membership changes that have taken place.

`wsrep_cluster_size`

Current number of nodes in the cluster.

`wsrep_cluster_state_uuid`

This variable contains UUID state of the cluster. When this value is the same as the one in `wsrep_local_state_uuid`, node is synced with the cluster.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

`wsrep_cluster_status`

Status of the cluster component. Possible values are:

- `Primary`
- `Non-Primary`
- `Disconnected`

`wsrep_commit_oooe`

This variable shows how often a transaction was committed out of order.

`wsrep_commit_oool`

This variable currently has no meaning.

`wsrep_commit_window`

Average distance between highest and lowest concurrently committed sequence number.

`wsrep_connected`

This variable shows if the node is connected to the cluster. If the value is `OFF`, the node has not yet connected to any of the cluster components. This may be due to misconfiguration.

`wsrep_evs_delayed`

Comma separated list of nodes that are considered delayed. The node format is `<uuid>:<address>:<count>`, where `<count>` is the number of entries on delayed list for that node.

`wsrep_evs_evict_list`

List of UUIDs of the evicted nodes.

`wsrep_evs_repl_latency`

This status variable provides information regarding group communication replication latency. This latency is measured in seconds from when a message is sent out to when a message is received.

The format of the output is `<min>/<avg>/<max>/<std_dev>/<sample_size>`.

`wsrep_evs_state`

Internal EVS protocol state.

`wsrep_flow_control_interval`

This variable shows the lower and upper limits for Galera flow control. The upper limit is the maximum allowed number of requests in the queue. If the queue reaches the upper limit, new requests are denied. As existing requests get processed, the queue decreases, and once it reaches the lower limit, new requests will be allowed again.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

`wsrep_flow_control_interval_high`

Shows the upper limit for flow control to trigger.

`wsrep_flow_control_interval_low`

Shows the lower limit for flow control to stop.

`wsrep_flow_control_paused`

Time since the last status query that was paused due to flow control.

`wsrep_flow_control_paused_ns`

Total time spent in a paused state measured in nanoseconds.

`wsrep_flow_control_recv`

The number of `FC_PAUSE` events received since the last status query. Unlike most status variables, this counter does not reset each time you run the query. This counter is reset when the server restarts.

`wsrep_flow_control_requested`

This variable returns whether or not a node requested a replication pause.

`wsrep_flow_control_status`

The number of `FC_PAUSE` events sent since the last status query. Unlike most status variables, this counter does not reset each time you run the query. This counter is reset when the server restarts.

`wsrep_flow_control_sent`

This variable has been implemented in `5.7.17-29.20`. The variable shows whether a node has flow control enabled for normal traffic. It does not indicate the status of flow control during SST.

`wsrep_gcache_pool_size`

This variable shows the size of the page pool and dynamic memory allocated for GCache (in bytes).

`wsrep_gcomm_uuid`

This status variable exposes UUIDs in `gvwstate.dat`, which are Galera view IDs (thus unrelated to cluster state UUIDs). This UUID is unique for each node. You will need to know this value when using manual eviction feature.

`wsrep_incoming_addresses`

Shows the comma-separated list of incoming node addresses in the cluster.

`wsrep_ist_receive_status`

This variable has been implemented in `5.7.17-29.20`. Displays the progress of IST for joiner node. If IST is not running, the value is blank. If IST is running, the value is the percentage of transfer completed.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

`wsrep_ist_receive_seqno_end`

The sequence number of the last transaction in IST.

`wsrep_ist_receive_seqno_current`

The sequence number of the current transaction in IST.

`wsrep_ist_receive_seqno_start`

The sequence number of the first transaction in IST.

`wsrep_last_applied`

This variable has been implemented in `5.7.20-29.24`. Sequence number of the last applied transaction.

`wsrep_last_committed`

Sequence number of the last committed transaction.

`wsrep_local_bf_aborts`

Number of local transactions that were aborted by replica transactions while being executed.

`wsrep_local_cached_downto`

The lowest sequence number in GCache. This information can be helpful with determining IST and SST. If the value is `0`, then it means there are no writesets in GCache (usual for a single node).

`wsrep_local_cert_failures`

Number of writesets that failed the certification test.

`wsrep_local_commits`

Number of writesets commited on the node.

`wsrep_local_index`

Node's index in the cluster.

`wsrep_local_recv_queue`

Current length of the receive queue (that is, the number of writesets waiting to be applied).

`wsrep_local_recv_queue_avg`

Average length of the receive queue since the last status query. When this number is bigger than `0` this means node can't apply writesets as fast as they are received. This could be a sign that the node is overloaded and it may cause replication throttling.

`wsrep_local_replays`

Number of transaction replays due to *asymmetric lock granularity*.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

`wsrep_local_send_queue`

Current length of the send queue (that is, the number of writesets waiting to be sent).

`wsrep_local_send_queue_avg`

Average length of the send queue since the last status query. When cluster experiences network throughput issues or replication throttling, this value will be significantly bigger than `0`.

`wsrep_local_state`

`wsrep_local_state_comment`

Internal number and the corresponding human-readable comment of the node's state. Possible values are:

Num	Comment	Description
1	Joining	Node is joining the cluster
2	Donor/Desynced	Node is the donor to the node joining the cluster
3	Joined	Node has joined the cluster
4	Synced	Node is synced with the cluster

`wsrep_local_state_uuid`

The UUID of the state stored on the node.

`wsrep_protocol_version`

Version of the wsrep protocol used.

`wsrep_provider_name`

Name of the wsrep provider (usually `Galera`).

`wsrep_provider_vendor`

Name of the wsrep provider vendor (usually `Codership Oy`)

`wsrep_provider_version`

Current version of the wsrep provider.

`wsrep_ready`

This variable shows if node is ready to accept queries. If status is `OFF`, almost all queries will fail with `ERROR 1047 (08S01) Unknown Command` error (unless the `wsrep_on` variable is set to `0`).

`wsrep_received`

Total number of writesets received from other nodes.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

`wsrep_received_bytes`

Total size (in bytes) of writesets received from other nodes.

`wsrep_repl_data_bytes`

Total size (in bytes) of data replicated.

`wsrep_repl_keys`

Total number of keys replicated.

`wsrep_repl_keys_bytes`

Total size (in bytes) of keys replicated.

`wsrep_repl_other_bytes`

Total size of other bits replicated.

`wsrep_replicated`

Total number of writesets sent to other nodes.

`wsrep_replicated_bytes`

Total size (in bytes) of writesets sent to other nodes.

Last update: 2023-09-08

11.2 Index of wsrep system variables

Percona XtraDB Cluster introduces a number of MySQL system variables related to write-set replication.

`pxc_encrypt_cluster_traffic`

Option	Description
Command Line:	`--pxc-encrypt-cluster-traffic`
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	`OFF`

This variable has been implemented in `5.7.16`. Enables automatic configuration of SSL encryption. When disabled, you need to configure SSL manually to encrypt Percona XtraDB Cluster traffic.

Possible values:

- `OFF`, `0`, `false`: Disabled (default)
- `ON`, `1`, `true`: Enabled

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

For more information, see SSL Automatic Configuration.

`pxc_maint_mode`

Option	Description
Command Line:	`--pxc-maint-mode`
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	`DISABLED`

This variable has been implemented in `5.7.16`. Specifies the maintenance mode for taking a node down without adjusting settings in ProxySQL.

The following values are available:

- `DISABLED`: This is the default state that tells ProxySQL to route traffic to the node as usual.
- `SHUTDOWN`: This state is set automatically when you initiate node shutdown.
- `MAINTENANCE`: You can manually change to this state if you need to perform maintenance on a node without shutting it down.

For more information, see Assisted Maintenance Mode.

`pxc_maint_transition_period`

Option	Description
Command Line:	`--pxc-maint-transition-period`
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	`10` (ten seconds)

This variable has been implemented in `5.7.16`. Defines the transition period when you change `pxc_maint_mode` to `SHUTDOWN`. By default, the period is set to 10 seconds, which should be enough for most transactions to finish. You can increase the value to accommodate for longer-running transactions.

For more information, see Assisted Maintenance Mode.

`pxc_strict_mode`

Option	Description
Command Line:	`--pxc-strict-mode`
Config File:	Yes
Scope:	Global
Dynamic:	Yes

Percona and/or its affiliates © 2023 — Cookie Consent

Option	Description
Default Value:	`ENFORCING` or `DISABLED`

This variable has been implemented in `5.7`.Controls PXC Strict Mode, which runs validations to avoid the use of experimental and unsupported features in Percona XtraDB Cluster.

Depending on the actual mode you select, upon encountering a failed validation, the server will either throw an error (halting startup or denying the operation), or log a warning and continue running as normal. The following modes are available:

- `DISABLED`: Do not perform strict mode validations and run as normal.
- `PERMISSIVE`: If a validation fails, log a warning and continue running as normal.
- `ENFORCING`: If a validation fails during startup, halt the server and throw an error. If a validation fails during runtime, deny the operation and throw an error.
- `MASTER`: The same as `ENFORCING` except that the validation of explicit table locking is not performed. This mode can be used with clusters in which write operations are isolated to a single node.

By default, `pxc_strict_mode` is set to `ENFORCING`, except if the node is acting as a standalone server or the node is bootstrapping, then `pxc_strict_mode` defaults to `DISABLED`.

Note

When changing the value of `pxc_strict_mode` from `DISABLED` or `PERMISSIVE` to `ENFORCING` or `MASTER`, ensure that the following configuration is used:

- `wsrep_replicate_myisam=OFF`
- `binlog_format=ROW`
- `log_output=FILE` or `log_output=NONE` or `log_output=FILE,NONE`

The `SERIALIZABLE` method of isolation is not allowed in `ENFORCING` mode.

For more information, see PXC Strict Mode.

`wsrep_auto_increment_control`

Option	Description
Command Line:	`--wsrep-auto-increment-control`
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	`ON`

Enables automatic adjustment of auto-increment system variables depending on the size of the cluster:

- `auto_increment_increment` controls the interval between successive `AUTO_INCREMENT` column values
- `auto_increment_offset` determines the starting point for the `AUTO_INCREMENT` column value

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

This helps prevent auto-increment replication conflicts across the cluster by giving each node its own range of auto-increment values. It is enabled by default.

Automatic adjustment may not be desirable depending on application's use and assumptions of auto-increments. It can be disabled in source-replica clusters.

`wsrep_causal_reads`

Option	Description
Command Line:	`--wsrep-causal-reads`
Config File:	Yes
Scope:	Global, Session
Dynamic:	Yes
Default Value:	`OFF`

This variable has been implemented in `5.6.20-25.7`. In some cases, the source may apply events faster than a replica, which can cause source and replica to become out of sync for a brief moment. When this variable is set to `ON`, the replica will wait until that event is applied before doing any other queries. Enabling this variable will result in larger latencies.

Note

This variable was deprecated because enabling it is the equivalent of setting `wsrep_sync_wait` to `1`.

`wsrep_certify_nonPK`

Option	Description
Command Line:	`--wsrep-certify-nonpk`
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	`ON`

Enables automatic generation of primary keys for rows that don't have them. Write set replication requires primary keys on all tables to allow for parallel applying of transactions. This variable is enabled by default. As a rule, make sure that all tables have primary keys.

`wsrep_cluster_address`

Option	Description
Command Line:	`--wsrep-cluster-address`
Config File:	Yes
Scope:	Global
Dynamic:	Yes

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Defines the back-end schema, IP addresses, ports, and options that the node uses when connecting to the cluster. This variable needs to specify at least one other node's address, which is alive and a member of the cluster. In practice, it is best (but not necessary) to provide a complete list of all possible cluster nodes. The value should be of the following format:

```
<schema>://<address>[?<option1>=<value1>[&<option2>=<value2>]],...
```

The only back-end schema currently supported is `gcomm`. The IP address can contain a port number after a colon. Options are specified after `?` and separated by `&`. You can specify multiple addresses separated by commas.

For example:

```
wsrep_cluster_address="gcomm://192.168.0.1:4567?gmcast.listen_addr=0.0.0.0:5678"
```

If an empty `gcomm://` is provided, the node will bootstrap itself (that is, form a new cluster). It is not recommended to have empty cluster address in production config after the cluster has been bootstrapped initially. If you want to bootstrap a new cluster with a node, you should pass the `--wsrep-new-cluster` option when starting.

`wsrep_cluster_name`

Option	Description
Command Line:	`--wsrep-cluster-name`
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	`my_wsrep_cluster`

Specifies the name of the cluster and must be identical on all nodes. A node checks the value when attempting to connect to the cluster. If the names match, the node connects.

Edit the value in the `my.cnf` in the [galera] section.

```
[galera]

    wsrep_cluster_name=simple-cluster
```

Execute `SHOW VARIABLES` with the LIKE operator to view the variable:

```
mysql> SHOW VARIABLES LIKE 'wsrep_cluster_name';
```

Expected output

```
+--------------------+----------------+
| Variable_name      | Value          |
+--------------------+----------------+
| wsrep_cluster_name | simple-cluster |
+--------------------+----------------+
```


<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Note

It should not exceed 32 characters. A node cannot join the cluster if the cluster names do not match. You must re-bootstrap the cluster after a name change.

`wsrep_convert_lock_to_trx`

Option	Description
Command Line:	`--wsrep-convert-lock-to-trx`
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	`OFF`

This variable has been deprecated in `5.7.23-31.31`. Defines whether locking sessions should be converted into transactions. By default, this is disabled.

Enabling this variable can help older applications to work in a multi-source setup by converting `LOCK/UNLOCK TABLES` statements into `BEGIN/COMMIT` statements. It is not the same as support for locking sessions, but it does prevent the database from ending up in a logically inconsistent state. Enabling this variable can also result in having huge write-sets.

`wsrep_data_home_dir`

Option	Description
Command Line:	No
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	`/var/lib/mysql` (or whatever path is specified by `datadir`)

Specifies the path to the directory where the wsrep provider stores its files (such as `grastate.dat`).

`wsrep_dbug_option`

Option	Description
Command Line:	`--wsrep-dbug-option`
Config File:	Yes
Scope:	Global
Dynamic:	Yes

Defines `DBUG` options to pass to the wsrep provider.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

`wsrep_debug`

Option	Description
Command Line:	`--wsrep-debug`
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	`OFF`

Enables additional debugging output for the database server error log. By default, it is disabled. This variable can be used when trying to diagnose problems or when submitting a bug.

You can set `wsrep_debug` in the following `my.cnf` groups:

- Under `[mysqld]` it enables debug logging for `mysqld` and the SST script
- Under `[sst]` it enables debug logging for the SST script only

Note

Do not enable debugging in production environments, because it logs authentication info (that is, passwords).

`wsrep_desync`

Option	Description
Command Line:	No
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	`OFF`

Defines whether the node should participate in Flow Control. By default, this variable is disabled, meaning that if the receive queue becomes too big, the node engages in Flow Control: it works through the receive queue until it reaches a more manageable size. For more information, see `wsrep_local_recv_queue` and `wsrep_flow_control_interval`.

Enabling this variable will disable Flow Control for the node. It will continue to receive write-sets that it is not able to apply, the receive queue will keep growing, and the node will keep falling behind the cluster indefinitely.

Toggling this back to `OFF` will require an IST or an SST, depending on how long it was desynchronized. This is similar to cluster desynchronization, which occurs during RSU TOI. Because of this, it's not a good idea to enable `wsrep_desync` for a long period of time or for several nodes at once.

Note

You can also desync a node using the `/\*! WSREP_DESYNC \*/` query comment.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

`wsrep_dirty_reads`

Option	Description
Command Line:	`--wsrep-dirty-reads`
Config File:	Yes
Scope:	Session, Global
Dynamic:	Yes
Default Value:	`OFF`

Defines whether the node accepts read queries when in a non-operational state, that is, when it loses connection to the Primary Component. By default, this variable is disabled and the node rejects all queries, because there is no way to tell if the data is correct.

If you enable this variable, the node will permit read queries (`USE`, `SELECT`, `LOCK TABLE`, and `UNLOCK TABLES`), but any command that modifies or updates the database on a non-operational node will still be rejected (including DDL and DML statements, such as `INSERT`, `DELETE`, and `UPDATE`).

To avoid deadlock errors, set the `wsrep_sync_wait` variable to `0` if you enable `wsrep_dirty_reads`.

`wsrep_drupal_282555_workaround`

Option	Description
Command Line:	`--wsrep-drupal-282555-workaround`
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	`OFF`

This variable has been announced as deprecated in `5.7.24-31.33`. Enables a workaround for MySQL InnoDB bug that affects Drupal (Drupal bug #282555 and MySQL bug #41984). In some cases, duplicate key errors would occur when inserting the `DEFAULT` value into an `AUTO_INCREMENT` column.

`wsrep_forced_binlog_format`

Option	Description
Command Line:	`--wsrep-forced-binlog-format`
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	`NONE`

This variable has been announced as deprecated in `5.7.22-29.26`. Defines a binary log format that will always be effective, regardless of the client session `binlog_format` variable value.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Possible values for this variable are:

- `ROW` : Force row-based logging format
- `STATEMENT` : Force statement-based logging format
- `MIXED` : Force mixed logging format
- `NONE` : Do not force the binary log format and use whatever is set by the `binlog_format` variable (default)

`wsrep_load_data_splitting`

Option	Description
Command Line:	`--wsrep-load-data-splitting`
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	`ON`

Defines whether the node should split large `LOAD DATA` transactions. This variable is enabled by default, meaning that `LOAD DATA` commands are split into transactions of 10 000 rows or less.

If you disable this variable, then huge data loads may prevent the node from completely rolling the operation back in the event of a conflict, and whatever gets committed stays committed.

Note

It doesn't work as expected with `autocommit=0` when enabled.

`wsrep_log_conflicts`

Option	Description
Command Line:	`--wsrep-log-conflicts`
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	`OFF`

Defines whether the node should log additional information about conflicts. By default, this variable is disabled and Percona XtraDB Cluster uses standard logging features in MySQL.

If you enable this variable, it will also log table and schema where the conflict occurred, as well as the actual values for keys that produced the conflict.

`wsrep_max_ws_rows`

Option	Description
Command Line:	`--wsrep-max-ws-rows`

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Option	Description
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	`0` (no limit)

Defines the maximum number of rows each write-set can contain.

By default, there is no limit for the maximum number of rows in a write-set. The maximum allowed value is `1048576`.

`wsrep_max_ws_size`

Option	Description
Command Line:	`--wsrep_max_ws_size`
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	`2147483647` (2 GB)

Defines the maximum write-set size (in bytes). Anything bigger than the specified value will be rejected.

You can set it to any value between `1024` and the default `2147483647`.

`wsrep_node_address`

Option	Description
Command Line:	`--wsrep-node-address`
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	IP of the first network interface (`eth0`) and default port (`4567`)

Specifies the network address of the node. By default, this variable is set to the IP address of the first network interface (usually `eth0` or `enp2s0`) and the default port (`4567`).

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

While default value should be correct in most cases, there are situations when you need to specify it manually. For example:

- Servers with multiple network interfaces
- Servers that run multiple nodes
- Network Address Translation (NAT)
- Clusters with nodes in more than one region
- Container deployments, such as Docker
- Cloud deployments, such as Amazon EC2 (use the global DNS name instead of the local IP address)

The value should be specified in the following format:

```
<ip_address>[:port]
```

Note

The value of this variable is also used as the default value for the `wsrep_sst_receive_address` variable and the `ist.recv_addr` option.

`wsrep_node_incoming_address`

Option	Description
Command Line:	`--wsrep-node-incoming-address`
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	`AUTO`

Specifies the network address from which the node expects client connections. By default, it uses the IP address from `wsrep_node_address` and port number 3306.

This information is used for the `wsrep_incoming_addresses` variable which shows all active cluster nodes.

`wsrep_node_name`

Option	Description
Command Line:	`--wsrep-node-name`
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	The node's host name

Defines a unique name for the node. Defaults to the host name.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

In many situations, you may use the value of this variable as a means to identify the given node in the cluster as the alternative to using the node address (the value of the `wsrep_node_address`).

Note

The variable `wsrep_sst_donor` is an example where you may only use the value of `wsrep_node_name` and the node address is not permitted.

`wsrep_notify_cmd`

Option	Description
Command Line:	`--wsrep-notify-cmd`
Config File:	Yes
Scope:	Global
Dynamic:	No

Specifies the notification command that the node should execute whenever cluster membership or local node status changes. This can be used for alerting or to reconfigure load balancers.

Note

The node will block and wait until the command or script completes and returns before it can proceed. If the script performs any potentially blocking or long-running operations, such as network communication, you should consider initiating such operations in the background and have the script return immediately.

`wsrep_on`

Option	Description
Command Line:	No
Config File:	No
Scope:	Session
Dynamic:	Yes
Default Value:	`ON`

Defines if current session transaction changes for a node are replicated to the cluster.

If set to `OFF` for a session, no transaction changes are replicated in that session. The setting does not cause the node to leave the cluster, and the node communicates with other nodes.

`wsrep_OSU_method`

Option	Description
Command Line:	`--wsrep-OSU-method`
Config File:	Yes
Scope:	Global, Session

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Option	Description
Dynamic:	Yes
Default Value:	`TOI`

Defines the method for Online Schema Upgrade that the node uses to replicate DDL statements. The following methods are available:

`TOI`: When the *Total Order Isolation* method is selected, data definition language (DDL) statements are processed in the same order with regards to other transactions in each node. This guarantees data consistency.

In the case of DDL statements, the cluster will have parts of the database locked and it will behave like a single server. In some cases (like big `ALTER TABLE`) this could have impact on cluster's performance and availability, but it could be fine for quick changes that happen almost instantly (like fast index changes).

When DDL statements are processed under TOI, the DDL statement will be replicated up front to the cluster. That is, the cluster will assign global transaction ID for the DDL statement before DDL processing begins. Then every node in the cluster has the responsibility to execute the DDL statement in the given slot in the sequence of incoming transactions, and this DDL execution has to happen with high priority.

Important

Under the `TOI` method, when DDL operations are performed, `MDL` is ignored. If `MDL` is important, use the `RSU` method.

`RSU`: When the *Rolling Schema Upgrade* method is selected, DDL statements won't be replicated across the cluster. Instead, it's up to the user to run them on each node separately.

The node applying the changes will desynchronize from the cluster briefly, while normal work happens on all the other nodes. When a DDL statement is processed, the node will apply delayed replication events.

The schema changes must be backwards compatible for this method to work, otherwise, the node that receives the change will likely break Galera replication. If replication breaks, SST will be triggered when the node tries to join again but the change will be undone.

Note

This variable's behavior is consistent with MySQL behavior for variables that have both global and session scope. This means if you want to change the variable in current session, you need to do it with `SET wsrep_OSU_method` (without the `GLOBAL` keyword). Setting the variable with `SET GLOBAL wsrep_OSU_method` will change the variable globally but it won't have effect on the current session.

`wsrep_preordered`

Option	Description
Command Line:	`--wsrep-preordered`
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	`OFF`

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

This variable has been announced as deprecated in `5.7.24-31.33`. Defines whether the node should use transparent handling of preordered replication events (like replication from traditional source). By default, this is disabled.

If you enable this variable, such events will be applied locally first before being replicated to other nodes in the cluster. This could increase the rate at which they can be processed, which would be otherwise limited by the latency between the nodes in the cluster.

Preordered events should not interfere with events that originate on the local node. Therefore, you should not run local update queries on a table that is also being updated through asynchronous replication.

`wsrep_provider`

Option	Description
Command Line:	`--wsrep-provider`
Config File:	Yes
Scope:	Global
Dynamic:	No

Specifies the path to the Galera library. This is usually `/usr/lib64/libgalera_smm.so` on *CentOS/RHEL* and `/usr/lib/libgalera_smm.so` on *Debian/Ubuntu*.

If you do not specify a path or the value is not valid, the node will behave as standalone instance of MySQL.

`wsrep_provider_options`

Option	Description
Command Line:	`--wsrep-provider-options`
Config File:	Yes
Scope:	Global
Dynamic:	No

Specifies optional settings for the replication provider documented in Index of :variable:`wsrep_provider` options. These options affect how various situations are handled during replication.

`wsrep_recover`

Option	Description
Command Line:	`--wsrep-recover`
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	`OFF`
Location:	mysqld_safe`

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Recovers database state after crash by parsing GTID from the log. If the GTID is found, it will be assigned as the initial position for server.

`wsrep_reject_queries`

Option	Description
Command Line:	No
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	`NONE`

Defines whether the node should reject queries from clients. Rejecting queries can be useful during upgrades, when you want to keep the node up and apply write-sets without accepting queries.

When a query is rejected, the following error is returned:

```
Error 1047: Unknown command
```

The following values are available:

- `NONE`: Accept all queries from clients (default)
- `ALL`: Reject all new queries from clients, but maintain existing client connections
- `ALL_KILL`: Reject all new queries from clients and kill existing client connections

Note

This variable doesn't affect Galera replication in any way, only the applications that connect to the database are affected. If you want to desync a node, use `wsrep_desync`.

`wsrep_replicate_myisam`

Option	Description
Command Line:	`--wsrep-replicate-myisam`
Config File:	Yes
Scope:	Session, Global
Dynamic:	No
Default Value:	`OFF`

Defines whether DML statements for MyISAM tables should be replicated. It is disabled by default, because MyISAM replication is still experimental.

On the global level, `wsrep_replicate_myisam` can be set only during startup. On session level, you can change it during runtime as well.

For older nodes in the cluster, `wsrep_replicate_myisam` should work since the TOI decision (for MyISAM DDL) is done on origin node. Mixing of non-MyISAM and MyISAM tables in the same DDL statement is not

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

recommended when `wsrep_replicate_myisam` is disabled, since if any table in the list is MyISAM, the whole DDL statement is not put under TOI.

Note

You should keep in mind the following when using MyISAM replication:

- DDL (CREATE/DROP/TRUNCATE) statements on MyISAM will be replicated irrespective of `wsrep_replicate_myisam` value
- DML (INSERT/UPDATE/DELETE) statements on MyISAM will be replicated only if `wsrep_replicate_myisam` is enabled
- SST will get full transfer irrespective of `wsrep_replicate_myisam` value (it will get MyISAM tables from donor)
- Difference in configuration of `pxc-cluster` node on enforce_storage_engine front may result in picking up different engine for the same table on different nodes
- `CREATE TABLE AS SELECT` (CTAS) statements use non-TOI replication and are replicated only if there is involvement of InnoDB table that needs transactions (in case of MyISAM table, CTAS statements will not be replicated).

`wsrep_restart_slave`

Option	Description
Command Line:	`--wsrep-restart-slave`
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	`OFF`

Defines whether replication replica should be restarted when the node joins back to the cluster. Enabling this can be useful because asynchronous replication replica thread is stopped when the node tries to apply the next replication event while the node is in non-primary state.

`wsrep_retry_autocommit`

Option	Description
Command Line:	`--wsrep-retry-autocommit`
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	`1`

Specifies the number of times autocommit transactions will be retried in the cluster if it encounters certification errors. In case there is a conflict, it should be safe for the cluster node to simply retry the statement without returning an error to the client, hoping that it will pass next time.

This can be useful to help an application using autocommit to avoid deadlock errors that can be triggered by replication conflicts.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

If this variable is set to `0`, autocommit transactions won't be retried.

`wsrep_RSU_commit_timeout`

Option	Description
Command Line:	`--wsrep-RSU-commit-timeout`
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	`5000`
Range:	From `5000` (5 milliseconds) to `31536000000000` (365 days)

Specifies the timeout in microseconds to allow active connection to complete COMMIT action before starting RSU.

While running RSU it is expected that user has isolated the node and there is no active traffic executing on the node. RSU has a check to ensure this, and waits for any active connection in `COMMIT` state before starting RSU.

By default this check has timeout of 5 milliseconds, but in some cases COMMIT is taking longer. This variable sets the timeout, and has allowed values from the range of (5 milliseconds, 365 days). The value is to be set in microseconds. Unit of variable is in micro-secs so set accordingly.

Note

RSU operation will not auto-stop node from receiving active traffic. So there could be a continuous flow of active traffic while RSU continues to wait, and that can result in RSU starvation. User is expected to block active RSU traffic while performing operation.

`wsrep_slave_FK_checks`

Option	Description
Command Line:	`--wsrep-slave-FK-checks`
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	`ON`

Defines whether foreign key checking is done for applier threads. This is enabled by default.

`wsrep_slave_threads`

Option	Description
Command Line:	`--wsrep-slave-threads`
Config File:	Yes

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Option	Description
Scope:	Global
Dynamic:	Yes
Default Value:	`1`

Specifies the number of threads that can apply replication transactions in parallel. Galera supports true parallel replication that applies transactions in parallel only when it is safe to do so. This variable is dynamic. You can increase/decrease it at any time.

Note

When you decrease the number of threads, it won't kill the threads immediately, but stop them after they are done applying current transaction (the effect with an increase is immediate though).

If any replication consistency problems are encountered, it's recommended to set this back to `1` to see if that resolves the issue. The default value can be increased for better throughput.

Review the Galera Cluster documentation for flow control for suggested settings.

You can also estimate the optimal value for this from `wsrep_cert_deps_distance` as suggested in the Galera Cluster documentation.

For more configuration tips, see this document.

`wsrep_slave_UK_checks`

Option	Description
Command Line:	`--wsrep-slave-UK-checks`
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	`OFF`

Defines whether unique key checking is done for applier threads. This is disabled by default.

`wsrep_sst_auth`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	:

Specifies authentication information for State Snapshot Transfer (SST). Required information depends on the method specified in the `wsrep_sst_method` variable.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

For more information about SST authentication, see State Snapshot Transfer.

Note

Value of this variable is masked in the log and in the `SHOW VARIABLES` query output.

`wsrep_sst_donor`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	Yes

Specifies a list of nodes (using their `wsrep_node_name` values) that the current node should prefer as donors for SST and IST.

Warning

Using IP addresses of nodes instead of node names (the value of `wsrep_node_name`) as values of `wsrep_sst_donor` results in an error.

```
ERROR] WSREP: State transfer request failed unrecoverably: 113 (No route
to host). Most likely it is due to inability to communicate with the
cluster primary component. Restart required.
```

If the value is empty, the first node in SYNCED state in the index becomes the donor and will not be able to serve requests during the state transfer.

To consider other nodes if the listed nodes are not available, add a comma at the end of the list, for example:

```
wsrep_sst_donor=node1,node2,
```

If you remove the trailing comma from the previous example, then the joining node will consider *only* `node1` and `node2`.

Note

By default, the joiner node does not wait for more than 100 seconds to receive the first packet from a donor. This is implemented via the `sst-initial-timeout` option. If you set the list of preferred donors without the trailing comma or believe that all nodes in the cluster can often be unavailable for SST (this is common for small clusters), then you may want to increase the initial timeout (or disable it completely if you don't mind the joiner node waiting for the state transfer indefinitely).

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

`wsrep_sst_donor_rejects_queries`

Option	Description
Command Line:	`--wsrep-sst-donor-rejects-queries`
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	OFF

Defines whether the node should reject blocking client sessions when it is serving as a donor during a blocking state transfer method (when `wsrep_sst_method` is set to `mysqldump` or `rsync`). This is disabled by default, meaning that the node accepts such queries.

If you enable this variable, queries will return the `Unknown command` error. This can be used to signal load-balancer that the node isn't available.

`wsrep_sst_method`

Option	Description
Command Line:	`--wsrep-sst-method`
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	xtrabackup-v2

Defines the method or script for State Snapshot Transfer (SST).

Available values are:

- `xtrabackup-v2`: Uses *Percona XtraBackup* to perform SST. This method requires `wsrep_sst_auth` to be set up with credentials (`<user>:<password>`) on the donor node. Privileges and permissions for running *Percona XtraBackup* can be found in Percona XtraBackup documentation. This is the **recommended** and default method for Percona XtraDB Cluster. For more information, see Percona XtraBackup SST Configuration.
- `rsync`: Uses `rsync` to perform SST. This method doesn't use the `wsrep_sst_auth` variable.
- `mysqldump`: Uses `mysqldump` to perform SST This method requires superuser credentials for the donor node to be specified in the `wsrep_sst_auth` variable.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Note

This method is deprecated as of `5.7.22-29.26` and not recommended unless it is required for specific reasons. Also, it is not compatible with `bind_address` set to `127.0.0.1` or `localhost`, and will cause startup to fail in this case.

- `<custom_script_name>`: Galera supports Scriptable State Snapshot Transfer. This enables users to create their own custom scripts for performing SST. For example, you can create a script `/usr/bin/wsrep_MySST.sh` and specify `MySST` for this variable to run your custom SST script.
- `skip`: Use this to skip SST. This can be used when initially starting the cluster and manually restoring the same data to all nodes. It shouldn't be used permanently because it could lead to data inconsistency across the nodes.

Note

Only `xtrabackup-v2` and `rsync` provide support for clusters with GTIDs and async replicas.

`wsrep_sst_receive_address`

Option	Description
Command Line:	`--wsrep-sst-receive-address`
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	`AUTO`

Specifies the network address where donor node should send state transfers. By default, this variable is set to `AUTO`, meaning that the IP address from `wsrep_node_address` is used.

`wsrep_start_position`

Option	Description
Command Line:	`--wsrep-start-position`
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	`00000000-0000-0000-0000-00000000000000:-1`

Specifies the node's start position as `UUID:seqno`. By setting all the nodes to have the same value for this variable, the cluster can be set up without the state transfer.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

`wsrep_sync_wait`

Option	Description
Command Line:	`--wsrep-sync-wait`
Config File:	Yes
Scope:	Session, Global
Dynamic:	Yes
Default Value:	`0`

This variable has been implemented in `5.6.20-25.7`. Controls cluster-wide causality checks on certain statements. Checks ensure that the statement is executed on a node that is fully synced with the cluster.

Note

Causality checks of any type can result in increased latency.

The type of statements to undergo checks is determined by bitmask:

- `0`: Do not run causality checks for any statements. This is the default.
- `1`: Perform checks for `READ` statements (including `SELECT`, `SHOW`, and `BEGIN` or `START TRANSACTION`).
- `2`: Perform checks for `UPDATE` and `DELETE` statements.
- `3`: Perform checks for `READ`, `UPDATE`, and `DELETE` statements.
- `4`: Perform checks for `INSERT` and `REPLACE` statements.
- `5`: Perform checks for `READ`, `INSERT`, and `REPLACE` statements.
- `6`: Perform checks for `UPDATE`, `DELETE`, `INSERT`, and `REPLACE` statements.
- `7`: Perform checks for `READ`, `UPDATE`, `DELETE`, `INSERT`, and `REPLACE` statements.

Note

Setting `wsrep_sync_wait` to `1` is the equivalent of setting the deprecated `wsrep_causal_reads` to `ON`.

Last update: 2023-02-01

11.3 Index of wsrep_provider options

The following variables can be set and checked in the `wsrep_provider_options` variable. The value of the variable can be changed in the *MySQL* configuration file, `my.cnf`, or by setting the variable value in the *MySQL* client.

To change the value in `my.cnf`, the following syntax should be used:

```
wsrep_provider_options="variable1=value1;[variable2=value2]"
```

For example to set the size of the Galera buffer storage to 512 MB, specify the following in `my.cnf`:

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

```
wsrep_provider_options="gcache.size=512M"
```

Dynamic variables can be changed from the *MySQL* client using the `SET GLOBAL` command. For example, to change the value of the `pc.ignore_sb`, use the following command:

```
mysql> SET GLOBAL wsrep_provider_options="pc.ignore_sb=true";
```

11.3.1 Index

`base_dir`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	value of `datadir`

This variable specifies the data directory.

`base_host`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	value of `wsrep_node_address`

This variable sets the value of the node's base IP. This is an IP address on which Galera listens for connections from other nodes. Setting this value incorrectly would stop the node from communicating with other nodes.

`base_port`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	4567

This variable sets the port on which Galera listens for connections from other nodes. Setting this value incorrectly would stop the node from communicating with other nodes.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

`cert.log_conflicts`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	no

This variable is used to specify if the details of the certification failures should be logged.

`debug`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	no

When this variable is set to `yes`, it will enable debugging.

`evs.auto_evict`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	0

Number of entries allowed on delayed list until auto eviction takes place. Setting value to `0` disables auto eviction protocol on the node, though node response times will still be monitored. EVS protocol version (`evs.version`) `1` is required to enable auto eviction.

`evs.causal_keepalive_period`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Option	Description
Default Value:	value of `evs.keepalive_period`

This variable is used for development purposes and shouldn't be used by regular users.

`evs.debug_log_mask`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	0x1

This variable is used for EVS (Extended Virtual Synchrony) debugging. It can be used only when `wsrep_debug` is set to `ON`.

`evs.delay_margin`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	PT1S

Time period that a node can delay its response from expected until it is added to delayed list. The value must be higher than the highest RTT between nodes.

`evs.delayed_keep_period`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	PT30S

Time period that node is required to remain responsive until one entry is removed from delayed list.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

`evs.evict`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	Yes

Manual eviction can be triggered by setting the `evs.evict` to a certain node value. Setting the `evs.evict` to an empty string will clear the evict list on the node where it was set.

`evs.inactive_check_period`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	PT0.5S

This variable defines how often to check for peer inactivity.

`evs.inactive_timeout`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	PT15S

This variable defines the inactivity limit, once this limit is reached the node will be considered dead.

`evs.info_log_mask`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	0

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

This variable is used for controlling the extra EVS info logging.

`evs.install_timeout`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	PT7.5S

This variable defines the timeout on waiting for install message acknowledgments.

`evs.join_retrans_period`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	PT1S

This variable defines how often to retransmit EVS join messages when forming cluster membership.

`evs.keepalive_period`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	PT1S

This variable defines how often to emit keepalive beacons (in the absence of any other traffic).

`evs.max_install_timeouts`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Option	Description
Default Value:	1

This variable defines how many membership install rounds to try before giving up (total rounds will be `evs.max_install_timeouts` + 2).

`evs.send_window`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	10

This variable defines the maximum number of data packets in replication at a time. For WAN setups, the variable can be set to a considerably higher value than default (for example,512). The value must not be less than `evs.user_send_window`.

`evs.stats_report_period`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	PT1M

This variable defines the control period of EVS statistics reporting.

`evs.suspect_timeout`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	PT5S

This variable defines the inactivity period after which the node is “suspected” to be dead. If all remaining nodes agree on that, the node will be dropped out of cluster even before `evs.inactive_timeout` is reached.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

`evs.use_aggregate`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	true

When this variable is enabled, smaller packets will be aggregated into one.

`evs.user_send_window`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	4

This variable defines the maximum number of data packets in replication at a time. For WAN setups, the variable can be set to a considerably higher value than default (for example, 512).

`evs.version`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	0

This variable defines the EVS protocol version. Auto eviction is enabled when this variable is set to `1`. Default `0` is set for backwards compatibility.

`evs.view_forget_timeout`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Option	Description
Default Value:	PID

This variable defines the timeout after which past views will be dropped from history.

`gcache.dir`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	`datadir`

This variable can be used to define the location of the `galera.cache` file.

`gcache.freeze_purge_at_seqno`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Local, Global
Dynamic:	Yes
Default Value:	0

This variable controls the purging of the gcache and enables retaining more data in it. This variable makes it possible to use IST (Incremental State Transfer) when the node rejoins instead of SST (State Snapshot Transfer).

Set this variable on an existing node of the cluster (that will continue to be part of the cluster and can act as a potential donor node). This node continues to retain the write-sets and allows restarting the node to rejoin by using IST.

See also

Percona Database Performance Blog:

- All You Need to Know About GCache (Galera-Cache)
- Want IST Not SST for Node Rejoins? We Have a Solution!

The `gcache.freeze_purge_at_seqno` variable takes three values:

-1 (default)

No freezing of gcache, the purge operates as normal.

A valid seqno in gcache

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

The freeze purge of write-sets may not be smaller than the selected seqno. The best way to select an optimal value is to use the value of the variable :variable: `wsrep_last_applied` from the node that you plan to shut down.

now The freeze purge of write-sets is no less than the smallest seqno currently in gcache. Using this value results in freezing the gcache-purge instantly. Use this value if selecting a valid seqno in gcache is difficult.

`gcache.keep_pages_count`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Local, Global
Dynamic:	Yes
Default Value:	0

This variable is used to limit the number of overflow pages rather than the total memory occupied by all overflow pages. Whenever `gcache.keep_pages_count` is set to a non-zero value, excess overflow pages will be deleted (starting from the oldest to the newest).

Whenever either the `gcache.keep_pages_count` or the `gcache.keep_pages_size` variable is updated at runtime to a non-zero value, cleanup is called on excess overflow pages to delete them.

`gcache.keep_pages_size`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Local, Global
Dynamic:	No
Default Value:	0

This variable is used to limit the total size of overflow pages rather than the count of all overflow pages. Whenever `gcache.keep_pages_size` is set to a non-zero value, excess overflow pages will be deleted (starting from the oldest to the newest) until the total size is below the specified value.

Whenever either the `gcache.keep_pages_count` or the `gcache.keep_pages_size` variable is updated at runtime to a non-zero value, cleanup is called on excess overflow pages to delete them.

`gcache.mem_size`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Option	Description
Default Value:	0

This variable has been deprecated in `5.6.22-25.8` and shouldn't be used as it could cause a node to crash.

This variable was used to define how much RAM is available for the system.

`gcache.name`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	/var/lib/mysql/galera.cache

This variable can be used to specify the name of the Galera cache file.

`gcache.page_size`

Option	Description
Command Line:	No
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	128M

Size of the page files in page storage. The limit on overall page storage is the size of the disk. Pages are prefixed by gcache.page.

See also

- Galera Documentation: gcache.page_size
- [Percona Database Performance Blog: All You Need to Know About GCache (https://www.percona.com/blog/2016/11/16/all-you-need-to-know-about-gcache-galera-cache/)

`gcache.recover`

Option	Description
Command Line:	No
Config File:	Yes
Scope:	Global

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Option	Description
Dynamic:	No
Default Value:	No

Attempts to recover a node's gcache file to a usable state on startup. If the node can successfully recover the gcache file, the node can provide IST to the remaining nodes. This ability can reduce the time needed to bring up the cluster.

An example of setting the value to yes in the configuration file:

```
wsrep_provider_options="gcache.recover=yes"
```

`gcache.size`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	128M

Size of the transaction cache for Galera replication. This defines the size of the `galera.cache` file which is used as source for IST. The bigger the value of this variable, the better are chances that the re-joining node will get IST instead of SST.

`gcomm.thread_prio`

Raises the gcomm thread priority to a higher level. Use this variable when the gcomm thread does not receive enough CPU time due to other competing threads. For example, if the gcomm threads are not frequently run, a node may drop from the cluster because of the timeout.

The format for this variable is: <policy>:<priority>. The policy value supports the following options: `other`, `fifo`, and `rr`. The priority value is an integer.

Note

Setting a priority value of 99 is not recommended. This value blocks system threads.

An example of the variable:

```
wsrep_provider_options="gcomm.thread_prio=fifo:3"
```

The description of the `policy` parameter follows:

Option	Description
other	This policy is the default Linux time-sharing scheduling. Threads run until one of the following events occur: * Thread exit

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Option	Description
	I/O request blocks the thread Higher priority thread preempts the thread
fifo	The policy uses a First-in First-out (FIFO) scheduling. These threads always immediately preempt any currently running other, batch or idle threads. The threads are run in a FIFO manner until completion, unless a higher priority thread preempts or blocks them. This policy does not use time slicing.
rr	The threads use round-robin scheduling. This thread always preempts a currently running other, batch or idle thread. The scheduler runs threads with the same priority for a fixed time in a round-robin style. When this time period is exceeded, the scheduler stops the thread and moves it to the end of the list, and runs another round-robin thread with the same priority.

See also

For information, see the Galera Cluster documentation

`gcs.fc_debug`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	0

This variable specifies after how many writesets the debug statistics about SST flow control will be posted.

`gcs.fc_factor`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	1

This variable is used for replication flow control. Replication is resumed when the replica queue drops below `gcs.fc_factor` * `gcs.fc_limit`.

`gcs.fc_limit`

Option	Description
Command Line:	Yes

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Option	Description
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	100

The dafaul velue has been changed from `16` to `100` in `5.7.17-29.20`.

This variable is used for replication flow control. Replication is paused when the replica queue exceeds this limit. In the default operation mode, flow control limit is dynamically recalculated based on the amount of nodes in the cluster, but this recalculation can be turned off with use of the `gcs.fc_master_slave` variable to make manual setting of the `gcs.fc_limit` having an effect (e.g., for configurations when writing is done to a single node in Percona XtraDB Cluster).

`gcs.fc_master_slave`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	NO
Default Value:	NO

This variable is used to specify if there is only one source node in the cluster. It affects whether flow control limit is recalculated dynamically (when `NO`) or not (when `YES`).

`gcs.max_packet_size`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	64500

This variable is used to specify the writeset size after which they will be fragmented.

`gcs.max_throttle`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Option	Description
Dynamic:	No
Default Value:	0.25

This variable specifies how much the replication can be throttled during the state transfer in order to avoid running out of memory. Value can be set to `0.0` if stopping replication is acceptable in order to finish state transfer.

`gcs.recv_q_hard_limit`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	9223372036854775807

This variable specifies the maximum allowed size of the receive queue. This should normally be `(RAM + swap) / 2`. If this limit is exceeded, Galera will abort the server.

`gcs.recv_q_soft_limit`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	0.25

This variable specifies the fraction of the `gcs.recv_q_hard_limit` after which replication rate will be throttled.

`gcs.sync_donor`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	No

This variable controls if the rest of the cluster should be in sync with the donor node. When this variable is set to `YES`, the whole cluster will be blocked if the donor node is blocked with SST.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

`gmcast.listen_addr`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	tcp://0.0.0.0:4567

This variable defines the address on which the node listens to connections from other nodes in the cluster.

`gmcast.mcast_addr`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	None

This variable should be set up if UDP multicast should be used for replication.

`gmcast.mcast_ttl`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	1

This variable can be used to define TTL for multicast packets.

`gmcast.peer_timeout`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	PT3S

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

This variable specifies the connection timeout to initiate message relaying.

`gmcast.segment`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	0

This variable specifies the group segment this member should be a part of. Same segment members are treated as equally physically close.

`gmcast.time_wait`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	PT5S

This variable specifies the time to wait until allowing peer declared outside of stable view to reconnect.

`gmcast.version`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	0

This variable shows which gmcast protocol version is being used.

`ist.recv_addr`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Option	Description
Dynamic:	No
Default Value:	value of `wsrep_node_address`

This variable specifies the address on which the node listens for Incremental State Transfer (IST).

`pc.announce_timeout`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	PT3S

Cluster joining announcements are sent every ½ second for this period of time or less if other nodes are discovered.

`pc.checksum`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	true

This variable controls whether replicated messages should be checksummed or not.

`pc.ignore_quorum`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	false

When this variable is set to `TRUE`, the node will completely ignore quorum calculations. This should be used with extreme caution even in source-replica setups, because replicas won't automatically reconnect to source in this case.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

`pc.ignore_sb`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	false

When this variable is set to `TRUE`, the node will process updates even in the case of a split brain. This should be used with extreme caution in multi-source setup, but should simplify things in source-replica cluster (especially if only 2 nodes are used).

`pc.linger`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	PT20S

This variable specifies the period for which the PC protocol waits for EVS termination.

`pc.npvo`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	false

When this variable is set to `TRUE`, more recent primary components override older ones in case of conflicting primaries.

`pc.recovery`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Option	Description
Dynamic:	No
Default Value:	true

When this variable is set to `true`, the node stores the Primary Component state to disk. The Primary Component can then recover automatically when all nodes that were part of the last saved state re-establish communication with each other. This feature allows automatic recovery from full cluster crashes, such as in the case of a data center power outage. A subsequent graceful full cluster restart will require explicit bootstrapping for a new Primary Component.

`pc.version`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	0

This status variable is used to check which PC protocol version is used.

`pc.wait_prim`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	true

When set to `TRUE`, the node waits for a primary component for the period of time specified in `pc.wait_prim_timeout`. This is useful to bring up a non-primary component and make it primary with `pc.bootstrap`.

`pc.wait_prim_timeout`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	PT30S

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

This variable is used to specify the period of time to wait for a primary component.

`pc.weight`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	1

This variable specifies the node weight that's going to be used for Weighted Quorum calculations.

`protonet.backend`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	asio

This variable is used to define which transport backend should be used. Currently only `ASIO` is supported.

`protonet.version`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	0

This status variable is used to check which transport backend protocol version is used.

`repl.causal_read_timeout`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	Yes

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Option	Description
Default Value:	PT30S

This variable specifies the causal read timeout.

`repl.commit_order`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	3

This variable is used to specify out-of-order committing (which is used to improve parallel applying performance). The following values are available:

- `0` - BYPASS: all commit order monitoring is turned off (useful for measuring performance penalty)
- `1` - OOOC: allow out-of-order committing for all transactions
- `2` - LOCAL_OOOC: allow out-of-order committing only for local transactions
- `3` - NO_OOOC: no out-of-order committing is allowed (strict total order committing)

`repl.key_format`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	Yes
Default Value:	FLAT8

This variable is used to specify the replication key format. The following values are available:

- `FLAT8` - short key with higher probability of key match false positives
- `FLAT16` - longer key with lower probability of false positives
- `FLAT8A` - same as `FLAT8` but with annotations for debug purposes
- `FLAT16A` - same as `FLAT16` but with annotations for debug purposes

`repl.max_ws_size`

Option	Description
Command Line:	Yes

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Option	Description
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	2147483647

This variable is used to specify the maximum size of a write-set in bytes. This is limited to 2 gygabytes.

`repl.proto_max`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	7

This variable is used to specify the highest communication protocol version to accept in the cluster. Used only for debugging.

`socket.checksum`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	2

This variable is used to choose the checksum algorithm for network packets. The following values are available:

- `0` - disable checksum
- `1` - plain `CRC32` (used in Galera 2.x)
- `2` - hardware accelerated `CRC32-C`

`socket.ssl`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Option	Description
Dynamic:	No
Default Value:	No

This variable is used to specify if SSL encryption should be used.

`socket.ssl_ca`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No

This variable is used to specify the path to the Certificate Authority (CA) certificate file.

`socket.ssl_cert`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No

This variable is used to specify the path to the server's certificate file (in PEM format).

`socket.ssl_key`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No

This variable is used to specify the path to the server's private key file (in PEM format).

`socket.ssl_compression`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Option	Description
Dynamic:	No
Default Value:	Yes

This variable is used to specify if the SSL compression is to be used.

`socket.ssl_cipher`

Option	Description
Command Line:	Yes
Config File:	Yes
Scope:	Global
Dynamic:	No
Default Value:	AES128-SHA

This variable is used to specify what cypher will be used for encryption.

Last update: 2022-08-16

11.4 Index of files created by PXC

- `GRA_\*.log`

  These files contain binlog events in ROW format representing the failed transaction. That means that the replica thread was not able to apply one of the transactions. For each of those file, a corresponding warning or error message is present in the mysql error log file. Those error can also be false positives like a bad `DDL` statement (dropping a table that doesn't exists for example) and therefore nothing to worry about. However it's always recommended to check these log to understand what's is happening.

  To be able to analyze these files binlog header needs to be added to the log file. To create the `GRA_HEADER` file you need an instance running with `binlog_checksum` set to `NONE` and extract first 120 bytes from the binlog file:

```
$ head -c 123 mysqld-bin.000001 > GRA_HEADER
$ cat GRA_HEADER > /var/lib/mysql/GRA_1_2-bin.log
$ cat /var/lib/mysql/GRA_1_2.log >> /var/lib/mysql/GRA_1_2-bin.log
$ mysqlbinlog -vvv /var/lib/mysql/GRA_1_2-bin.log

/*!50530 SET @@SESSION.PSEUDO_SLAVE_MODE=1*/;
/*!50003 SET @OLD_COMPLETION_TYPE=@@COMPLETION_TYPE,COMPLETION_TYPE=0*/;
DELIMITER /*!*/;
# at 4
#160809  16:04:05 server id 3  end_log_pos 123     Start: binlog v 4, server v
5.7.12-5rc1-log created 160809 16:04:05 at startup
# Warning: this binlog is either in use or was not closed properly.
ROLLBACK/*!*/;
BINLOG '
nbGpVw8DAAAAdwAAAHsAAAABAAQANS43LjEyLTVyYzEtbG9nAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAACdsalXEzgNAAgAEgAEBAQEEgAAXwAEGggAAAAICAgCAAAACgoKKioAEjQA
ALfQ8hw=
'/*!*/;
# at 123
```


<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

```
#160809  16:05:49 server id 2  end_log_pos 75     Query    thread_id=11    exec_time=0
error_code=0
use `test`/*!*/;
SET TIMESTAMP=1470738949/*!*/;
SET @@session.pseudo_thread_id=11/*!*/;
SET @@session.foreign_key_checks=1, @@session.sql_auto_is_null=0,
@@session.unique_checks=1, @@session.autocommit=1/*!*/;
SET @@session.sql_mode=1436549152/*!*/;
SET @@session.auto_increment_increment=1, @@session.auto_increment_offset=1/*!*/;
/*!\C utf8 *//*!*/;
SET
@@session.character_set_client=33,@@session.collation_connection=33,@@session.collation_serv
*!*/;
SET @@session.lc_time_names=0/*!*/;
SET @@session.collation_database=DEFAULT/*!*/;
drop table t
/*!*/;
SET @@SESSION.GTID_NEXT= 'AUTOMATIC' /* added by mysqlbinlog */ /*!*/;
DELIMITER ;
# End of log file
/*!50003 SET COMPLETION_TYPE=@OLD_COMPLETION_TYPE*/;
/*!50530 SET @@SESSION.PSEUDO_SLAVE_MODE=0*/;
```

This information can be used for checking the *MySQL* error log for the corresponding error message.

```
160805  9:33:37 8:52:21 [ERROR] Slave SQL: Error 'Unknown table 'test'' on query. Default
database: 'test'. Query: 'drop table test', Error_code: 1051
160805  9:33:37 8:52:21 [Warning] WSREP: RBR event 1 Query apply warning: 1, 3
```

In this example `DROP TABLE` statement was executed on a table that doesn't exist.

- `gcache.page`

  See `gcache.page_size`

> **See also**
>
> **Percona Database Performance Blog: All You Need to Know About GCache (Galera-Cache)** https://www.percona.com/blog/2016/11/16/all-you-need-to-know-about-gcache-galera-cache/

- `galera.cache`

  This file is used as a main writeset store. It's implemented as a permanent ring-buffer file that is preallocated on disk when the node is initialized. File size can be controlled with the variable `gcache.size`. If this value is bigger, more writesets are cached and chances are better that the re-joining node will get IST instead of SST. Filename can be changed with the `gcache.name` variable.

- `grastate.dat`

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

This file contains the Galera state information.

- `version` - grastate version
- `uuid` - a unique identifier for the state and the sequence of changes it undergoes.For more information on how UUID is generated see UUID.
- `seqno` - Ordinal Sequence Number, a 64-bit signed integer used to denote the position of the change in the sequence. `seqno` is `0` when no writesets have been generated or applied on that node, i.e., not applied/generated across the lifetime of a `grastate` file. `-1` is a special value for the `seqno` that is kept in the `grastate.dat` while the server is running to allow Galera to distinguish between a clean and an unclean shutdown. Upon a clean shutdown, the correct `seqno` value is written to the file. So, when the server is brought back up, if the value is still `-1` , this means that the server did not shut down cleanly. If the value is greater than `0`, this means that the shutdown was clean. `-1` is then written again to the file in order to allow the server to correctly detect if the next shutdown was clean in the same manner.
- `cert_index` - cert index restore through grastate is not implemented yet

Examples of this file look like this:

In case server node has this state when not running it means that that node crashed during the transaction processing.

```
# GALERA saved state
version: 2.1
uuid:    1917033b-7081-11e2-0800-707f5d3b106b
seqno:   -1
cert_index:
```

In case server node has this state when not running it means that the node was gracefully shut down.

```
# GALERA saved state
version: 2.1
uuid:    1917033b-7081-11e2-0800-707f5d3b106b
seqno:   5192193423942
cert_index:
```

In case server node has this state when not running it means that the node crashed during the DDL.

```
# GALERA saved state
version: 2.1
uuid:    00000000-0000-0000-0000-000000000000
seqno:   -1
cert_index:
```

- `gvwstate.dat`

This file is used for Primary Component recovery feature. This file is created once primary component is formed or changed, so you can get the latest primary component this node was in. And this file is deleted when the node is shutdown gracefully.

First part contains the node UUID information. Second part contains the view information. View information is written between `#vwbeg` and `#vwend`. View information consists of:

```
* view_id: [view_type] [view_uuid] [view_seq]. - `view_type` is always `3` which means
primary view. `view_uuid` and `view_seq` identifies a unique view, which could be perceived
as identifier of this primary component.

* bootstrap: [bootstarp_or_not]. - it could be `0` or `1`, but it does not affect primary
```


<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

```
component recovery process now.

* member: [node's uuid] [node's segment]. - it represents all nodes in this primary
component.

Example of this file looks like this:

```text
my_uuid: c5d5d990-30ee-11e4-aab1-46d0ed84b408
#vwbeg
view_id: 3 bc85bd53-31ac-11e4-9895-1f2ce13f2542 2
bootstrap: 0
member: bc85bd53-31ac-11e4-9895-1f2ce13f2542 0
member: c5d5d990-30ee-11e4-aab1-46d0ed84b408 0
#vwend
```
```

Last update: 2022-07-20

11.5 Frequently Asked Questions

11.5.1 How do I report bugs?

All bugs can be reported on JIRA. Please submit `error.log` files from **all** the nodes.

11.5.2 How do I solve locking issues like auto-increment?

For auto-increment, Percona XtraDB Cluster changes `auto_increment_offset` for each new node. In a single-node workload, locking is handled in the same way as *InnoDB*. In case of write load on several nodes, Percona XtraDB Cluster uses optimistic locking and the application may receive lock error in response to `COMMIT` query.

11.5.3 What if a node crashes and InnoDB recovery rolls back some transactions?

When a node crashes, after restarting, it will copy the whole dataset from another node (if there were changes to data since the crash).

11.5.4 How can I check the Galera node health?

To check the health of a Galera node, use the following query:

```
SELECT 1 FROM dual;
```

The following results of the previous query are possible:

- You get the row with `id=1` (node is healthy)
- Unknown error (node is online, but Galera is not connected/synced with the cluster)
- Connection error (node is not online)

You can also check a node's health with the `clustercheck` script. First set up the `clustercheck` user:

```
GRANT USAGE ON *.* TO 'clustercheck'@'localhost' IDENTIFIED BY PASSWORD
'*2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19';
```


Percona and/or its affiliates © 2023 — Cookie Consent

You can then check a node's health by running the `clustercheck` script:

```
/usr/bin/clustercheck clustercheck password 0
```

If the node is running, you should get the following status:

```
HTTP/1.1 200 OK
Content-Type: text/plain
Connection: close
Content-Length: 40

Percona XtraDB Cluster Node is synced.
```

In case node isn't synced or if it is offline, status will look like:

```
HTTP/1.1 503 Service Unavailable
Content-Type: text/plain
Connection: close
Content-Length: 44

Percona XtraDB Cluster Node is not synced.
```

Note

The `clustercheck` script has the following syntax:

`<user> <pass> <available_when_donor=0|1> <log_file> <available_when_readonly=0|1> <defaults_extra_file>`

Recommended: `server_args = user pass 1 /var/log/log-file 0 /etc/my.cnf.local`

Compatibility: `server_args = user pass 1 /var/log/log-file 1 /etc/my.cnf.local`

11.5.5 How does Percona XtraDB Cluster handle big transactions?

Percona XtraDB Cluster populates write set in memory before replication, and this sets the limit for the size of transactions that make sense. There are wsrep variables for maximum row count and maximum size of write set to make sure that the server does not run out of memory.

11.5.6 Is it possible to have different table structures on the nodes?

For example, if there are four nodes, with four tables: `sessions_a`, `sessions_b`, `sessions_c`, and `sessions_d`, and you want each table in a separate node, this is not possible for InnoDB tables. However, it will work for MEMORY tables.

11.5.7 What if a node fails or there is a network issue between nodes?

The quorum mechanism in Percona XtraDB Cluster will decide which nodes can accept traffic and will shut down the nodes that do not belong to the quorum. Later when the failure is fixed, the nodes will need to copy data from the working cluster.

The algorithm for quorum is Dynamic Linear Voting (DLV). The quorum is preserved if (and only if) the sum weight of the nodes in a new component strictly exceeds half that of the preceding Primary Component, minus the nodes which left gracefully.

The mechanism is described in detail in Galera documentation.

Percona and/or its affiliates © 2023 — Cookie Consent

11.5.8 How would the quorum mechanism handle split brain?

The quorum mechanism cannot handle split brain. If there is no way to decide on the primary component, Percona XtraDB Cluster has no way to resolve a split brain. The minimal recommendation is to have 3 nodes. However, it is possible to allow a node to handle traffic with the following option:

```
wsrep_provider_options="pc.ignore_sb = yes"
```

11.5.9 Why a node stops accepting commands if the other one fails in a 2-node setup?

This is expected behavior to prevent split brain. For more information, see previous question or Galera documentation.

11.5.10 Is it possible to set up a cluster without state transfer?

It is possible in two ways:

- By default, Galera reads starting position from a text file `<datadir>/grastate.dat`. Make this file identical on all nodes, and there will be no state transfer after starting a node.
- Use the `wsrep_start_position` variable to start the nodes with the same `UUID:seqno` value.

11.5.11 What TCP ports are used by Percona XtraDB Cluster?

You may need to open up to four ports if you are using a firewall:

1. Regular MySQL port (default is 3306).
2. Port for group communication (default is 4567). It can be changed using the following option:

   ```
   wsrep_provider_options ="gmcast.listen_addr=tcp://0.0.0.0:4010; "
   ```

3. Port for State Snapshot Transfer (default is 4444). It can be changed using the following option:

   ```
   wsrep_sst_receive_address=10.11.12.205:5555
   ```

4. Port for Incremental State Transfer (default is port for group communication + 1 or 4568). It can be changed using the following option:

   ```
   wsrep_provider_options = "ist.recv_addr=10.11.12.206:7777; "
   ```

11.5.12 Is there “async” mode or only “sync” commits are supported?

Percona XtraDB Cluster does not support “async” mode, all commits are synchronous on all nodes. To be precise, the commits are “virtually” synchronous, which means that the transaction should pass *certification* on nodes, not physical commit. Certification means a guarantee that the transaction does not have conflicts with other transactions on the corresponding node.

11.5.13 Does it work with regular MySQL replication?

Yes. On the node you are going to use as source, you should enable `log-bin` and `log-slave-update` options.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

11.5.14 Why the init script (/etc/init.d/mysql) does not start?

Try to disable SELinux with the following command:

```
echo 0 > /selinux/enforce
```

11.5.15 What does “nc: invalid option – ‘d’” in the sst.err log file mean?

This is Debian/Ubuntu specific error. Percona XtraDB Cluster uses `netcat-openbsd` package. This dependency has been fixed in recent releases. Future releases of Percona XtraDB Cluster will be compatible with any `netcat` (see bug #959970).

Last update: 2022-07-20

11.6 Glossary

LSN

Each InnoDB page (usually 16kb in size) contains a log sequence number, or LSN. The LSN is the system version number for the entire database. Each page’s LSN shows how recently it was changed.

InnoDB

Storage engine which provides ACID-compliant transactions and foreign key support, among others improvements over MyISAM. It is the default engine for *MySQL* as of the 5.5 series.

MyISAM

Previous default storage engine for *MySQL* for versions prior to 5.5. It doesn’t fully support transactions but in some scenarios may be faster than InnoDB. Each table is stored on disk in 3 files: .frm,i `.MYD`, `.MYI`.

GTID

Global Transaction ID, in *Percona XtraDB Cluster* it consists of UUID and an ordinal sequence number which denotes the position of the change in the sequence.

HAProxy

HAProxy is a free, very fast and reliable solution offering high availability, load balancing, and proxying for TCP and HTTP-based applications. It is particularly suited for web sites crawling under very high loads while needing persistence or Layer7 processing. Supporting tens of thousands of connections is clearly realistic with todays hardware. Its mode of operation makes its integration into existing architectures very easy and riskless, while still offering the possibility not to expose fragile web servers to the net.

IST

Incremental State Transfer. Functionality which instead of whole state snapshot can catch up with te group by receiving the missing writesets, but only if the writeset is still in the donor’s writeset cache.

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

SST

State Snapshot Transfer is the full copy of data from one node to another. It's used when a new node joins the cluster, it has to transfer data from existing node. There are three methods of SST available in *Percona XtraDB Cluster*: **mysqldump**, **rsync** and **xtrabackup**. The downside of mysqldump and rsync is that the node becomes *READ-ONLY* while data is being copied from one node to another (SST applies **FLUSH TABLES WITH READ LOCK** command). Xtrabackup SST does not require **READ LOCK** for the entire syncing process, only for syncing the *MySQL* system tables and writing the information about the binlog, galera and replica information (same as the regular *Percona XtraBackup* backup). State snapshot transfer method can be configured with the `wsrep_sst_method` variable.

UUID

Universally Unique IDentifier which uniquely identifies the state and the sequence of changes node undergoes. 128-bit UUID is a classic DCE UUID Version 1 (based on current time and MAC address). Although in theory this UUID could be generated based on the real MAC-address, in the Galera it is always (without exception) based on the generated pseudo-random addresses ("locally administered" bit in the node address (in the UUID structure) is always equal to unity).

Complete structure of the 128-bit UUID field and explanation for its generation are as follows:

From	To	Length	Content
0	31	32	Bits 0-31 of Coordinated Universal Time (UTC) as a count of 100-nanosecond intervals since 00:00:00.00, 15 October 1582 encoded as big-endian 32-bit number.
32	47	16	Bits 32-47 of UTC as a count of 100-nanosecond intervals since 00:00:00.00, 15 October 1582, encoded as big-endian 16-bit number.
48	59	12	Bits 48-59 of UTC as a count of 100-nanosecond intervals since 00:00:00.00, 15 October 1582, encoded as big-endian 16-bit number.
60	63	4	UUID version number: always equal to 1 (DCE UUID).
64	69	6	most-significants bits of random number, which generated from the server process PID and Coordinated Universal Time (UTC) as a count of 100-nanosecond intervals since 00:00:00.00, 15 October 1582.
70	71	2	UID variant: always equal to binary 10 (DCE variant).
72	79	8	8 least-significant bits of random number, which generated from the server process PID and Coordinated Universal Time (UTC) as a count of 100-nanosecond intervals since 00:00:00.00, 15 October 1582.
80	80	1	Random bit ("unique node identifier").

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

From	To	Length	Content
81	81	1	Always equal to the one (“locally administered MAC address”).
82	127	46	Random bits (“unique node identifier”): readed from the `/dev/urandom` or (if `/dev/urandom` is unavailable) generated based on the server process PID, current time and bits of the default “zero node identifier” (entropy data).

XtraBackup

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

XtraDB

Percona XtraDB is an enhanced version of the InnoDB storage engine, designed to better scale on modern hardware, and including a variety of other features useful in high performance environments. It is fully backwards compatible, and so can be used as a drop-in replacement for standard InnoDB. More information here .

XtraDB Cluster

Percona XtraDB Cluster is a high availability solution for MySQL.

Percona XtraDB Cluster

Percona XtraDB Cluster (PXC) is a high availability solution for MySQL.

my.cnf

This file refers to the database server’s main configuration file. Most Linux distributions place it as `/etc/mysql/my.cnf` or `/etc/my.cnf`, but the location and name depends on the particular installation. Note that this is not the only way of configuring the server, some systems does not have one even and rely on the command options to start the server and its defaults values.

cluster replication

Normal replication path for cluster members. Can be encrypted (not by default) and unicast or multicast (unicast by default). Runs on tcp port 4567 by default.

Database replication

See Database replication in Galera Cluster documentation

Slave database server

See Database replication in Galera Cluster documentation

Multi-master replication

See Multi-master replication in Galera Cluster documentation

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

Node

See Database replication in Galera Cluster documentation

datadir

The directory in which the database server stores its databases. Most Linux distribution use `/var/lib/mysql` by default.

donor node

The node elected to provide a state transfer (SST or IST).

ibdata

Default prefix for tablespace files, e.g., `ibdata1` is a 10MB autoextendable file that *MySQL* creates for the shared tablespace by default.

joiner node

The node joining the cluster, usually a state transfer target.

node

A cluster node – a single mysql instance that is in the cluster.

primary cluster

A cluster with quorum. A non-primary cluster will not allow any operations and will give `Unknown command` errors on any clients attempting to read or write from the database.

quorum

A majority (> 50%) of nodes. In the event of a network partition, only the cluster partition that retains a quorum (if any) will remain Primary by default.

split brain

Split brain occurs when two parts of a computer cluster are disconnected, each part believing that the other is no longer running. This problem can lead to data inconsistency.

.frm

For each table, the server will create a file with the `.frm` extension containing the table definition (for all storage engines).

Last update: 2022-07-20

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

11.7 Copyright and licensing information

11.7.1 Documentation licensing

Percona XtraDB Cluster documentation is (C)2009-2023 Percona LLC and/or its affiliates and is distributed under the Creative Commons Attribution 4.0 International License.

Last update: 2023-04-05

11.8 Trademark policy

This Trademark Policy is to ensure that users of Percona-branded products or services know that what they receive has really been developed, approved, tested and maintained by Percona. Trademarks help to prevent confusion in the marketplace, by distinguishing one company's or person's products and services from another's.

Percona owns a number of marks, including but not limited to Percona, XtraDB, Percona XtraDB, XtraBackup, Percona XtraBackup, Percona Server, and Percona Live, plus the distinctive visual icons and logos associated with these marks. Both the unregistered and registered marks of Percona are protected.

Use of any Percona trademark in the name, URL, or other identifying characteristic of any product, service, website, or other use is not permitted without Percona's written permission with the following three limited exceptions.

First, you may use the appropriate Percona mark when making a nominative fair use reference to a bona fide Percona product.

Second, when Percona has released a product under a version of the GNU General Public License ("GPL"), you may use the appropriate Percona mark when distributing a verbatim copy of that product in accordance with the terms and conditions of the GPL.

Third, you may use the appropriate Percona mark to refer to a distribution of GPL-released Percona software that has been modified with minor changes for the sole purpose of allowing the software to operate on an operating system or hardware platform for which Percona has not yet released the software, provided that those third party changes do not affect the behavior, functionality, features, design or performance of the software. Users who acquire this Percona-branded software receive substantially exact implementations of the Percona software.

Percona reserves the right to revoke this authorization at any time in its sole discretion. For example, if Percona believes that your modification is beyond the scope of the limited license granted in this Policy or that your use of the Percona mark is detrimental to Percona, Percona will revoke this authorization. Upon revocation, you must immediately cease using the applicable Percona mark. If you do not immediately cease using the Percona mark upon revocation, Percona may take action to protect its rights and interests in the Percona mark. Percona does not grant any license to use any Percona mark for any other modified versions of Percona software; such use will require our prior written permission.

Neither trademark law nor any of the exceptions set forth in this Trademark Policy permit you to truncate, modify or otherwise use any Percona mark as part of your own brand. For example, if XYZ creates a modified version of the Percona Server, XYZ may not brand that modification as "XYZ Percona Server" or "Percona XYZ Server", even if that modification otherwise complies with the third exception noted above.

In all cases, you must comply with applicable law, the underlying license, and this Trademark Policy, as amended from time to time. For instance, any mention of Percona trademarks should include the full trademarked name, with proper spelling and capitalization, along with attribution of ownership to Percona Inc. For example, the full proper name for XtraBackup is Percona XtraBackup. However, it is acceptable to

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent

omit the word “Percona” for brevity on the second and subsequent uses, where such omission does not cause confusion.

In the event of doubt as to any of the conditions or exceptions outlined in this Trademark Policy, please contact trademarks@percona.com for assistance and we will do our very best to be helpful.

Last update: 2023-06-26

<a href="https://www.percona.com/about">Percona LLC and/or its affiliates © 2023 — Cookie Consent