Percona Distribution for MySQL Documentation

8.1.0 (2023-12-23)

Percona Technical Documentation Team

Table of contents

1. Percona Distribution for MySQL 8.1 Documentation

Percona Distribution for MySQL is a single solution with the best and most critical enterprise components from the MySQL open source community, designed and tested to work together. With Percona Server for MySQL as the base server, the distribution brings you the enterprise-grade features for free. The set of carefully selected components helps you operate your MySQL database to meet your application and business needs.

1.1 Features

- **Increased stability and availability** - a set of high-availability and backup options help you ensure your data is saved and available for your business applications.
- **Improved performance and efficiency** - integrated tools help DBAs maintain, manage and monitor the database performance and timely respond to changing demands.
- **Reduced costs** - save on purchasing software licensing by using the distribution - the open-source enterprise-grade solution.
- **Easy-to-integrate with PMM** - benefit from all the features of PMM for monitoring and managing the health of your database.

1.2 Get started

Follow the installation instructions to get started with Percona Distribution for MySQL.

Read more about solutions you can deploy with Percona Distribution for MySQL in High availability solution with Group Replication.

Learn more about what's new in Percona Distribution for MySQL in the release notes.

1.3 Read more

- Deployment variants
- Percona Distribution for MySQL components

1.4 Get expert help

If you need assistance, visit the community forum for comprehensive and free database knowledge, or contact our Percona Database Experts for professional support and services.

Community Forum **Get a Percona Expert**

Last update: 2023-11-28

2. Release notes

2.1 Percona Distribution for MySQL 8.1 release notes index

2.1.1 Get expert help

If you need assistance, visit the community forum for comprehensive and free database knowledge, or contact our Percona Database Experts for professional support and services.

Community Forum **Get a Percona Expert**

Last update: 2023-12-22

2.2 Percona Distribution for MySQL 8.1.0 using Percona Server for MySQL Second Update (2023-12-23)

Percona Distribution for MySQL is the most stable, scalable, and secure open source MySQL distribution based on Percona Server for MySQL. Install Percona Distribution for MySQL.

This update to the release of Percona Distribution for MySQL using the Percona Server for MySQL includes the new version of Percona Toolkit 3.5.7.

2.2.1 Get expert help

If you need assistance, visit the community forum for comprehensive and free database knowledge, or contact our Percona Database Experts for professional support and services.

Community Forum **Get a Percona Expert**

Last update: 2023-12-22

2.3 Percona Distribution for MySQL 8.1.0 using Percona Server for MySQL Update (2023-12-21)

Percona Distribution for MySQL is the most stable, scalable, and secure open source MySQL distribution based on Percona Server for MySQL. Install Percona Distribution for MySQL.

This update to the release of Percona Distribution for MySQL using the Percona Server for MySQL includes the new version of Percona Toolkit 3.5.6 that fixes Go security vulnerability.

2.3.1 Get expert help

If you need assistance, visit the community forum for comprehensive and free database knowledge, or contact our Percona Database Experts for professional support and services.

Community Forum **Get a Percona Expert**

Last update: 2023-12-21

2.4 Percona Distribution for MySQL 8.1.0 using Percona Server for MySQL (2023-11-27)

Percona Distribution for MySQL is the most stable, scalable, and secure open source MySQL distribution based on Percona Server for MySQL. Install Percona Distribution for MySQL.

This release is based on Percona Server for MySQL 8.1.0-1.

2.4.1 Release highlights

Percona Server for MySQL implements telemetry that fills in the gaps in our understanding of how you use Percona Server for MySQL 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 information in the Telemetry on Percona Server fo MySQL document.

The following user-defined function (UDF) shared objects (so) are converted to components:

- The `data_masking` plugin converted into the `component_masking_functions` component
- The `binlogs_utils_udf` UDF shared object (.so) converted to the `component_binlog_utils` component
- The `percona-udf` UDF shared object (.so) converted to the `component_percona-udf` component

A user does not need to execute a separate `CREATE FUNCTION ... SONAME ...` statement for each function. Installing the components with the `INSTALL COMPONENT 'file://componenet_xxx` statement performs the auto-registration operations.

The `keyring_vault` plugin converted into the `component_keyring_vault` component. This conversion aligns the keyring_vault with the KMIP and KMS keyrings and supports “ALTER INSTANCE RELOAD KEYRING” to update the configuration automatically.

The `audit_log_filter` plugin converted to the `component_audit_log_filter` component. The following changes are also available:

- Adds the `mysql_event_tracking_parse` audit log event
- Reworked, optimized, and reorganized the audit event data members
- Data deduplication within the audit event data members

The current version of `percona-release` does not support the `setup` subcommand with the `pdps-8x-innovation` and `pdps-8.1.0` repositories. Use `percona-release enable` instead. The support of the `pdps-8x-innovation` and `pdps-8.1.0` repositories for the `setup` subcommand will be added in the next release of `percona-release`.

The PS 8.1.0 MTR suites are reorganized. The existing percona-specific MTR test cases are regrouped and put into separate test suites:

- component_encryption_udf
- percona
- percona_innodb

Improvements and bug fixes introduced by Oracle for MySQL 8.1 and included in Percona Server for MySQL are the following:

- The `EXPLAIN FORMAT=JSON` can output the data to a user variable.
- New messages written to the MySQL error log during shutdown:
    - Startup and shutdown log messages, including when the server was started with `--initialize`
    - Start and end of shutdown phases for plugins and components
    - Start-of-phase and end-of-phase messages for connection closing phases
    - The number and ID of threads still alive after being forcibly disconnected and potentially causing a wait

Find the full list of bug fixes and changes in the MySQL 8.1 Release Notes.

2.4.2 Deprecation or removal

- The `mysql_native_password` authentication plugin is deprecated and subject to removal in a future version.
- The TokuDB is removed. The following items are also removed:
    - Percona-TokuBackup submodule
    - PerconaFT submodule
    - TokuDB storage engine code
    - TokuDB MTR test suites
    - plugin/tokudb-backup-plugin
- The MyRocks ZenFS is removed. The following items are also removed:
    - zenfs submodule
    - libzdb submodule
    - RocksDB MTR changes are reverted
- Travis CI integration
- Supporting `readline` as a alternative to editline library is removed.
- The `audit_log` (audit version 1) plugin is removed
- The “include/ext” pre-C++17 compatibility headers are removed.
- The `keyring_vault` plugin is removed.

- The `data_masking` UDF shared object (.so) is removed.
- The `binlog_utils_udf` UDF shared object (.so) is removed.
- The `percona_udf` UDF shared object (.so) is removed.

2.4.3 Platform support

- Percona Server for MySQL 8.1.0-1 is not supported on Ubuntu 18.04.

2.4.4 Supplied components

Review each component's release notes for What's new, improvements, or bug fixes. The following is a list of the components supplied with the Percona Server for MySQL-based variation of the Percona Distribution for MySQL:

Component	Version	Description
Orchestrator	3.2.6-11	The replication topology manager for Percona Server for MySQL
ProxySQL	2.5.5	A high performance, high-availability, protocol-aware proxy for MySQL
Percona XtraBackup	8.1.0	An open-source hot backup utility for MySQL-based servers
Percona Toolkit	3.5.5	The set of scripts to simplify and optimize database operation
MySQL Shell	8.1.0	An advanced client and code editor for MySQL Server
MySQL Router	8.1.0	Lightweight middleware that provides transparent routing between your application and back-end MySQL servers

2.4.5 Get expert help

If you need assistance, visit the community forum for comprehensive and free database knowledge, or contact our Percona Database Experts for professional support and services.

Community Forum **Get a Percona Expert**

Last update: 2023-11-28

3. Discover Percona Distribution for MySQL

3.1 Components

Percona Distribution for MySQL consists of the following **components**:

- Percona Server for MySQL is a drop-in replacement for MySQL Community Edition with the enterprise-grade features embedded by Percona.
- Percona XtraBackup is an open-source hot backup utility for MySQL-based servers that doesn't lock your database during the backup.
- Orchestrator is the replication topology manager for *Percona Server for MySQL*.
- ProxySQL is a high performance, high-availability, protocol-aware proxy for MySQL.
- Percona Toolkit is the set of scripts to simplify and optimize database operation.
- MySQL Shell is an advanced client and code editor for MySQL Server.
- MySQL Router is lightweight middleware that provides transparent routing between your application and back-end MySQL servers.

3.1.1 Get expert help

If you need assistance, visit the community forum for comprehensive and free database knowledge, or contact our Percona Database Experts for professional support and services.

Community Forum **Get a Percona Expert**

Last update: 2023-11-13

3.2 Deployment variants

Percona Distribution for MySQL provides two deployment variants: one is *Percona Server for MySQL*-based with asynchronous replication and another one is *Percona Server for MySQL*-based with group replication. The table below lists what components are available with Percona Server for MySQL:

Components	Percona Server for MySQL
Orchestrator	YES
HAProxy	NO
ProxySQL	YES
Percona XtraBackup	YES
Percona Toolkit	YES
MySQL Shell	YES

Components	Percona Server for MySQL
MySQL Router	YES

3.2.1 What deployment variant to choose?

The **Percona Server-based deployment variant** with asynchronous replication utilizes the primary / secondary replication model. It enables you to create geographically distributed infrastructures with the support for disaster recovery. However, this deployment variant does not guarantee data consistency on all nodes at the given moment and provides high availability of up to 4 nines.

The **Percona Server-based deployment variant** with Group Replication enables you to create fault-tolerant systems with redundancy by replicating the system state to a set of servers. *Percona Server for MySQL*-based deployment with Group Replication offers a high grade of high availability (4-5 nines) and almost instant fail over when associated with a proxy.

3.2.2 Get expert help

If you need assistance, visit the community forum for comprehensive and free database knowledge, or contact our Percona Database Experts for professional support and services.

Community Forum **Get a Percona Expert**

Last update: 2023-11-13

4. Install and update

4.1 Install Percona Distribution for MySQL

We recommend to install Percona Distribution for MySQL from Percona repositories using the package manager of your operating system:

- `apt` - for Debian and Ubuntu Linux
- `yum` - for Red Hat Enterprise Linux and compatible Linux derivatives

Find the full list of supported platforms on the Percona Software and Platform Lifecycle page.

pository overview: Major and Minor repositories

Percona provides two repositories for every deployment variant of Percona Distribution for MySQL.

The *Major Release repository* includes the latest version packages (for example, `pdps-8x-innovation`). Whenever a package is updated, the package manager of your operating system detects that and prompts you to update. As long as you update all Distribution packages at the same time, you can ensure that the packages you're using have been tested and verified by *Percona*. Installing Percona Distribution for MySQL from the Major Release Repository is the recommended method.

The *Minor Release repository* includes a particular minor release of the database and all of the packages that were tested and verified to work with that minor release (for example, `pdps-8.1.0`). You may choose to install Percona Distribution for MySQL from the Minor Release repository if you have decided to standardize on a particular release which has passed rigorous testing procedures and which has been verified to work with your applications. This allows you to deploy to a new host and ensure that you'll be using the same version of all the Distribution packages, even if newer releases exist in other repositories.

The disadvantage of using a Minor Release repository is that you are locked in this particular release. When potentially critical fixes are released in a later minor version of the database, you will not be prompted for an upgrade by the package manager of your operating system. You would need to change the configured repository in order to install the upgrade.

4.1.1 Prerequisites

To install Percona software, you need to configure the required repository. To simplify this process, use the `percona-release` repository management tool.

1. Install GnuPG and curl

```
$ sudo apt install gnupg2 curl
```

2. Install percona-release. If you have it installed, update percona-release to the latest version.

4.1.2 Procedure

On Debian and Ubuntu Linux   On Red Hat Enterprise Linux and derivatives

Important

Run the following commands as the root user or via `sudo`.

Enable Percona repository

To enable the desired repository, we recommend to use the `enable` subcommand of `percona-release`.

```
$ sudo percona-release enable pdps-8x-innovation
```

Tip

To enable the minor version repository, use the following command:

```
$ sudo percona-release enable pdps-8.1.0
```

Install Percona Distribution for MySQL packages

1. Install *Percona Server for MySQL*:

   ```
   $ sudo apt install percona-server-server
   ```

2. Install the components. Use the commands below to install the required components:

   Install Percona XtraBackup:

   ```
   $ sudo apt install percona-xtrabackup-81
   ```

   Install Percona Toolkit:

   ```
   $ sudo apt install percona-toolkit
   ```

   Install Orchestrator:

   ```
   $ sudo apt install percona-orchestrator percona-orchestrator-cli percona-
   orchestrator-client
   ```

   Install MySQL Shell:

   ```
   $ sudo apt install percona-mysql-shell
   ```

   Install ProxySQL:

   ```
   $ sudo apt install proxysql2
   ```

   Install MySQL Router:

   ```
   $ sudo apt install percona-mysql-router
   ```

Run Percona Distribution for MySQL

Percona Distribution for MySQL is not started automatically on Red Hat Enterprise Linux and CentOS after the installation is complete.

Start it manually using the following command:

```
$ sudo systemctl start mysql
```

Confirm that the service is running:

```
$ sudo systemctl status mysql
```

Stop the service:

```
$ sudo systemctl stop mysql
```

4.1.3 Get expert help

If you need assistance, visit the community forum for comprehensive and free database knowledge, or contact our Percona Database Experts for professional support and services.

Community Forum   Get a Percona Expert

Last update: 2023-11-28

4.2 Upgrade Percona Distribution for MySQL

Minor releases include bug fixes and feature enhancements. We recommend to have Percona Distribution for MySQL updated to the latest version.

Though minor releases don't change the behavior, even a minor upgrade is a risky process. We recommend to back up your data before upgrading.

4.2.1 Preconditions

To upgrade Percona Distribution for MySQL, install the `percona-release` repository management tool or update it to the latest version.

4.2.2 Procedure

Important

Run the following commands as the root user or via `sudo`.

1. Enable Percona repository

The Major Release repository automatically includes new version packages of Percona Distribution for MySQL. If you installed Percona Distribution for MySQL from a Minor Release repository, enable the new version repository:

```
$ sudo percona-release setup pdps-XXX
```

where `XXX` is the required version.

Read more about major and Minor release repositories in Repository overview.

2. Stop `mysql` service

```
$ sudo systemctl mysql stop
```

3. Install new version packages using the package manager of your operating system.
4. Restart `mysql` service:

```
$ sudo systemctl mysql start
```

To upgrade the components, refer to Installing Percona Distribution for MySQL for installation instructions relevant to your operating system.

4.2.3 Get expert help

If you need assistance, visit the community forum for comprehensive and free database knowledge, or contact our Percona Database Experts for professional support and services.

Community Forum **Get a Percona Expert**

Last update: 2023-10-04

4.3 Downgrade Percona Distribution for MySQL

Following the MySQL downgrade policy, the downgrade to a previous version of Percona Distribution of MySQL is not supported.

4.3.1 Get expert help

If you need assistance, visit the community forum for comprehensive and free database knowledge, or contact our Percona Database Experts for professional support and services.

Community Forum **Get a Percona Expert**

Last update: 2023-11-13

5. Solutions - high availability

5.1 High availability solution with Group Replication

Every architecture and deployment depends on customer requirements and application demands for high availability and the estimated level of usage. For example, using a high read or a high write application, or both with 99.999% availability.

This guide gives architecture and deployment recommendations along with a technical overview for a solution that provides a high level of high availability and assumes the usage of high read / write applications (20K or more queries per second). It also provides step-by-step deployment guidelines.

This solution assumes the use of *Percona Server for MySQL* based deployment variant of Percona Distribution for MySQL with Group Replication.

5.1.1 High availability overview

How to measure availability and at what point does it become “high” availability?

Generally speaking, the measurement of availability is done by establishing a measurement time frame and dividing it by the time that it was available. This ratio will rarely be 1, which is equal to 100% availability. A solution is considered to be highly available if it is at least 99% or “two nines” available.

The following table provides downtime calculations per high availability level:

Availability, %	Downtime per year	Downtime per month	Downtime per week	Downtime per day
99% (“two nines”)	3.65 days	7.31 hours	1.68 hours	14.40 minutes
99.5% (“two nines five”)	1.83 days	3.65 hours	50.40 minutes	7.20 minutes
99.9% (“three nines”)	8.77 hours	43.83 minutes	10.08 minutes	1.44 minutes
99.95% (“three nines five”)	4.38 hours	21.92 minutes	5.04 minutes	43.20 seconds
99.99% (“four nines”)	52.60 minutes	4.38 minutes	1.01 minutes	8.64 seconds
99.995% (“four nines five”)	26.30 minutes	2.19 minutes	30.24 seconds	4.32 seconds
99.999% (“five nines”)	5.26 minutes	26.30 seconds	6.05 seconds	864.00 milliseconds

How is high availability achieved?

There are three key components to achieve high availability:

- **Infrastructure** - this is the physical or virtual hardware that database systems rely on to run. Without enough infrastructure (VM's, networking, etc.), there cannot be high availability. The easiest example is: `there is no way to make a single server highly available`.
- **Topology management** - this is the software management related specifically to the database and managing its ability to stay consistent in the event of a failure. Many clustering or synchronous replication solutions offer this capability out of the box. However, asynchronous replication is handled by additional software.
- **Connection management** - this is the software management related specifically to the networking and connectivity aspect of the database. Clustering solutions typically bundle with a connection manager. However, in asynchronous clusters, deploying a connection manager is mandatory for high availability.

This solution is based on a tightly coupled database cluster. It offers a high availability level of 99.995% when coupled with the Group Replication setting `group_replication_consistency=AFTER`.

image

5.1.2 Failovers

A database failure or configuration change that requires a restart should not affect the stability of the database infrastructure, if it is properly planned and architected. Failovers are an integral part of a stability strategy and aligning the business requirements for availability and uptime with failover methodologies is critical.

The following are the three main types of failovers that can occur in database environments:

- **Planned failover**. This is a failover that has been scheduled in advance or occurs at a regular interval. There can be many reasons for planned failovers including patching, large data operations, retiring existing infrastructure, or simply to test the failover strategy.
- **Unplanned failover**. This is what occurs when a database has unexpectedly become unresponsive or experiences instability. An unplanned failover could also include emergency changes that do not fall under the planned failover cadence or scheduling parameters. Unplanned failovers are generally considered higher risk operations due to the high stress and high potential for data corruption or data fragmentation.
- **Regional or disaster recovery (DR) failover**. Unplanned failovers still work with the assumption that additional database infrastructure is immediately available and in a usable state. However, in a regional or DR failover, it is assumed that there is a large scale infrastructure outage which requires the business to move its operations away from its current availability zone.

5.1.3 Maintenance windows

Major vs Minor maintenance

Although it may not be obvious at first, not all maintenance activities are created equal and do not have the same dependencies. It is good to separate maintenance that demands downtime or failover from maintenance that can be done without impacting those important stability metrics.

When defining these maintenance dependencies, there can be a change in the actual maintenance process that allows for a different cadence.

Maintenance without service interruption

It is possible to cover both major and minor maintenance without service interruption with rolling restart and using proper version upgrade.

5.1.4 Uptime

When referring to database stability, uptime is likely the largest indicator of stability and often is the most obvious symptom of an unstable database environment. Uptime is composed of three key components and, contrary to common perception, is based on what happens when the database software cannot take incoming requests rather than maintain the ability to take requests with errors.

The uptime components are:

- **Recovery Time Objective (RTO)**

RTO can be characterized by a simple question “How long can the business sustain a database outage?” Once the business is aligned with a minimum viable recovery time objective, it is much more straightforward to plan and invest in the infrastructure required to meet that requirement. It is important to acknowledge that while everyone desires 100% uptime, there need to be realistic expectations that align with the business needs and not a technical desire.

- **Recovery Point Objective (RPO)**

There is a big distinction between the Recovery Point and the Recovery Time for a database infrastructure. The database can be available, but not to the exact state that it was when it became unavailable. That is where Recovery Point comes in. The question to ask here is “How much data can the business lose during a database outage?” All businesses have their own requirements here yet it is always the goal to never sustain any data loss. But this is framed in the worst case scenario, how much data could be lost and the business maintains the ability to continue.

- **Disaster recovery**

RTO and RPO are great for unplanned outages or small scale hiccups to the infrastructure. Disaster recovery is a major large scale outage not strictly for the database infrastructure. How capable is the business of restarting operations with the assumption that all resources are completely unavailable in the main availability zone? The assumption here is that there is no viable restoration point or time that aligns with the business requirements. While each disaster recovery scenario is unique based on available infrastructure, backup strategy and technology stack, there are some common threads for any scenario.

The described solution **helps improve uptime**. It will help you to significantly reduce both RPO and RTO. Given the tightly coupled cluster solution approach, the failure of a single node will not result in service interruption.

Increasing the number of nodes will also improve the cluster resilience by the formula:

```
F = (N -1) / 2
```

where:

- `F` is the number of admissible failures
- `N` is the number of nodes in the cluster.

Example

- In a cluster of 5 nodes, F = (5 - 1)/2 = 2. The cluster can support up to 2 failures.
- In a cluster of 4 nodes, F = (4 - 1)/2 = 1. The cluster can support up to 1 failure.

This solution also allows for a more restrictive backup policy, dedicating a node to the backup cycle, which will help in keeping RPO low.

As previously mentioned, disaster recovery is not covered by default by this solution. It will require an additional replication setup and controller.

Based on the material from Percona Database Performance Blog

This document is based on the blog post Percona Distribution for MySQL: High Availability with Group Replication Solution by *Marco Tusa*

5.1.5 Get expert help

If you need assistance, visit the community forum for comprehensive and free database knowledge, or contact our Percona Database Experts for professional support and services.

Community Forum **Get a Percona Expert**

Last update: 2023-11-13

5.2 Architecture and components

The following is the architecture layout for *Percona Server for MySQL* based deployment variant of Percona Distribution for MySQL with Group Replication.

5.2.1 Architecture layout

image

Components

The architecture is composed of two main layers:

- Connection and distribution layer
- Relational Database Management System (RDBMS) layer

CONNECTION AND DISTRIBUTION LAYER

The connection and distribution layer consists of the following:

- **Application to proxy redirection mechanism.** This mechanism can be anything from a Virtual IP managed by Keepalived local service to a DNS resolution service like Amazon Route 53. The mechanism's function is to redirect the traffic to the active Proxy node.
- **Proxy connection distribution.** The distribution consists of two or more nodes and its role is to redirect the traffic to the active nodes of the Group Replication cluster. In cases like ProxySQL where the proxy is a level 7 proxy and can perform a read / write split, this layer is also in charge of redirecting writes to the Primary node and reads to the replicas, and of high availability to prevent a single point of failure.

RDBMS LAYER

The data layer consists of the following:

- **Primary (or source) node** serving write requests. This is the node that accepts writes and DDL modifications. Data will be processed following the ACID (atomicity, consistency, isolation, durability) model and replicated to all other nodes.
- **Replica nodes** serving read requests. Some replica nodes can be elected Primary in case of the Primary node's failure. A replica node should be able to leave and join back to a healthy cluster without impacting the service.
- **Replication mechanism** distributing changes across nodes. In this solution, it is done with Group Replication. Group Replication is a tightly coupled solution, which means that the database cluster is based on a Datacentric approach (single state of the data, distributed commit). In this case, the data is consistent in time across nodes though this type of replication requires a high performant link. Given that, the main Group Replication mechanism does not implicitly support Disaster Recovery (DR) and geographic distribution is not permitted.

The node characteristics such as CPU/RAM/Storage are not relevant to the solution design. They must reflect the estimated workload that the solution will have to cover, and this is a case by case identification.

However, it is important that all nodes that are part of the cluster must have the same characteristics. Otherwise, the cluster is imbalanced and services will be affected.

As a generic indication we recommend using nodes with at least 8 cores and 16GB RAM when in production.

5.2.2 Get expert help

If you need assistance, visit the community forum for comprehensive and free database knowledge, or contact our Percona Database Experts for professional support and services.

Community Forum **Get a Percona Expert**

Last update: 2023-11-13

5.3 Measurement and monitoring

To ensure that database infrastructure is performing as intended or at its best, specific metrics need to be measured and alerts are to be raised when some of these metrics are not in line with expectations. A periodic review of these measurements is also encouraged to promote stability and understand potential risks associated with the database infrastructure.

The following are the 3 aspects of database performance measurement and monitoring:

- **Measurement** - to understand how a database infrastructure is performing, multiple aspects of the infrastructure need to be measured. With measurement it's important to understand the impact of the sample sizes, sample timing, and sample types.
- **Metrics** - metrics refer to the actual parts of the database infrastructure being measured. When we discuss metrics, more isn't always better as it could introduce unintentional noise or make troubleshooting overly burdensome.
- **Alerting** - when one or many metrics of the database infrastructure is not within a normal or acceptable range, an alert should be generated so that the team responsible for the appropriate portion of the database infrastructure can investigate and remedy it.

Monitoring and measurement for this solution are covered by Percona Monitoring and Management. It has a specific dashboard to monitor the Group Replication state and cluster status as a whole. For more information, read Percona Monitoring and Management Documentation.

5.3.1 Get expert help

If you need assistance, visit the community forum for comprehensive and free database knowledge, or contact our Percona Database Experts for professional support and services.

Community Forum **Get a Percona Expert**

Last update: 2023-10-04

5.4 Deploying high availability solution with Group Replication

This document provides step-by-step instructions on how to deploy high availability solution with Group Replication.

5.4.1 Preconditions

We will use the following elements:

- 1 Virtual IP for ProxySQL failover - 192.168.4.194
- 2 ProxySQL nodes
    - Proxy1 192.168.4.191
    - Proxy2 192.168.4.192
- 4 MySQL nodes in Single Primary mode
    - Gr1 192.168.4.81 - Initial Primary
    - Gr2 192.168.4.82 - Replica / failover
    - Gr3 192.168.4.83 - Replica / failover
    - Gr4 192.168.4.84 - Replica / Backup
- All of the following ports must be open if a firewall is in place or any other restriction like AppArmor or SELinux.
    - ProxySQL:
        - 6033
        - 6032
        - 3306
    - MySQL - Group Replication:
        - 3306
        - 33060
        - 33061

5.4.2 Nodes configuration

Preparation

1. Install Percona Server-based variant of Percona Distribution for MySQL on each MySQL node (Gr1-Gr4).
2. Make sure that all the nodes use the same time-zone and time

```
$ date
Tue Aug 18 08:22:12 EDT 2020
```

3. Also check that `ntpd` service is present and enabled
4. Make sure that each node resolves the other nodes by name

```
for i in 1 2 3 4 ; do ping -c 1 gr$i > /dev/null;echo $?; done
```

   If nodes aren't able to resolve, add the entries in the `/etc/hosts` file.

5. After instances are up and running, check *Percona Server for MySQL* version on each node:

```
mysql>\s
--------------
/opt/mysql_templates/PS-8P/bin/mysql  Ver 8.1.0-1 for Linux on x86_64 (Percona
Server (GPL), Release 11, Revision 159f0eb)
```

Step 1 Create an administration user

1. Create a user for administration. We will use the user `dba` in our setup:

```
CREATE user dba@localhost identified by 'dbapw';
CREATE user dba@'192.168.%' identified by 'dbapw';

GRANT ALL on *.* to dba@localhost with grant option;
GRANT ALL on *.* to dba@'192.168.%' with grant option;
```

Log out from the client as the root user and log in as the `dba` user.

2. Make sure to have a good and unique SERVER_ID value:

```
mysql> show global variables like 'server_id';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| server_id     |     1 |
+---------------+-------+
1 row in set (0.01 sec)
```

The `server_id` value must be unique on each node

Step 2. Add Group Replication settings

1. Stop all the nodes

```
$ service mysql stop
```

2. In the `my.cnf` configuration file, add the following:

```
#####################
#Replication + binlog settings
#####################
auto-increment-increment                                    =1
auto-increment-offset                                       =1

log-bin                                             =<path_to_logs>/binlog
log-bin-index                                       =binlog.index
binlog-checksum                                             =NONE
binlog-format                                               =ROW
binlog-row-image                                            =FULL
log-slave-updates                                                             =1
binlog-transaction-dependency-tracking                   =WRITESET_SESSION

enforce-gtid-consistency                                    =TRUE
```

```
gtid-mode                                                   =ON

master-info-file                                            =master.info
master-info-repository                                      =TABLE
relay_log_info_repository                                   =TABLE
relay-log                                            =<path_to_logs>/relay

sync-binlog                                                 =1

### SLAVE SECTION
skip-slave-start
slave-parallel-type                                        = LOGICAL_CLOCK
slave-parallel-workers                                      = 4
slave-preserve-commit-order                                 = 1


######################################
#Group Replication
######################################
plugin_load_add                                      ='group_replication.so'
plugin-load-add                                      ='mysql_clone.so'
group_replication_group_name       ="aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa"
#<-- Not good. Use something
                                         that will help you to identify the GR
transactions and from where they come from IE "dc1euz1-aaaa-aaaa-aaaa-
aaaaaaaaaaaa"
group_replication_start_on_boot                       =off
group_replication_local_address                       =
"192.168.4.81/2/3/4:33061"  <---- CHANGE THIS TO MATCH EACH NODE LOCAL IP
group_replication_group_seeds                         =
"192.168.4.81:33061,192.168.4.82:33061,192.168.4.83:33061,192.168.4.84:33061"
group_replication_bootstrap_group                     = off
transaction-write-set-extraction                             = XXHASH64
```

3. Restart all nodes:

```
$ service mysql start
```

4. Connect to the nodes

Step 3. Create a replication user

1. On every node, create a user for replication

```
SET SQL_LOG_BIN=0;
 CREATE USER replica@'192.168.4.%' IDENTIFIED BY 'replicapw';   #<--- Please
note the filter by IP is more restrictive
 GRANT REPLICATION SLAVE ON *.* TO replica@'192.168.4.%';
 FLUSH PRIVILEGES;
 SET SQL_LOG_BIN=1;
```

2. Link the nodes with the replication channel.

```
CHANGE MASTER TO MASTER_USER='replica', MASTER_PASSWORD='replicapw' FOR CHANNEL
'group_replication_recovery';
```

Run this command on all nodes.

3. Check the current status:

```
(dba@node1) [(none)]>\u performance_schema
    (dba@node1) [performance_schema]>show tables like '%repl%';
    +-------------------------------------------+
    | Tables_in_performance_schema (%repl%)     |
    +-------------------------------------------+
    | replication_applier_configuration         |
    | replication_applier_filters               |
    | replication_applier_global_filters        |
    | replication_applier_status                |
    | replication_applier_status_by_coordinator |
    | replication_applier_status_by_worker      |
    | replication_connection_configuration      |
    | replication_connection_status             |
    | replication_group_member_stats            |
    | replication_group_members                 | <------------------------
    +-------------------------------------------+

   (dba@node1) [performance_schema]>select * from replication_group_members\G
CHANNEL_NAME: group_replication_applier
     MEMBER_ID:
   MEMBER_HOST:
   MEMBER_PORT:
  MEMBER_STATE:
   MEMBER_ROLE: OFFLINE
MEMBER_VERSION:
1 row in set (0.00 sec)
```

At this stage, you should be able to start the first (Primary) cluster node.

4. Start the Primary node (Gr1) and enable Group Replication:

```
(dba@node1)[none]> SET GLOBAL group_replication_bootstrap_group=ON;
(dba@node1)[none]> START GROUP_REPLICATION;
(dba@node1)[none]> SET GLOBAL group_replication_bootstrap_group=OFF;
```

5. Check if the node registered correctly:

```
(dba@node1) [none]>select * from performance_schema.replication_group_members\G
     CHANNEL_NAME: group_replication_applier
     MEMBER_ID: 90a353b8-e6dc-11ea-98fa-08002734ed50
   MEMBER_HOST: gr1
   MEMBER_PORT: 3306
  MEMBER_STATE: ONLINE
   MEMBER_ROLE: PRIMARY
MEMBER_VERSION: 8.1.0
```

6. Once the Primary node is running, connect to the secondary node (Gr2 node) and enable Group Replication:

```
(dba@node2) [none]>START GROUP_REPLICATION;
Query OK, 0 rows affected (4.60 sec)
```

7. Check if the secondary node registered correctly:

```
(dba@node2) [performance_schema]>select * from replication_group_members\G
*************************** 1. row ***************************
  CHANNEL_NAME: group_replication_applier
     MEMBER_ID: 58ffd118-e6dc-11ea-8af8-08002734ed50
   MEMBER_HOST: gr2
   MEMBER_PORT: 3306
  MEMBER_STATE: ONLINE
   MEMBER_ROLE: SECONDARY
MEMBER_VERSION: 8.1.0
*************************** 2. row ***************************
  CHANNEL_NAME: group_replication_applier
     MEMBER_ID: 90a353b8-e6dc-11ea-98fa-08002734ed50
   MEMBER_HOST: gr1
   MEMBER_PORT: 3306
  MEMBER_STATE: ONLINE
   MEMBER_ROLE: PRIMARY
MEMBER_VERSION: 8.1.0
```

8. Test the replication:

   • On the Primary node, run the following command:

```
(dba@node1) [performance_schema]>create schema test;
Query OK, 1 row affected (0.76 sec)

(dba@node1) [performance_schema]>\u test
Database changed

(dba@node1) [test]>create table test1 (`id` int auto_increment primary key);
Query OK, 0 rows affected (0.32 sec)

(dba@node1) [test]>insert into test1 values(null);
Query OK, 1 row affected (0.34 sec)
```

   • On the secondary node:

```
(dba@node2) [performance_schema]>use \test
 Database changed
 (dba@node2) [test]>select * from test1;
 +----+
 | id |
 +----+
 |  1 |
 +----+
 1 row in set (0.00 sec)
```

9. Start Group Replication on the remaining nodes

```
(dba@node3) [performance_schema]>START GROUP_REPLICATION;
(dba@node4) [performance_schema]>START GROUP_REPLICATION;
```

5.4.3 Proxy setup

Step 1. Installation

1. Install ProxySQL. In our example, we install ProxySQL on Proxy1 192.168.4.191 and Proxy2 192.168.4.192 nodes.
2. Create the monitoring user on MySQL Group Replication nodes:

```
create user monitor@'192.168.4.%' identified by 'monitor';
grant usage on *.* to 'monitor'@'192.168.4.%';
grant select on sys.* to 'monitor'@'192.168.4.%';
```

3. Define basic variables:

```
update global_variables set Variable_Value='admin:admin;cluster1:clusterpass'
where Variable_name='admin-admin_credentials';
update global_variables set variable_value='cluster1' where
variable_name='admin-cluster_username';
update global_variables set variable_value='clusterpass' where
variable_name='admin-cluster_password';
update global_variables set Variable_Value=0  where Variable_name='mysql-
hostgroup_manager_verbose';
update global_variables set Variable_Value='true'  where Variable_name='mysql-
query_digests_normalize_digest_text';
update global_variables set Variable_Value='8.1.0'  where Variable_name='mysql-
server_version';
update global_variables set Variable_Value='utf8'  where Variable_name='mysql-
default_charset';
update global_variables set Variable_Value=300  where Variable_name='mysql-
tcp_keepalive_time';
update global_variables set Variable_Value='true'  where Variable_name='mysql-
use_tcp_keepalive';
update global_variables set Variable_Value='true'  where Variable_name='mysql-
verbose_query_error';
update global_variables set Variable_Value='true'  where Variable_name='mysql-
show_processlist_extended';
update global_variables set Variable_Value=50000  where Variable_name='mysql-
max_stmts_cache';
update global_variables set Variable_Value='false'  where Variable_name='admin-
web_enabled';
update global_variables set Variable_Value='0'  where Variable_name='mysql-
set_query_lock_on_hostgroup';

load admin variables to run;save admin variables to disk;
load mysql variables to run;save mysql variables to disk;
```

Note

The user name and password need to reflect your standards. The ones used above are just an example.

4. Set up the nodes as a cluster:

```
INSERT INTO proxysql_servers (hostname,port,weight,comment)
VALUES('192.168.4.191',6032,100,'PRIMARY');
```

```
INSERT INTO proxysql_servers (hostname,port,weight,comment)
VALUES('192.168.4.192',6032,100,'SECONDARY');
load proxysql servers to run;save proxysql servers to disk;
```

Step 2. Define users, servers and query rules for read / write split

1. Create one or more valid users. Define these user(s). For example, if you have a user named `app_gr` with the password `test`, and that has access to your Group Replication cluster, the command to define the user is the following:

```
insert into mysql_users
(username,password,active,default_hostgroup,default_schema,transaction_persistent,
values ('app_gr','test',1,400,'mysql',1,'application test user GR');
LOAD MYSQL USERS TO RUNTIME;SAVE MYSQL USERS TO DISK;
```

2. Define servers:

```
INSERT INTO mysql_servers
(hostname,hostgroup_id,port,weight,max_connections,comment) VALUES
('192.168.4.81',400,3306,10000,2000,'GR1');
INSERT INTO mysql_servers
(hostname,hostgroup_id,port,weight,max_connections,comment) VALUES
('192.168.4.81',401,3306,100,2000,'GR1');
INSERT INTO mysql_servers
(hostname,hostgroup_id,port,weight,max_connections,comment) VALUES
('192.168.4.82',401,3306,10000,2000,'GR2');
INSERT INTO mysql_servers
(hostname,hostgroup_id,port,weight,max_connections,comment) VALUES
('192.168.4.83',401,3306,10000,2000,'GR2');
INSERT INTO mysql_servers
(hostname,hostgroup_id,port,weight,max_connections,comment) VALUES
('192.168.4.84',401,3306,1,2000,'GR2');
LOAD MYSQL SERVERS TO RUNTIME; SAVE MYSQL SERVERS TO DISK;
```

3. Define query rules to get read / write split:

```
INSERT INTO mysql_query_rules
(rule_id,proxy_port,username,destination_hostgroup,active,retries,match_digest,app
values(4040,6033,'app_gr',400,1,3,'^SELECT.*FOR UPDATE',1);
INSERT INTO mysql_query_rules
(rule_id,proxy_port,username,destination_hostgroup,active,retries,match_digest,mul
values(4042,6033,'app_gr',401,1,3,'^SELECT.*$',1,1);
LOAD MYSQL QUERY RULES TO RUN;SAVE MYSQL QUERY RULES TO DISK;
```

Step 3. Create a view in SYS schema

Once all the configuration is ready, we need to have a special view in the SYS schema in Percona server nodes. Find the view working for the server version 8 and above here.

Run that sql on the **Primary** node of the Group Replication cluster.

Step 4. Activate support for Group Replication in ProxySQL

To activate the native support for Group Replication in ProxySQL, we will use the following group definition:

```
Writer HG-> 400
Reader HG-> 401
BackupW HG-> 402
Offline HG-> 9401
```

```
INSERT INTO mysql_group_replication_hostgroups
(writer_hostgroup,backup_writer_hostgroup,reader_hostgroup,
offline_hostgroup,active,max_writers,writer_is_also_reader,max_transactions_behind)
values (400,402,401,9401,1,1,1,100);
LOAD MYSQL SERVERS TO RUNTIME; SAVE MYSQL SERVERS TO DISK;
```

COMMENTS ABOUT PARAMETERS

To obtain the most reliable results, we recommend setting the number of writers always to 1, and `writer_is_also_reader` to 1 as well.

```
max_writers: 1
writer_is_also_reader: 1
```

The `max_transactions_behind` is a subjective parameter that you should calculate on the basis of your needs. If, for instance, you cannot have a stale read, it will be safe to set this value to a low number (i.e. 50) and to set in all Group Replication nodes:

```
set global group_replication_consistency=AFTER;
```

If instead, you have no issue or strict requirements about some stale read, you can relax the parameter and ignore the `group_replication_consistency` setting. Our recommended setting is `group_replication_consistency=AFTER` and `max_transactions_behind: 100`.

See also

ProxySQL Documentation: mysql_group_replication_hostgroups

Step 5. Enable high availability for ProxySQL

`keepalived` will be used to enable High Availability for ProxySQL.

1. Install `keepalived` on each ProxySQL node using the package manager of your operating system:

   on Debian/Ubuntu **On RHEL/derivatives**

   ```
   $ sudo apt install -y keepalived
   ```

   ```
   $ sudo yum install -y keepalived
   ```

2. Modify the `/etc/keepalived/keepalived.conf` file accordingly to your setup. In our case:

    - Proxy1 192.168.4.0/24 dev enp0s9 proto kernel scope link src 192.168.4.191
    - Proxy2 192.168.4.0/24 dev enp0s9 proto kernel scope link src 192.168.4.192
    - VIP 192.168.4.194

    Let's say Proxy1 is the primary node while Proxy2 is the secondary node.

    Given that, the config file looks as follows:

```
global_defs {
  # Keepalived process identifier
  router_id  proxy_HA
}
# Script used to check if Proxy is running
vrrp_script check_proxy {
  script "killall -0 proxysql"
  interval 2
  weight 2
}
# Virtual interface
# The priority specifies the order in which the assigned interface to take over
in a failover
vrrp_instance VI_01 {
  state MASTER
  interface enp0s9
  virtual_router_id 51
  priority 100  <----- This needs to be different for each ProxySQL node, like
100/99

  # The virtual ip address shared between the two load balancers
  virtual_ipaddress {
   192.168.4.194  dev enp0s9
  }
  track_script {
    check_proxy
  }
}
```

3. Start the `keepalived` service. From now on, the VIP will be associated with the Proxy1 unless the service is down.

5.4.4 Disaster recovery implementation

The implementation of a DR (Disaster Recovery) site will follow the same direction provided for the main site. There are only some generic rules to follow:

- A DR site should be located in a different geographic location than the main site (several hundred kilometers/miles away).
- The connection link between the main site and the DR site can only be established using *asynchronous replication* (standard MySQL replication setup).

5.4.5 Monitoring

Using Percona Management and Monitoring (PMM)

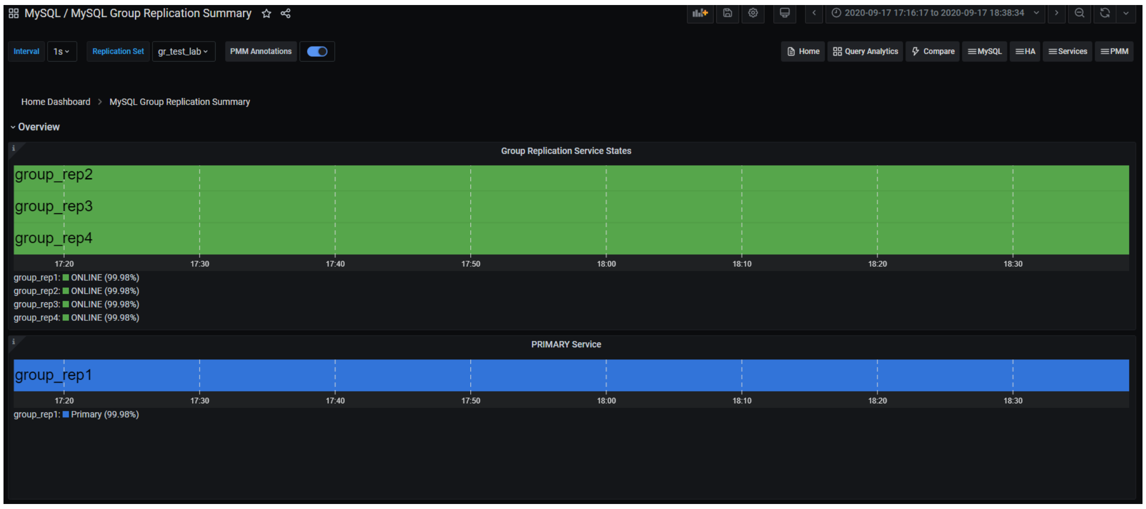
1. Use this quickstart to install Percona Monitoring and Management (PMM).
2. Specify the `replication_set` flag when registering the *Percona Server for MySQL* node or the MySQL node in PMM:

```
pmm-admin add mysql --username=pmm --password=pmm --query-source=perfschema --
replication-set=gr_test_lab  group_rep4 127.0.0.1:3306
```

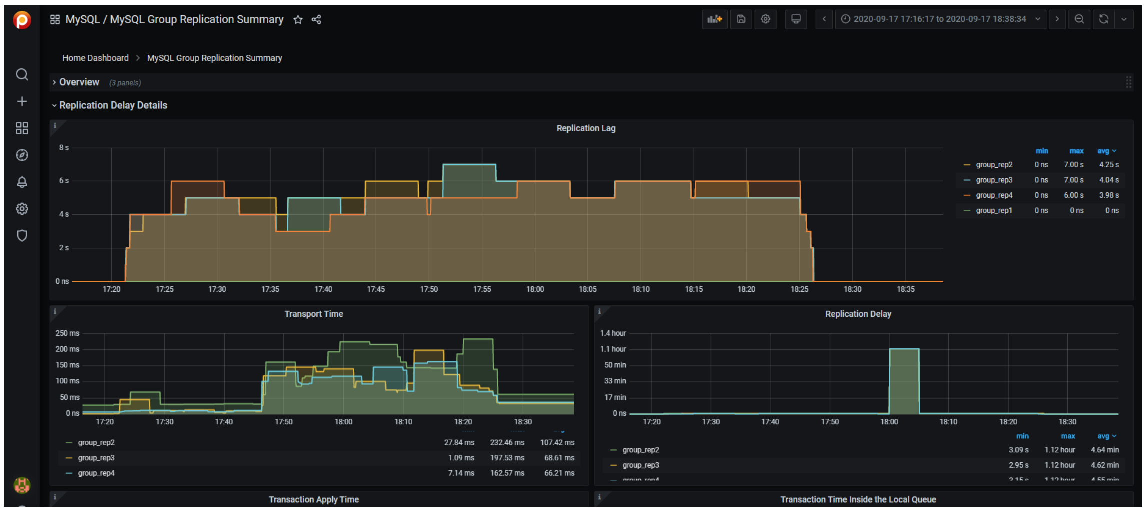
Then you can use the Group Replication Dashboard and monitor your cluster with a lot of details.

The dashboard sections are the following:

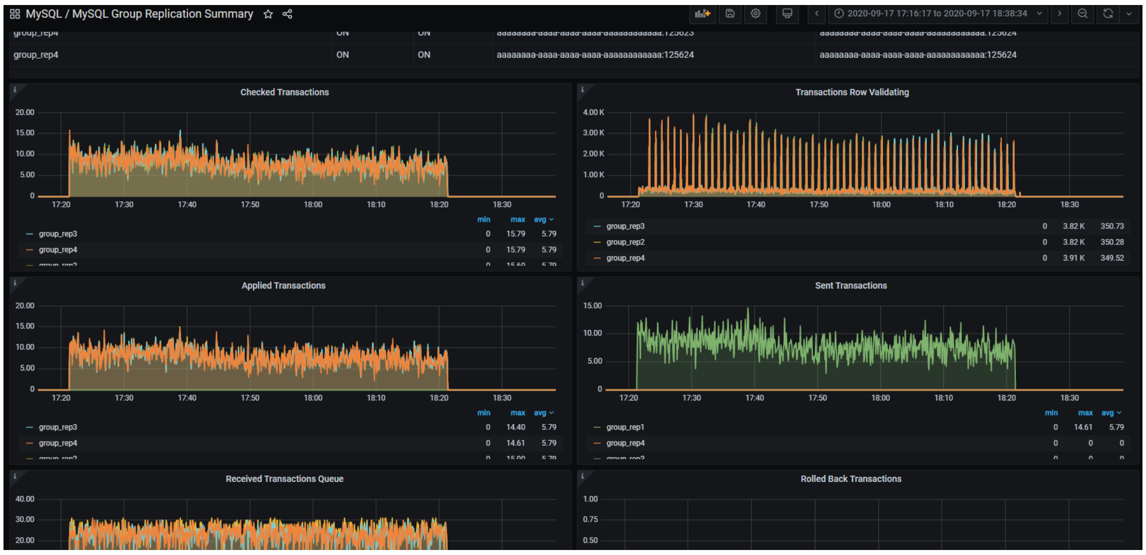
1. Overview:


MySQL / MySQL Group Replication Summary
Interval
1s
Replication Set
gr_test_lab
PMM Annotations
Home
Query Analytics
Compare
MySQL
HA
Services
PMM
Home Dashboard
MySQL Group Replication Summary
Overview
Group Replication Service States
group_rep2
group_rep3
group_rep4
17:20
17:30
17:40
17:50
18:00
18:10
18:20
18:30
group_rep1: ONLINE (99.98%)
group_rep2: ONLINE (99.98%)
group_rep3: ONLINE (99.98%)
group_rep4: ONLINE (99.98%)
PRIMARY Service
group_rep1
group_rep1: Primary (99.98%)

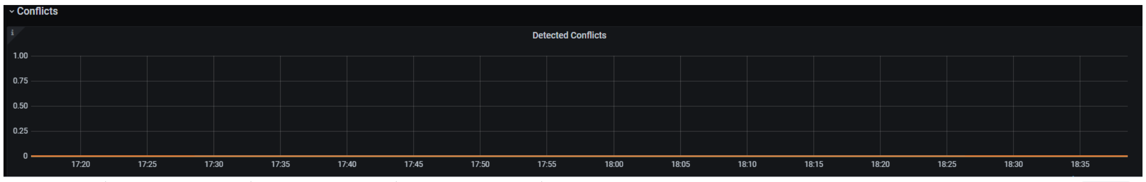
2. Replication delay details


MySQL / MySQL Group Replication Summary
2020-09-17 17:16:17 to 2020-09-17 18:38:34
Home Dashboard
MySQL Group Replication Summary
Overview
Replication Delay Details
Replication Lag
min
max
avg
group_rep2 0 ns 7.00 s 4.25 s
group_rep3 0 ns 7.00 s 4.04 s
group_rep4 0 ns 6.00 s 3.98 s
group_rep1 0 ns 0 ns 0 ns
Transport Time
group_rep2 27.84 ms 232.46 ms 107.42 ms
group_rep3 1.09 ms 197.53 ms 68.61 ms
group_rep4 7.14 ms 162.57 ms 66.21 ms
Replication Delay
group_rep2 3.09 s 1.12 hour 4.64 min
group_rep3 2.95 s 1.12 hour 4.62 min
Transaction Apply Time
Transaction Time Inside the Local Queue

3. Transactions


MySQL / MySQL Group Replication Summary
2020-09-17 17:16:17 to 2020-09-17 18:38:34
group_rep4 ON ON aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa:125624 aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa:125624
Checked Transactions
min max avg
group_rep3 0 15.79 5.79
group_rep4 0 15.79 5.79
Transactions Row Validating
group_rep3 0 3.82 K 350.73
group_rep2 0 3.82 K 350.28
group_rep4 0 3.91 K 349.52
Applied Transactions
min max avg
group_rep3 0 14.40 5.79
group_rep4 0 14.61 5.79
Sent Transactions
min max avg
group_rep1 0 14.61 5.79
group_rep4 0 0 0
Received Transactions Queue
Rolled Back Transactions

4. Conflicts



Using command line

From the command line, you need to manually query the tables in Performance schema:

```
+----------------------------------------------+
| replication_applier_configuration            |
| replication_applier_filters                  |
| replication_applier_global_filters           |
| replication_applier_status                   |
| replication_applier_status_by_coordinator    |
| replication_applier_status_by_worker         |
| replication_connection_configuration         |
| replication_connection_status                |
| replication_group_member_stats               |
| replication_group_members                    |
+----------------------------------------------+
```

For example, use this command to get the lag in number of transactions on a node:

```
select @last_exec:=SUBSTRING_INDEX(SUBSTRING_INDEX(
@@global.GTID_EXECUTED,':',-1),'-',-1) last_executed;select
@last_rec:=SUBSTRING_INDEX(SUBSTRING_INDEX(Received_transaction_set,':',-1),'-',-1)
last_received FROM performance_schema.replication_connection_status WHERE
Channel_name = 'group_replication_applier'; select (@last_rec - @last_exec) as
real_lag;
+---------------+
| last_executed |
+---------------+
| 125624        |
+---------------+
1 row in set, 1 warning (0.03 sec)

+---------------+
| last_received |
+---------------+
| 125624        |
+---------------+
1 row in set, 1 warning (0.00 sec)

+----------+
| real_lag |
+----------+
|        0 |
+----------+
1 row in set (0.00 sec)
```

You can use a more composite query to get information about each applier:

```
SELECT
  conn_status.channel_name as channel_name,
  conn_status.service_state as IO_thread,
  applier_status.service_state as SQL_thread,
  conn_status.LAST_QUEUED_TRANSACTION as last_queued_transaction,
  applier_status.LAST_APPLIED_TRANSACTION as last_applied_transaction,
  LAST_APPLIED_TRANSACTION_END_APPLY_TIMESTAMP -
                              LAST_APPLIED_TRANSACTION_ORIGINAL_COMMIT_TIMESTAMP 'rep
delay (sec)',
  LAST_QUEUED_TRANSACTION_START_QUEUE_TIMESTAMP -
                             LAST_QUEUED_TRANSACTION_ORIGINAL_COMMIT_TIMESTAMP
'transport time',
  LAST_QUEUED_TRANSACTION_END_QUEUE_TIMESTAMP -
                             LAST_QUEUED_TRANSACTION_START_QUEUE_TIMESTAMP 'time RL',
  LAST_APPLIED_TRANSACTION_END_APPLY_TIMESTAMP -
                             LAST_APPLIED_TRANSACTION_START_APPLY_TIMESTAMP 'apply
time',
  if(GTID_SUBTRACT(LAST_QUEUED_TRANSACTION, LAST_APPLIED_TRANSACTION) =
"","0" ,
abs(time_to_sec(if(time_to_sec(APPLYING_TRANSACTION_ORIGINAL_COMMIT_TIMESTAMP)=0,
0,timediff(APPLYING_TRANSACTION_ORIGINAL_COMMIT_TIMESTAMP,now()))))) `lag_in_sec`
FROM
  performance_schema.replication_connection_status AS conn_status
JOIN performance_schema.replication_applier_status_by_worker AS applier_status
  ON applier_status.channel_name = conn_status.channel_name
ORDER BY lag_in_sec, lag_in_sec desc\G
```

Expected output

```
*************************** 1. row ***************************
channel_name: group_replication_applier
IO_thread: ON
SQL_thread: ON
last_queued_transaction: aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa:125624
last_applied_transaction: aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa:125621
rep delay (sec): 3.153038
transport time: 0.061327
time RL: 0.001005
apply time: 0.388680
lag_in_sec: 0
```

Based on the material from Percona Database Performance Blog

This document is based on the blog post Percona Distribution for MySQL: High Availability with Group Replication Solution by *Marco Tusa*

5.4.6 Get expert help

If you need assistance, visit the community forum for comprehensive and free database knowledge, or contact our Percona Database Experts for professional support and services.

Community Forum **Get a Percona Expert**

Last update: 2023-12-11

6. Uninstall

6.1 Uninstalling Percona Distribution for MySQL

To uninstall Percona Distribution for MySQL, stop the `mysql` service and remove all the installed packages using the package manager of your operating system. Optionally, disable Percona repository.

Note

Should you need the data files later, back up your data before uninstalling Percona Distribution for MySQL.

Important

Run all commands as the root user or via `sudo`

On Debian / Ubuntu **On Red Hat Enterprise Linux / derivatives**

1. Stop the `mysql` service.

```
$ sudo systemctl stop mysql
```

2. Remove *Percona Server for MySQL*.

```
$ sudo apt remove percona-server*
```

3. Remove the components. Use the following commands to remove the required components.
   - Remove Percona XtraBackup

```
$ sudo apt remove percona-xtrabackup-81
```

   - Remove Percona Toolkit

```
$ sudo apt remove percona-toolkit
```

   - Remove Orchestrator

```
$ sudo apt remove percona-orchestrator*
```

   - Remove MySQL Shell

```
$ sudo apt remove percona-mysql-shell
```

   - Remove ProxySQL

```
$ sudo apt remove proxysql2
```

   - Remove MySQL Router

```
$ sudo apt remove percona-mysql-router
```

1. Stop the `mysql` service.

```
$ sudo systemctl stop mysql
```

2. Remove *Percona Server for MySQL*.

```
$ sudo yum remove percona-server*
```

3. Remove the components. Use the commands below to remove the required components.
   - Remove Percona XtraBackup

```
$ sudo yum remove percona-xtrabackup-81
```

   - Remove Percona Toolkit

```
$ sudo yum remove percona-toolkit
```

6.1.1 Get expert help

If you need assistance, visit the community forum for comprehensive and free database knowledge, or contact our Percona Database Experts for professional support and services.

Community Forum **Get a Percona Expert**

Last update: 2023-11-28

7. Copyright and licensing information

7.1 Documentation licensing

7.2 Get expert help

If you need assistance, visit the community forum for comprehensive and free database knowledge, or contact our Percona Database Experts for professional support and services.

Community Forum **Get a Percona Expert**

Last update: 2023-06-26

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 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.

8.1 Get expert help

If you need assistance, visit the community forum for comprehensive and free database knowledge, or contact our Percona Database Experts for professional support and services.

Community Forum **Get a Percona Expert**

Last update: 2023-06-26

9. Glossary

9.1 ACID

Set of properties that guarantee database transactions are processed reliably. Stands for `Atomicity`, `Consistency`, `Isolation`, `Durability`.

9.2 Asynchronous replication

Asynchronous replication is a technique where data is first written to the primary node. After the primary acknowledges the write, the data is written to secondary nodes.

9.3 Atomicity

Atomicity means that database operations are applied following an "all or nothing" rule. A transaction is either fully applied or not at all.

9.4 Consistency

In the context of backup and restore, consistency means that the data restored will be consistent in a given point in time. Partial or incomplete writes to disk of atomic operations (for example, to table and index data structures separately) won't be served to the client after the restore. The same applies to multi-document transactions, that started but didn't complete by the time the backup was finished.

9.5 Disaster recovery

Disaster recovery are means to regain access and functionality of a database infrastructure after unplanned events that caused its failure.

9.6 Downtime

Downtime is the period when a database infrastructure is unavailable due to expected (maintenance) or unexpected (outage, lost connectivity, hardware failure, etc.) reasons.

9.7 Durability

Once a transaction is committed, it will remain so.

9.8 Failover

Failover is switching automatically and seamlessly to a reliable backup system.

9.9 General availability (GA)

A finalized version of the product which is made available to the general public. It is the final stage in the software release cycle.

9.10 GTID

A global transaction identifier (GTID) is a unique identifier created and associated with each transaction committed on the server of the source. This identifier is unique across all servers in a given replication topology.

9.11 High availability

A high availability is the ability of a system to operate continuously without failure for a long time.

9.12 Isolation

The Isolation requirement means that no transaction can interfere with another.

9.13 Loosely-coupled cluster

A loosely-coupled cluster is the deployment where cluster nodes are independent in processing / applying transactions. Data state may not always be consistent in time on all nodes; however, a single node state does not affect the cluster. Loosely-coupled clusters use asynchronous replication and can be geographically distributed and/or serve as the disaster recovery site.

9.14 Multi-source replication

A multi-source replication topology requires at least one replica synchronized with at least two sources. The transactions can be received in parallel because the replica creates a separate replication channel for each source.

Multi-source replication allows a single server to back up or consolidate data from multiple servers. This type of replication also lets you merge table shards.

9.15 Nines of availability

Nines of availability refer to system availability as a percentage of total system time.

9.16 Semi-synchronous replication

A semi-synchronous replication is a technique where the primary node wait for at least one of the secondaries to acknowledge the transaction before processing further transactions.

9.17 Synchronous replication

A synchronous replication is a technique when data is written to the primary and secondary nodes simultaneously. Thus, both primary and secondaries are in sync and failover from the primary to one of the secondaries is possible any time.

9.18 Tech preview

A tech preview item can be a feature, a variable, or a value within a variable. The term designates that the item is not yet ready for production use and is not included in support by SLA. A tech preview item is included in a release so that users can provide feedback. The item is either updated and released as general availability(GA) or removed if not useful. The item's functionality can change from tech preview to GA.

9.19 Tightly-coupled cluster

A tightly-coupled cluster is the deployment in which transactions and information is synchronously distributed, consistent and available on all cluster nodes at any time.

9.20 Uptime

Uptime is the time when the system is continuously available.

9.21 Get expert help

If you need assistance, visit the community forum for comprehensive and free database knowledge, or contact our Percona Database Experts for professional support and services.

Community Forum   Get a Percona Expert

Last update: 2022-12-15