



# Databases run better with Percona

# Binary Log Server

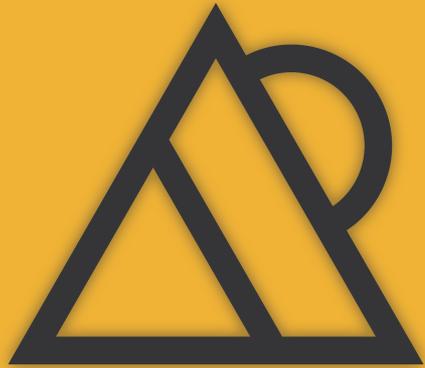
the missing MySQL infrastructure  
component



A huge fan of MySQL  
and c++

# Yura Sorokin

Principal Software Engineer at  
Percona



**PERCONA**®

# Introduction

motivational scenarios

# Continuous binlog backup for PITR

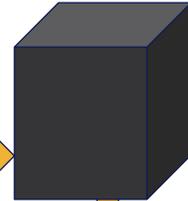
“mysqlbinlog” utility executed with  
 “--read-from-remote-server --raw”

- Can store binlog files only locally
- Does not try to reconnect
- Cannot resume from a partially downloaded file

MySQL Server



Local FS

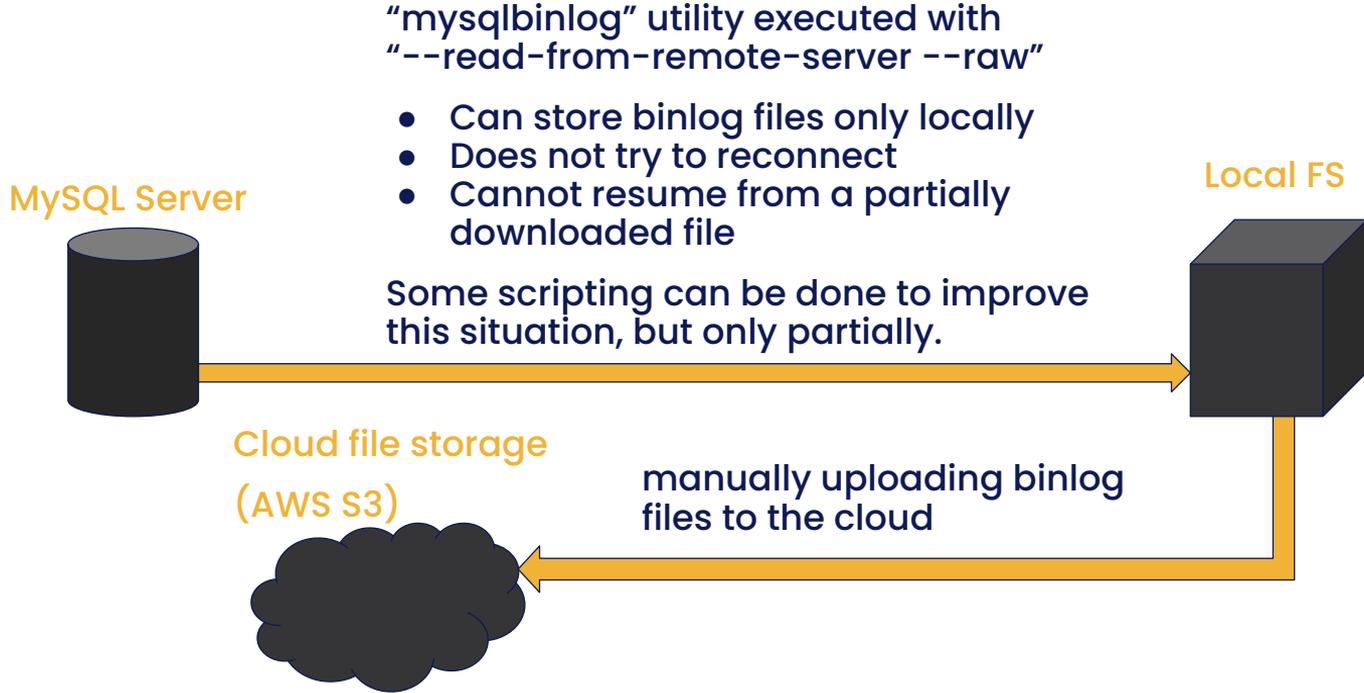


Some scripting can be done to improve this situation, but only partially.

Cloud file storage  
 (AWS S3)



manually uploading binlog files to the cloud

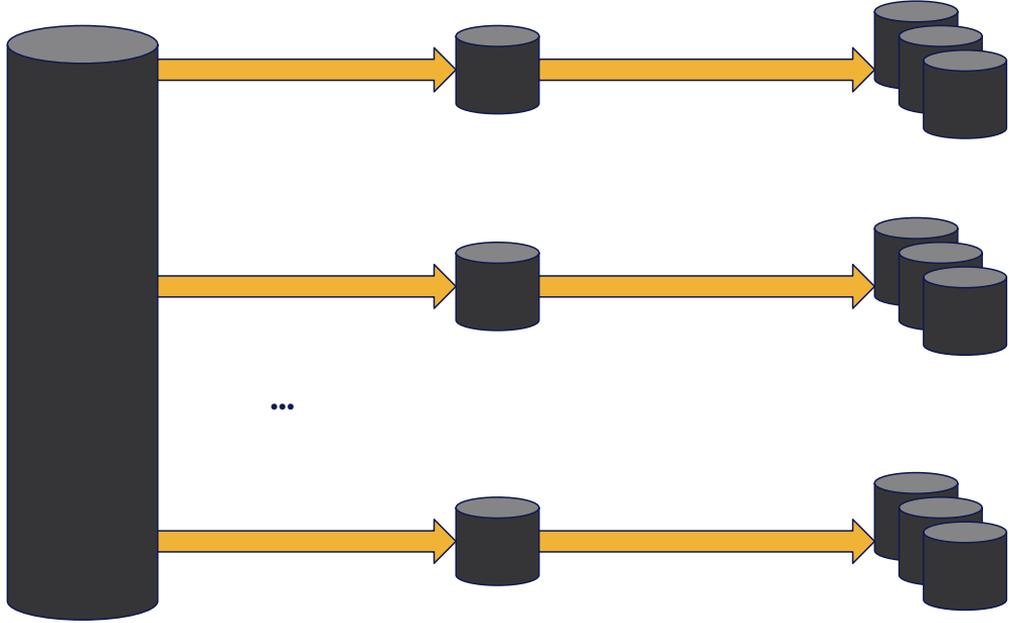


# Chained replication

MySQL Source

Level 0 MySQL replicas with Blackhole SE

Level 1 MySQL replicas



Group commit info is lost after binlog events pass through Level 0 MySQL replicas in which for all tables we created their dummy equivalents (using Blackhole Storage Engine).

This drastically affects the behavior of the parallel applier on Level 1 replicas.

# Chained replication (explained)

Some people call Level 0 replicas (those with the Blackhole Storage Engine) “Binlog Servers”.

Despite the simplicity of the Blackhole SE, Level 0 servers add significant overhead on the task they are meant to do – simply receiving binlog event from a source and passing them to multiple replicas.

# Chained replication (reported issues)

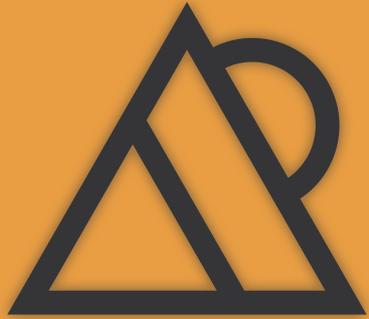
## [Percona Server Bug PS-4560](#)

“Slave writes less when connected intermediary master with blackhole engine”

## [MySQL Bug #91477](#)

“Slave writes less when connected intermediary master with blackhole engine”

Created: 28 Jun 2018



**PERCONA®**

# An opportunity

a new project arises

# Percona Binary Log Server (percona-binlog-server)

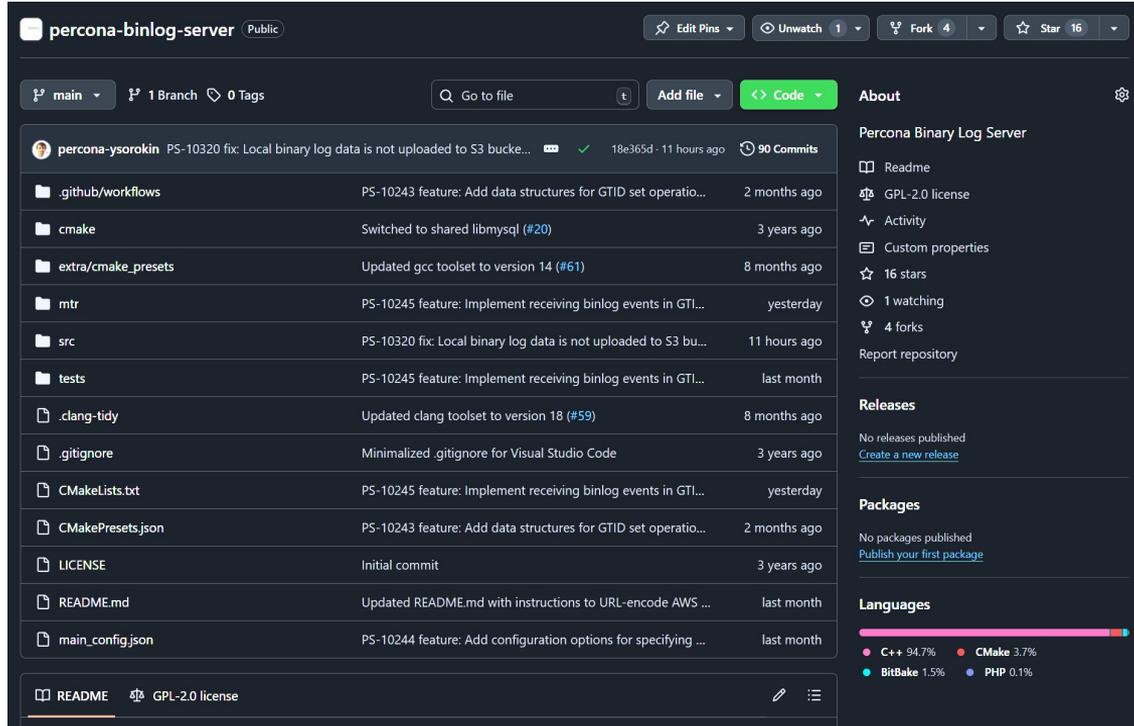
An new Open-Source project (under GPL-2.0 License).

Available on GitHub

<https://github.com/Percona-Lab/percona-binlog-server>

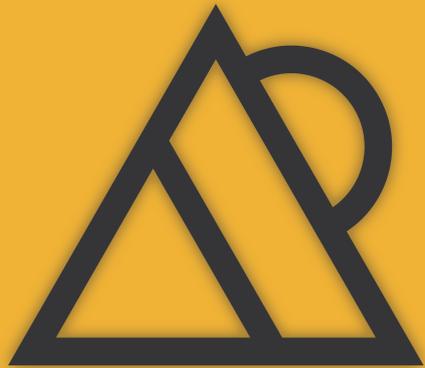
- Written in c++20
- Enabled GitHub Actions CI
- Enforced clang-format rules
- Enforced clang-tidy checking
- Depends on Boost c++ libraries and AWS SDK for C++
- Uses boost::test as a unit test framework
- Uses MTR for integration tests

# Percona Binary Log Server (proof)



The screenshot shows the GitHub repository page for 'percona-binlog-server'. The repository is public and has 16 stars, 4 forks, and 1 unwatch. It is currently on the 'main' branch with 1 branch and 0 tags. The repository contains 18 files and folders, including .github/workflows, cmake, extra/cmake\_presets, mtr, src, tests, .clang-tidy, .gitignore, CMakeLists.txt, CMakePresets.json, LICENSE, README.md, and main\_config.json. The most recent commit is by percona-ysorokin, titled 'PS-10320 fix: Local binary log data is not uploaded to S3 bucke...', made 11 hours ago. The repository also has a README, a GPL-2.0 license, and a language usage chart showing C++ at 94.7%, CMake at 3.7%, BitBake at 1.5%, and PHP at 0.1%.

File/Folder	Description	Time
.github/workflows	PS-10243 feature: Add data structures for GTID set operatio...	2 months ago
cmake	Switched to shared libmysql (#20)	3 years ago
extra/cmake_presets	Updated gcc toolset to version 14 (#61)	8 months ago
mtr	PS-10245 feature: Implement receiving binlog events in GTI...	yesterday
src	PS-10320 fix: Local binary log data is not uploaded to S3 bu...	11 hours ago
tests	PS-10245 feature: Implement receiving binlog events in GTI...	last month
.clang-tidy	Updated clang toolset to version 18 (#59)	8 months ago
.gitignore	Minimalized .gitignore for Visual Studio Code	3 years ago
CMakeLists.txt	PS-10245 feature: Implement receiving binlog events in GTI...	yesterday
CMakePresets.json	PS-10243 feature: Add data structures for GTID set operatio...	2 months ago
LICENSE	Initial commit	3 years ago
README.md	Updated README.md with instructions to URL-encode AWS ...	last month
main_config.json	PS-10244 feature: Add configuration options for specifying ...	last month



**PERCONA**®

## Current functionality

“mysqlbinlog” on steroids

## Binlog Server intro (a quote from the project's README.md)

`binlog_server` is a command line utility that can be considered as an enhanced version of `mysqlbinlog` in `--read-from-remote-server` mode which serves as a replication client and can stream binary log events from a remote Oracle MySQL Server / Percona Server for MySQL...

... It is capable of automatically reconnecting to the remote server and resume operation from the point when it was previously stopped / terminated.

## Key features

- In comparison to a separate dedicated instance of MySQL Server, Binlog Server operates only at binary log events level and has no overhead of full-blown RDBMS.
- In comparison to “mysqlbinlog” utility, Binlog Server supports continuous pulling mode and resuming from previously saved state.
- Binlog Server can work with cloud file storages directly:
  - AWS S3 is supported in the initial version.
  - Support for S3-compatible services added a bit later.
  - Pluggable infrastructure allows to easily add support for other Cloud providers (Azure, Google Cloud).
- Binlog Server operates in two modes:
  - “fetch” mode - start receiving binlog events from the remote server and stop when reached the very last event or an error occurred.
  - “pull” mode - continuous mode, that handles disconnects / reconnects and terminates only upon receiving a signal.

## More key features (PITR helpers)

Binlog Server can query its storage directly and answer the following questions:

- Give me the list of binlog files (along with their direct URLs) that include at least one binlog event created earlier than the provided **<TIMESTAMP>**
- Give me the minimal list of binlog files that covers all GTIDs specified in the provided **<GTID\_SET>**



# Usage

how to set up and run

# Binlog Server connection features

- Connections can be established to 8.0 and 8.4 / 9.x (currently tested with 9.5) MySQL Servers.
- We support full variety of mysqlclient SSL / TLS options.
- We support connections via “dns\_srv\_name” (used a lot in Kubernetes environments).
- Configurable connection / read timeouts.

```
"connection": {  
  "host": "127.0.0.1",  
  "port": 3306,  
  "user": "rpl_user",  
  "password": "rpl_password",  
  "connect_timeout": 20,  
  "read_timeout": 60,  
  "write_timeout": 60,  
  "ssl": {  
    "mode": "required"  
  }  
}
```

# Binlog Server replications features

- We support position-based replication
  - Producing physical copies of remote binlog files
- We support GTID-based replication
  - The received binlog files may not be identical to the remote ones but GTID sets allow to resume operations from a different server in the topology
- We support binlog event checksumming
- "server\_id" can be configured

```
"replication": {  
  "server_id": 42,  
  "idle_time": 10,  
  "verify_checksum": true,  
  "mode": "position"  
}
```

# Binlog Server storage features

- We support local filesystem as a binlog file storage
- We support AWS S3 as a binlog file storage
- We support S3-compatible services (like MinIO) with custom endpoint URLs.
- We support buffering and event data flushing upon checkpointing events:
  - size-based checkpoints, when data accumulated in the buffer exceeds the configured value
  - time-based checkpointing, when time elapsed from the last flush exceeded the configured value

```
"storage": {  
  "backend": "s3",  
  "uri": "https://key\_id:secret@192.168.0.100:9000/binsrv-bucket/vault",  
  "fs_buffer_directory": "/tmp/binsrv",  
  "checkpoint_size": "128M",  
  "checkpoint_interval": "30s"  
}
```

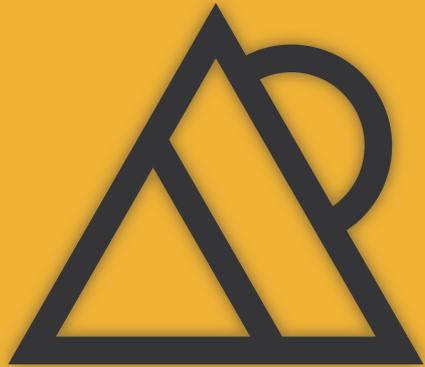
# Binlog Server how to use

1. Create a configuration file “config.json” with the 3 sections mentioned on previous slides + one section for logging. You basically specify 3 things:
  - a. From where to get data.
  - b. Where to store it.
  - c. How to interpret it (replication mode)
2. Create a directory on a local filesystem / a virtual directory in a bucket on S3 for storing binlog files.
3. Run “binlog\_server pull config.json”
4. You will soon see something like the following in your storage directory:

```
-rw-r--r-- 1 user group 682 Jan 20 00:42 binlog.000001
-rw-r--r-- 1 user group 131 Jan 20 00:42 binlog.000001.json
-rw-r--r-- 1 user group 519 Jan 20 00:42 binlog.000002
-rw-r--r-- 1 user group 131 Jan 20 00:42 binlog.000002.json
-rw-r--r-- 1 user group 4 Jan 20 00:42 binlog.000003
-rw-r--r-- 1 user group 129 Jan 20 00:42 binlog.000003.json
-rw-r--r-- 1 user group 48 Jan 20 00:42 binlog.index
-rw-r--r-- 1 user group 27 Jan 20 00:42 metadata.json
```

# Challenges

replication internals



**PERCONA**®

## Solved problems (part 1)

1. Support for 8.0 and 8.4/9.x MySQL servers. In 8.4/9.x servers there is new GTID\_TAGGED\_LOG event.
2. Deciding whether certain events from the initial event sequence should be written to storage or not (Artificial ROTATE, FORMAT\_DESCRIPTION and optional PREVIOUS\_GTIDS\_LOG received upon reconnection).
3. Identifying transaction boundaries (luckily, starting from 8.0 all GTID-related events include “transaction\_length” field in their body).
4. Making sure that we permanently write data to the storage only at transaction boundaries.

## Solved problems (part 2)

1. Implementing reconnection logic trying to minimize open/close operations in the storage.
2. Tracking GTID sets associated with binlog files.
3. Implementing event offsets rewriting in GTID mode (in case when we reconnected to another server with different physical binlog layout).
4. Implementing reorganizing binlog file splitting in the storage in GTID mode.
5. Implementing “append” operation on S3.



# Roadmap

the project is being actively developed

## Multi-tenant and control interface

- We want Binlog Server to be able to work with multiple MySQL Servers and multiple storages simultaneously.
- We want to have a control port that would accept control commands (**possible** syntax):
  - CREATE STORAGE 'vault1' ...
  - CREATE SERVER 'mysql1' AS 192.168.0.100:3306 ...
  - CREATE REPLICATION TASK 'task1' FROM 'mysql1' TO 'vault1' IN GTID MODE
  - START TASK 'task1'
  - INFO TASK 'task1'
  - INFO STORAGE 'vault1'
  - INFO SERVER 'mysql1'
  - STOP TASK 'task1'
  - DROP REPLICATION TASK 'task1'
  - DROP SERVER 'mysql1'
  - DROP STORAGE 'vault1'
  - ...

## REST for control interface

- PUT /api/v1/storages HTTP/1.1  
{ ... }
- GET /api/v1/tasks/task1 HTTP/1.1
- DELETE /api/v1/connections/mysql1 HTTP/1.1

## Make Binlog Server a “Server”

Add functionality that would allow to use Binlog Server as a MySQL replication source.

CHANGE REPLICATION SOURCE TO

```
SOURCE_HOST='<binlog_server_address>',  
SOURCE_USER='replication_user_name',  
SOURCE_PASSWORD='replication_password',
```

...

A large, solid yellow circle is positioned on the left side of the slide, containing the text 'Thank You!' in a bold, black, sans-serif font.

**Thank You!**

Databases run better with Percona