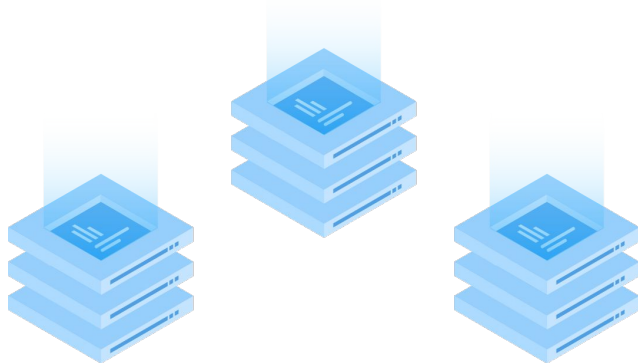


# From MySQL to TiDB and back again

Percona Live Austin 2019

Morgan Tocker (@morgo)



TiDB Community Slack Channel  
<https://pingcap.com/tidbslack/>



# Agenda

- **Why???**
- Demo time
- Discuss validation steps

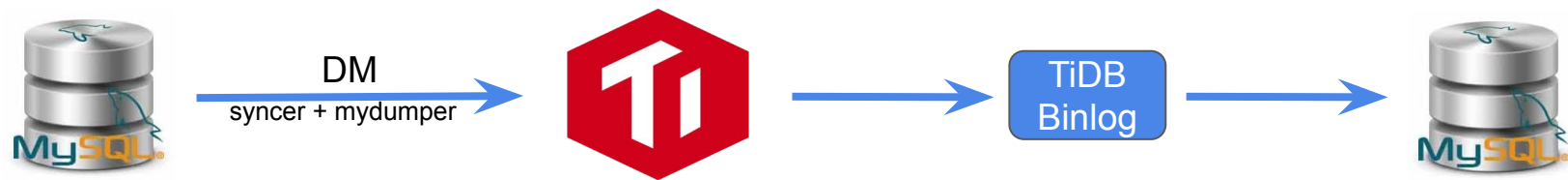


# Why?

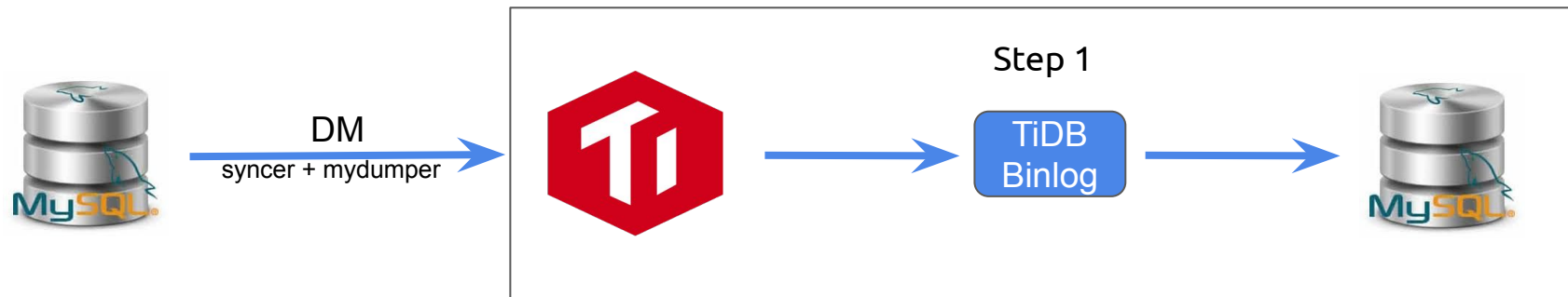
- Any good plan has a way to rollback if issues are encountered
- Let's assume that we want to use TiDB
- Making it easier to rollback reduces project risk
- Lower project risk = *lower barrier to entry*



# Workflow (visually)



# Workflow (visually)



# We will use the tutorial

No need to reinvent the wheel.

Deploys TiDB + TiDB-Binlog on a single desktop machine.

We will change that to DBdeployer (since there is both an upstream and downstream).

TiDB-Binlog Tutorial | Am: x +

← → C <https://pingcap.com/docs/dev/how-to/get-started/tidb-binlog/>

PingCAP Cloud TiDB Academy Docs Success Stories Blog Free Download Search...

Documentation

- Introduction
- Concepts >
- How-to >
  - Get Started >
    - Start a Local Cluster >
    - Explore SQL with TiDB
    - Import Example Database
    - Read Historical Data
    - TiDB-Binlog Tutorial**
    - TiDB Data Migration Tutorial
  - Deploy >
  - Configure >
  - Secure >
  - Monitor >
  - Migrate >
  - Maintain >

## TiDB-Binlog Tutorial

[Edit this page](#)  
[Request docs changes](#)

What's on this page

- TiDB-Binlog Overview**
- Architecture
- Installation
- Configuration
- Bootstrapping
- Connecting
- binlogctl
- Cleanup
- Conclusion

This tutorial starts with a simple TiDB-Binlog deployment with a single node of each component (Placement Driver, TiKV Server, TiDB Server, Pump, and Drainer), set up to push data into a MariaDB Server instance.

This tutorial is targeted toward users who have some familiarity with the [TiDB Architecture](#), who may have already set up a TiDB cluster (not mandatory), and who wants to gain hands-on experience with TiDB-Binlog. This tutorial is a good way to “kick the tires” of TiDB-Binlog and to familiarize yourself with the concepts of its architecture.

**Warning:**  
The instructions to deploy TiDB in this tutorial should **not** be used to deploy TiDB in a production or development setting.

This tutorial assumes you're using a modern Linux distribution on x86-64. A minimal CentOS 7 installation running in VMware is used in this tutorial for the examples. It's recommended that you start from a clean install, so that you aren't impacted by quirks of your existing environment. If you don't want to use local virtualization, you can easily start a CentOS 7 VM using your cloud service.

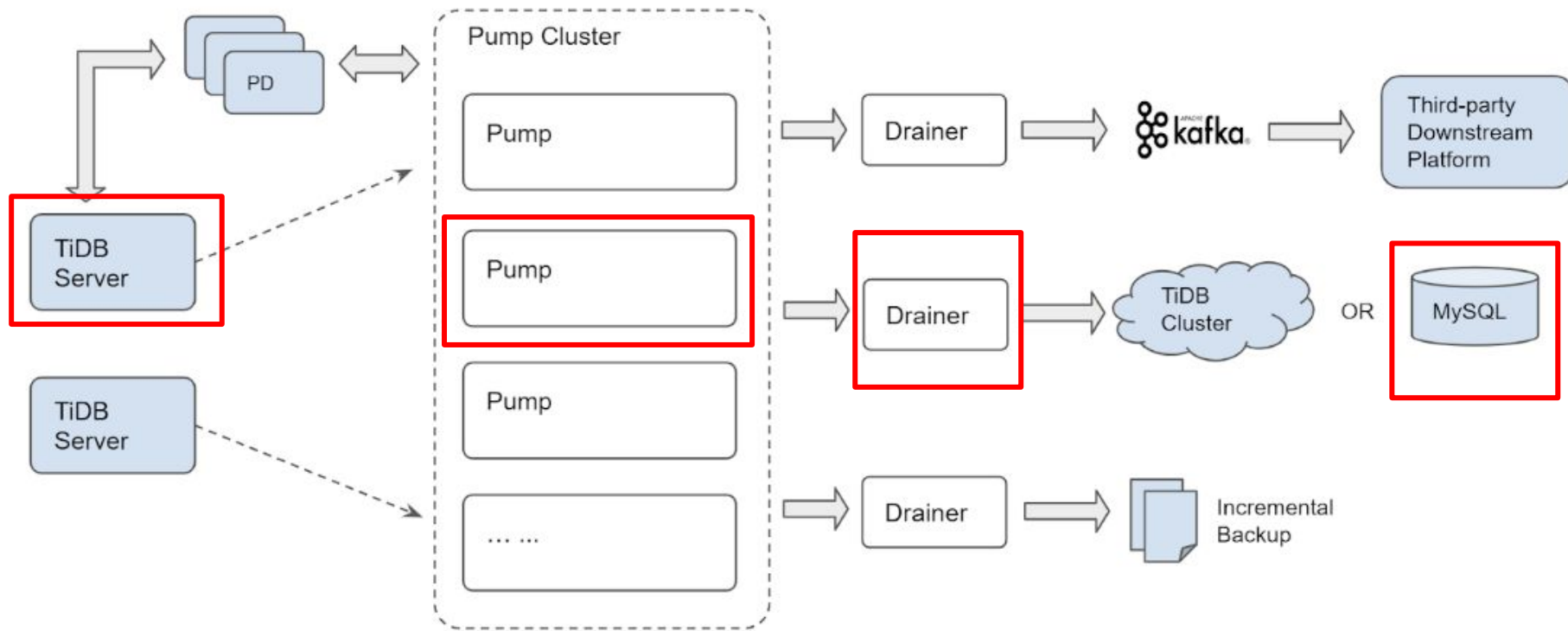
## TiDB-Binlog Overview

TiDB-Binlog is a solution to collect binary log data from TiDB and provide real-time data backup and replication. It pushes incremental data updates from a TiDB Server cluster into downstream platforms.

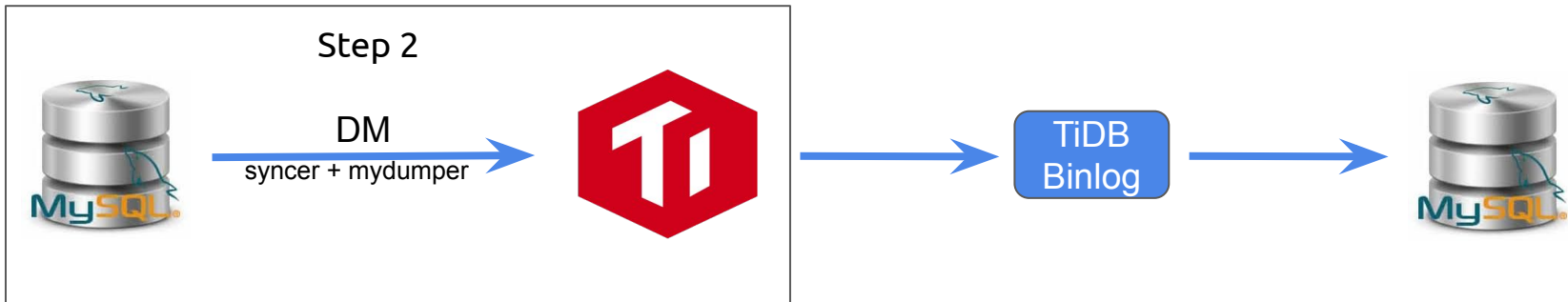
<https://drive.google.com/open?id=1YiBjzJdiqOVcP93qBhwr2NYaWfhD9XsA>



# What is deployed?



# Next Step

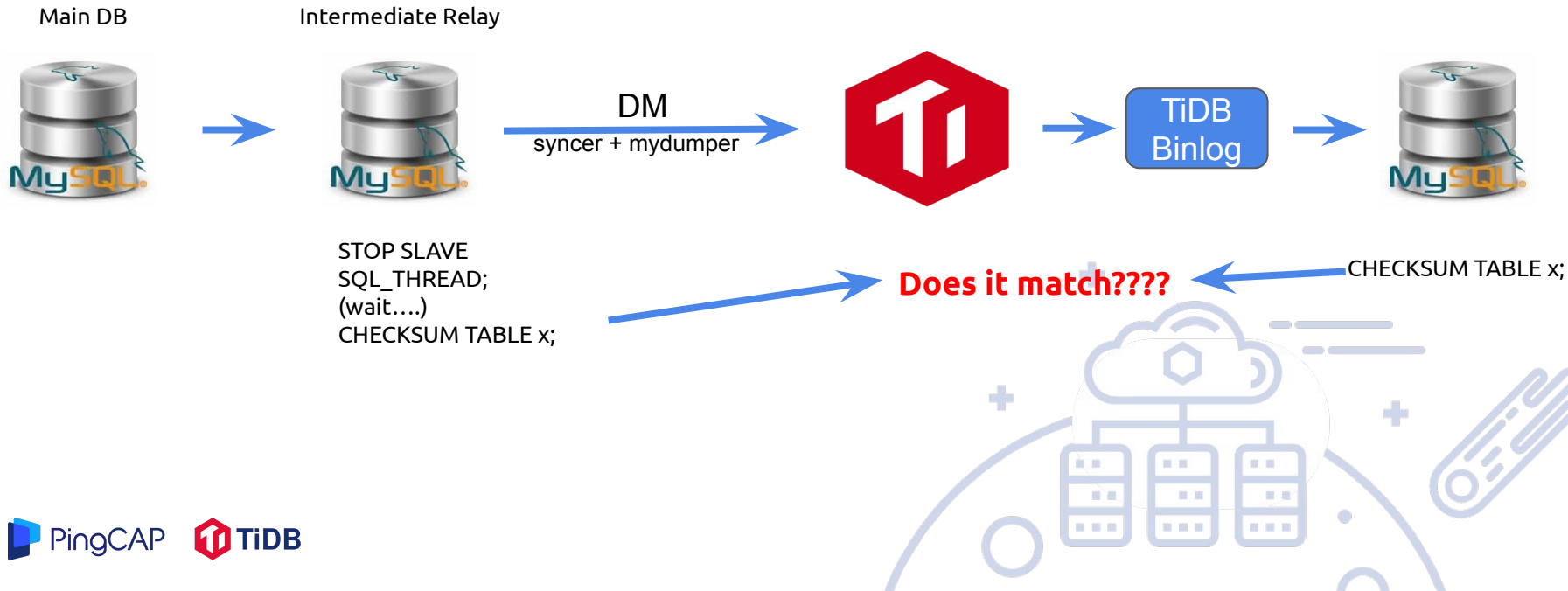


<https://drive.google.com/open?id=192DUbr5EDls1snmB20fP40udoCKx2y1Q>



# End to End Test

- Load data in front MySQL, CHECKSUM TABLE at final MySQL
- Can be done in production too, but requires intermediate slave:



<https://drive.google.com/open?id=1JM5lbWlzd9BFX2LHKgM0sJ7soEEKUY4>



# Testing Suggestions

- Replication traffic of DML is easy! (RBR events, not statements)
  - Need to validate that DDL replicates too!
- Need to validate SELECT statements:
  - For correctness
  - For performance
- Need to validate stability:
  - Stable memory, disk usage, errors...



# Testing Suggestions

- Replication traffic of DML is easy! (RBR events, not statements)
  - Need to validate that DDL replicates too!
- Need to validate SELECT statements:
  - For correctness
  - For performance
- Need to validate stability:
  - Stable memory, disk usage, errors...

**just leave  
replication  
running!**

**pt-upgrade**

**ProxySQL mirroring** (+ look at health metrics in Grafana)



## pt-upgrade example

```
morgo@ryzen:~$ pt-upgrade /home/morgo/sandboxes/msb_5_7_24/data/ryzen-slow.log h=127.0.0.1,u=msandbox,p=msandbox,P=5724  
h=127.0.0.1,u=root,P=4000,p=
```

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

```
File: /home/morgo/sandboxes/msb_5_7_24/data/ryzen-slow.log  
Size: 587458
```

```
#-----  
# Hosts  
#-----
```

host1:

```
DSN:      h=127.0.0.1,P=5724  
hostname: ryzen  
MySQL:    MySQL Community Server (GPL) 5.7.24
```

host2:

```
DSN:      h=127.0.0.1,P=4000  
hostname: ryzen  
MySQL:    MySQL Community Server (Apache License 2.0) 5.7.25
```

Let's see the differences  
after recording a small  
number of queries  
(including recording  
pt-upgrade itself)!



## pt-upgrade example (cont.)

```
#####  
# Query class D3F15165B8904806  
#####
```

Reporting class because there are 3 row diffs.

```
Total queries      3  
Unique queries     3  
Discarded queries  0
```

```
select @@sql_mode
```

```
##  
## Row diffs: 3  
##
```

```
-- 1.
```

```
@ row 1  
< ONLY_FULL_GROUP_BY,NO_AUTO_VALUE_ON_ZERO,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION  
> NO_AUTO_VALUE_ON_ZERO,ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION
```

```
SELECT @@SQL_MODE
```

```
-- 2.
```

```
@ row 1  
< ONLY_FULL_GROUP_BY,NO_AUTO_VALUE_ON_ZERO,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION  
> NO_AUTO_VALUE_ON_ZERO,ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION
```

```
SELECT @@SQL_MODE
```

```
-- 3.
```

```
@ row 1  
< ONLY_FULL_GROUP_BY,NO_AUTO_VALUE_ON_ZERO,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION  
> NO_AUTO_VALUE_ON_ZERO,ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION
```

```
SELECT @@SQL_MODE
```

Order of SQL modes differs! It could be considered a TiDB compatibility Bug...



## pt-upgrade example (cont.)

```
#####  
# Query class C727AFF0F56E1A27  
#####  
Reporting class because there are 3 query diffs, and there are 3 row diffs.
```

```
Total queries      3  
Unique queries     3  
Discarded queries  0
```

```
select @@server_id /*!? , @@hostname*/
```

```
##  
## Query time diffs: 3  
##
```

```
-- 1.
```

```
0.000068 vs. 0.001380 seconds ( 20.3x increase)
```

```
SELECT @@server_id /*!50038 , @@hostname*/
```

```
-- 2.
```

```
0.000107 vs. 0.001261 seconds ( 11.8x increase)
```

```
SELECT @@server_id /*!50038 , @@hostname*/
```

```
-- 3.
```

```
0.000104 vs. 0.001243 seconds ( 12.0x increase)
```

```
SELECT @@server_id /*!50038 , @@hostname*/
```

```
##  
## Row diffs: 3  
##
```

```
-- 1.
```

```
@ row 1  
< 2,ryzen  
> 0,ryzen
```

```
SELECT @@server_id /*!50038 , @@hostname*/
```

```
-- 2.
```

```
@ row 1  
< 2,ryzen  
> 0,ryzen
```

```
SELECT @@server_id /*!50038 , @@hostname*/
```

```
-- 3.
```

```
@ row 1  
< 2,ryzen  
> 0,ryzen
```

```
SELECT @@server_id /*!50038 , @@hostname*/
```

TiDB returns server id zero  
(expected to differ), and  
execution time is statistically  
significantly higher (12x-20x)



## pt-upgrade example (cont.)

```
#####  
# Query class 511E2D1FBE91A9D7  
#####
```

Reporting class because there are 3 warning diffs.

```
Total queries      3  
Unique queries     3  
Discarded queries  0
```

```
select @@query_cache_type
```

```
##  
## Warning diffs: 3  
##
```

```
-- 1.  
  
Code: 1287  
Level: Warning  
Message: '@@query_cache_type' is deprecated and will be removed in a future release.
```

vs.

No warning 1287

```
SELECT @@query_cache_type
```

```
-- 2.  
  
Code: 1287  
Level: Warning  
Message: '@@query_cache_type' is deprecated and will be removed in a future release.
```

vs.

No warning 1287

```
SELECT @@query_cache_type
```

TiDB does not yet report query cache is deprecated. This warning was introduced mid-5.7.

```
-- 3.  
  
Code: 1287  
Level: Warning  
Message: '@@query_cache_type' is deprecated and will be removed in a future release.
```

vs.

No warning 1287

```
SELECT @@query_cache_type
```



## pt-upgrade example (cont.)

```
#####  
# Query class 444F46FAC84BE30F  
#####
```

Reporting class because there are 3 query diffs.

```
Total queries      3  
Unique queries     3  
Discarded queries  0
```

```
select * from percona_schema.pt_upgrade limit ?
```

```
##  
## Query time diffs: 3  
##
```

```
-- 1.  
  
0.000090 vs. 0.000906 seconds ( 10.1x increase)  
  
SELECT * FROM percona_schema.pt_upgrade LIMIT 1 /* pt-upgrade clear warnings */  
  
-- 2.  
  
0.000055 vs. 0.000663 seconds ( 12.1x increase)  
  
SELECT * FROM percona_schema.pt_upgrade LIMIT 1 /* pt-upgrade clear warnings */  
  
-- 3.  
  
0.000089 vs. 0.000954 seconds ( 10.7x increase)  
  
SELECT * FROM percona_schema.pt_upgrade LIMIT 1 /* pt-upgrade clear warnings */
```

Results are the same (table was empty), but execution time is statistically significantly higher.



# ProxySQL

<https://github.com/sysown/proxysql/wiki/Mirroring>

The screenshot shows the GitHub Wiki page for ProxySQL's Mirroring feature. The page title is "Mirroring" and it was edited by René Cannao on 27 Jan. The page content includes a "Note that:" section with two bullet points: "specifications can change at any time" and "It doesn't support prepared statements". Below this is a section titled "Extensions to mysql\_query\_rules" which states that the table was modified to add two more columns. A table definition for mysql\_query\_rules is shown, including columns like rule\_id, username, schemaname, flagIN, match\_digest, match\_pattern, negate\_match\_pattern, flagOUT, replace\_pattern, and destination\_hostgroup. A sidebar on the right lists navigation links such as "Getting started", "Configuring ProxySQL", "Global Variables", "MySQL Server configuration", "Users configuration", "Password management", "Scheduler", "Query Logging", "Admin Schemas", "main (runtime)", "Servers", "Replication Hostgroups", "Galera Hostgroups", "Group Replication Hostgroups", "Query Rules", "Query Rules Fast Routing", "Global Variables", "ProxySQL Servers", "Scheduler", "Collations", and "Question/Feedback".

Mirroring

René Cannao edited this page on 27 Jan · 6 revisions

## Mirroring

Note that:

- specifications can change at any time
- It doesn't support prepared statements

### Extensions to mysql\_query\_rules

Table `mysql_query_rules` was modified to add 2 more columns:

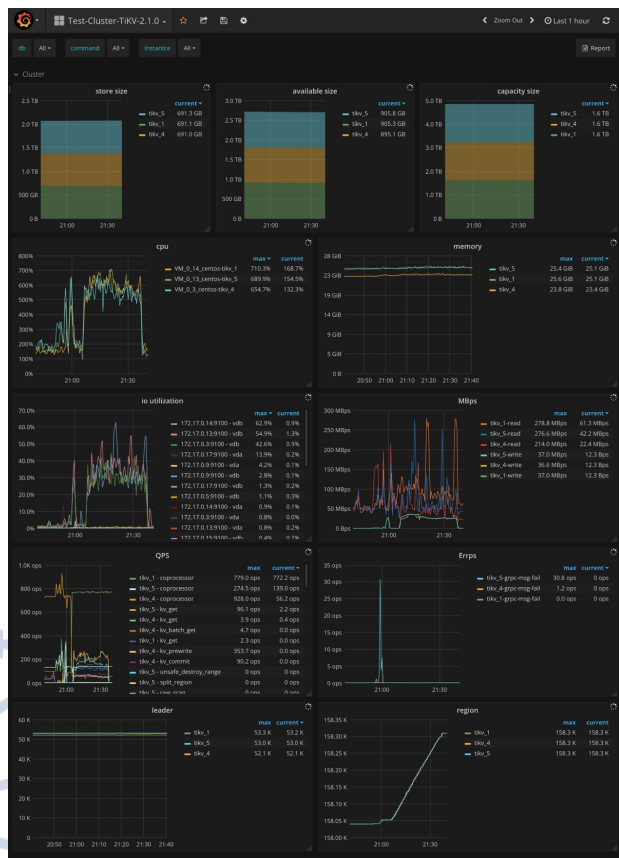
- `mirror_flagOUT`
- `mirror_hostgroup`

Therefore, the new table definition of `mysql_query_rules` becomes:

```
mysql> show create table mysql_query_rules\G
***** 1. row *****
      table: mysql_query_rules
Create Table: CREATE TABLE mysql_query_rules (
  rule_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
  active INT CHECK (active IN (0,1)) NOT NULL DEFAULT 0,
  username VARCHAR,
  schemaname VARCHAR,
  flagIN INT NOT NULL DEFAULT 0,
  match_digest VARCHAR,
  match_pattern VARCHAR,
  negate_match_pattern INT CHECK (negate_match_pattern IN (0,1)) NOT NULL DEFAULT 0,
  flagOUT INT,
  replace_pattern VARCHAR,
  destination_hostgroup INT DEFAULT NULL)
*****
```

<https://pingcap.com/docs/dev/reference/key-monitoring-metrics/tikv/>

+ Query Time in ProxySQL



**Thank you!**  
Have questions?  
[pingcap.com/tidbslack](https://pingcap.com/tidbslack)

