
PlanetScale

Introduction to Vitess
sharding framework for MySQL

Matthias Crauwels
Enterprise Customer Engineer, PlanetScale
@mcrauwel

// Introduction

Who am I?

- Living in Ghent, Belgium 🇧🇪
- Bachelor Computer Science
- ~25 years Linux user / admin
- ~15 years PHP developer
- ~10 years MySQL DBA
- 1st year at PlanetScale
- Currently Enterprise Customer Engineer
- Father of Leander

About PlanetScale

PlanetScale is a MySQL-compatible serverless database that brings you scale, performance, and reliability — without sacrificing developer experience.

With PlanetScale, you get the power of horizontal sharding, non-blocking schema changes, and many more powerful database features without the pain of implementing them.

About PlanetScale

PlanetScale is powered by Vitess, the open-source database technology that was invented at YouTube in 2010 to solve the scaling issues they faced with their massive MySQL database.

Vitess went on to become open source as a CNCF project and continues to scale massive companies like Slack, GitHub, and more.

Vitess serves millions of QPS in production

// Introduction

Agenda

- Architecture
- MySQL Compatibility
- VReplication
- Online Schema Changes
- Automatic failovers

Architecture

// Architecture

What is Vitess?

- Cloud Native Database
- Massively Scalable
- Highly Available
- MySQL Compatible

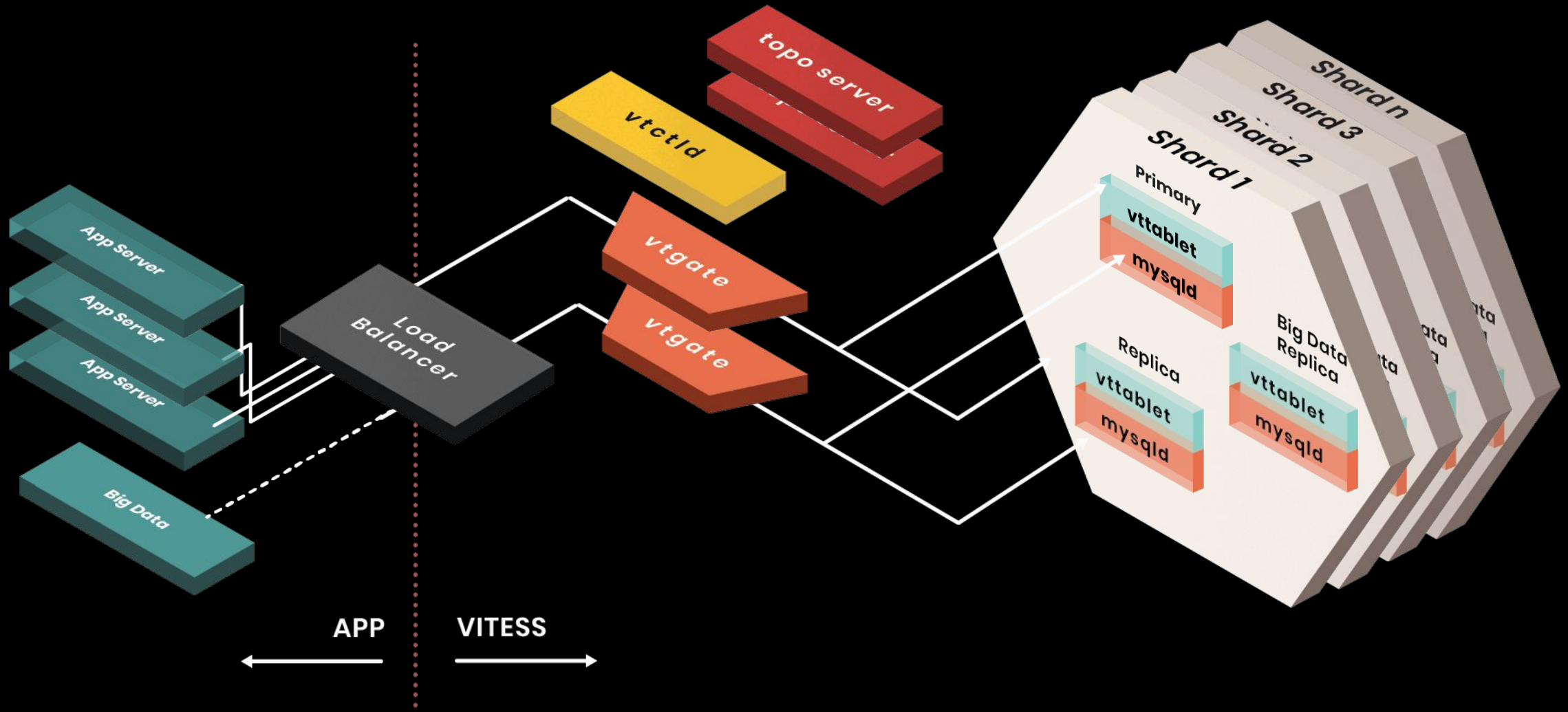
// Architecture

Concepts

- Keyspace
- Shard
- Topology server

// Architecture

Vitess Architecture



MySQL Compatibility

// MySQL Compatibility

Vitess provides an illusion

- A single database (server)
- MySQL 5.7 or MySQL 8.0
- A single, dedicated connection

// MySQL Compatibility

Vitess needs to deal with

- Database frameworks
- ORMs
- Legacy code
- Third Party Applications

VReplication

// VReplication

Use cases

- Migration
  - Import data into Vitess
- (Re)sharding
  - Move data around
- Materialisation
  - Improve query performance
  - Materialise unsharded data on each shard

Import workflow

- Create an “external” keyspace with a vttablet pointing to an existing MySQL server.
- Create a Vitess keyspace
- Use the MoveTables command to create a VReplication Workflow to move data into Vitess
- Confirm that all data was migrated (VDiff)
- Switch read traffic to go to Vitess
- Switch write traffic to go to Vitess
- Full support for rollback in case of issues

Online Schema Changes

// Online Schema Changes

ALTER TABLE issues

- Blocking
- Resource greedy
- Not auditable
- Not interruptible

ALTER TABLE Workarounds

- gh-ost, developed by GitHub
- pt-online-schema-change, part of Percona Toolkit
- External tools
- Requires a lot of operational knowledge of the environment
- Takes control out of the developer’s hands

// Online Schema Changes

Based on VReplication

```
mysql> SET @@ddl_strategy = 'online';
Query OK, 0 rows affected (0.00 sec)

mysql> ALTER TABLE table1 CHANGE id id bigint unsigned;
+--------------------------------------+
| uuid                                 |
+--------------------------------------+
| 4d8246f2_9801_11ed_a6ae_c87f5403e983 |
+--------------------------------------+
1 row in set (0.02 sec)
```

- Grab a coffee!

// Online Schema Changes

Based on VReplication

```
mysql> SHOW vitess_migrations LIKE '4d8246f2_9801_11ed_a6ae_c87f5403e983'\G

*************************** 1. row ***************************
...
     migration_uuid: 4d8246f2_9801_11ed_a6ae_c87f5403e983
    added_timestamp: 2023-01-19 14:58:08
completed_timestamp: 2023-01-19 14:58:18
   migration_status: complete
migration_statement: alter table table1 change column id id bigint unsigned
           strategy: online
...
1 row in set (0.00 sec)
```

But there is more…

- Migrations become fully reversible
- Because the VReplication stream could easily be reversed, the old table is kept up-to-date.
- Now a revert operation is as simple as

```
mysql> REVERT vitess_migration '4d8246f2_9801_11ed_a6ae_c87f5403e983';
+--------------------------------------+
| uuid                                 |
+--------------------------------------+
| 20f5337f_9826_11ed_a6ae_c87f5403e983 |
+--------------------------------------+
1 row in set (0.04 sec)
```

Declarative migrations

- No need to write `ALTER` statements anymore, just resubmit the `CREATE TABLE` statement and Vitess will figure out the difference and run the migration…

```
mysql> SET @@ddl_strategy = 'online -declarative';
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE TABLE `table1` (
    ->   `id` bigint unsigned NOT NULL,
    ->   `data` varchar(512) DEFAULT NULL,
    ->   PRIMARY KEY (`id`)
    -> ) ENGINE=InnoDB;
+--------------------------------------+
| uuid                                 |
+--------------------------------------+
| c423f39b_982c_11ed_a6ae_c87f5403e983 |
+--------------------------------------+
1 row in set (0.02 sec)
```

Conclusion

- Schema changes are being put back into the hands of the developers!
- Easy to run
- Easy to revert
- Due to the robustness of VReplication, it can even survive a primary failover or other type of outage

Automatic failovers

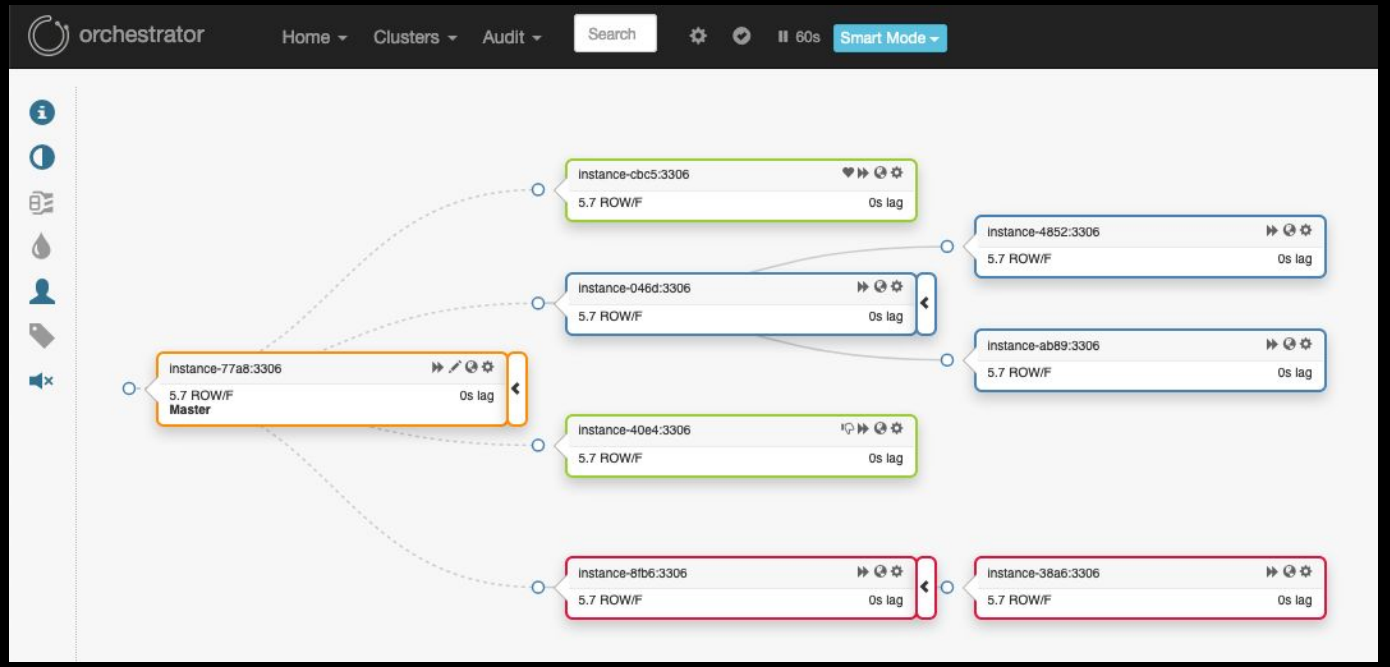
// Automatic failovers

Orchestrator

Orchestrator is open-source software commonly used as a MySQL Topology management tool. It's purpose is to observe MySQL replication topologies and potentially fix these topologies when failures are observed.

Core functionalities:

- Discover
- Visualize
- Monitor
- Refactor



// Automatic failovers

VTOrc

- Vitess component based on Orchestrator
- Stateless, topology information is stored in the topology service
- Discovery is already taken care of by Vitess
- Refactoring will automatically update the topo
- Requirements
  - At least one replica
  - Optional: semi-sync replication

Questions?

Thank you!

Matthias, Crauwels

Enterprise Customer Engineer, PlanetScale

@mcrauwel