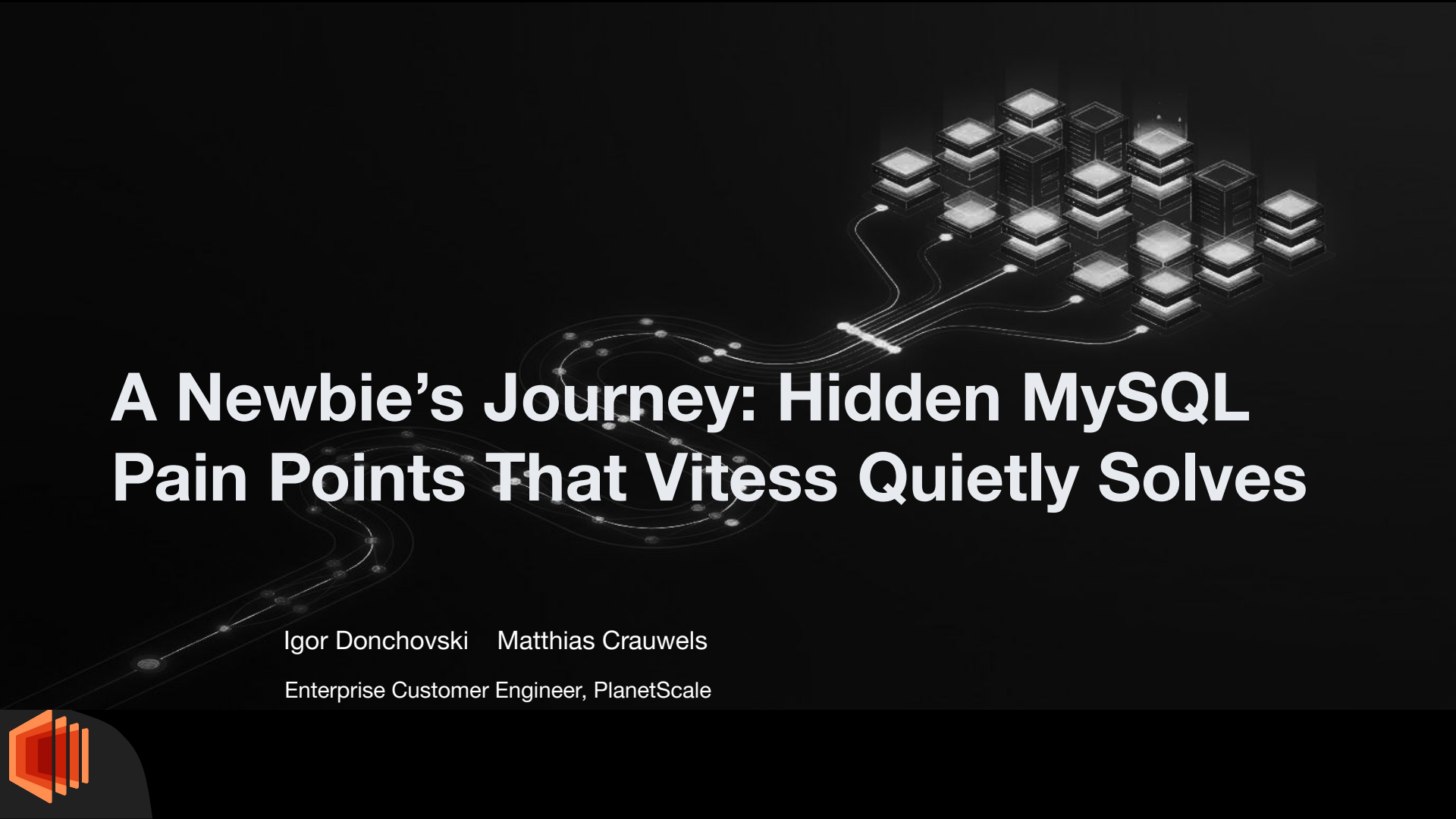


# A Newbie’s Journey: Hidden MySQL Pain Points That Vitess Quietly Solves

Igor Donchovski  Matthias Crauwels

Enterprise Customer Engineer, PlanetScale

# About me

👨‍💻 Enterprise Support Engineer - Planetscale

🎓 Education: MSc in Software Engineering

📜 Certifications

- MongoDB Certified DBA
- Oracle Professional MySQL 5.7
- Terraform Associate Certified
- GCP Professional Architect

💻 Expertise: 20+ years in IT, 15 years with MySQL, 10 years MongoDB

🌟 Personal: Husband and Father, Avid Traveler, Speaker

👨‍💻 Enterprise Support Engineer - Planetscale

🎓 Education: Bs Applied Computer Science

📜 Certifications

- Oracle Professional MySQL 5.7 and 8.0
- MariaDB Certified 10.3
- AWS, GCP and Azure certified

💻 Experience: 20+ years in IT, first as PHP developer, ~15 years as MySQL DBA, 4 years with Vitess.

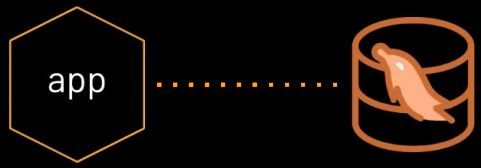
🌟 Personal: Husband, Father, Traveller, Soccer player, Speaker

# Agenda

- MySQL Limits
- Online DDL
- Query Management
- High Availability
- Disaster Recovery
- Vertical Scaling
- Horizontal Scaling
- Vitess

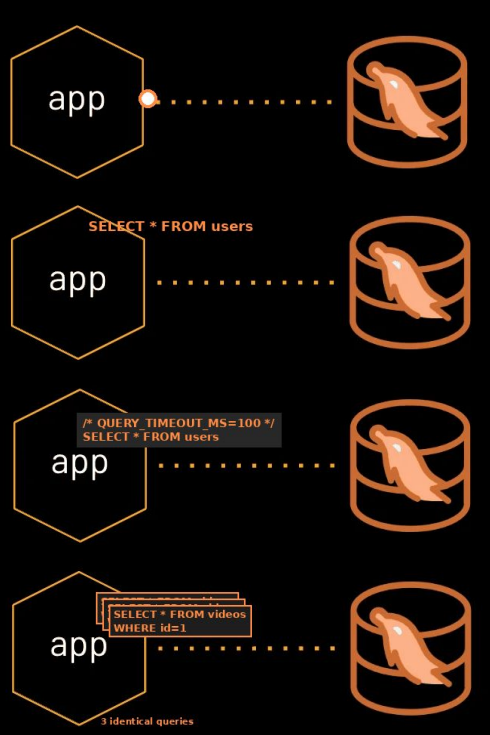
# MySQL

- Physical limits (32TB RAM, 60-144+ cores per socket (8 sockets CPU))
- Database size (no actual limits)
- Table size (64TB with 16k page size)
- Max connections (100k)
- Max Threads Running (< 100k)
- Number of rows (18 quintillion for bigint)



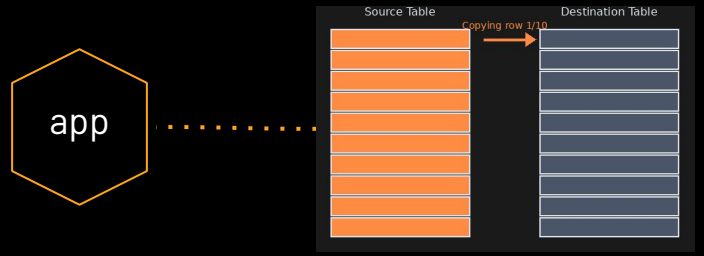

# MySQL

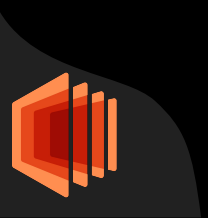
- Query timeout
- Limit result set size
- Limit rows returned
- Comment Directives
- Query consolidation



# MySQL

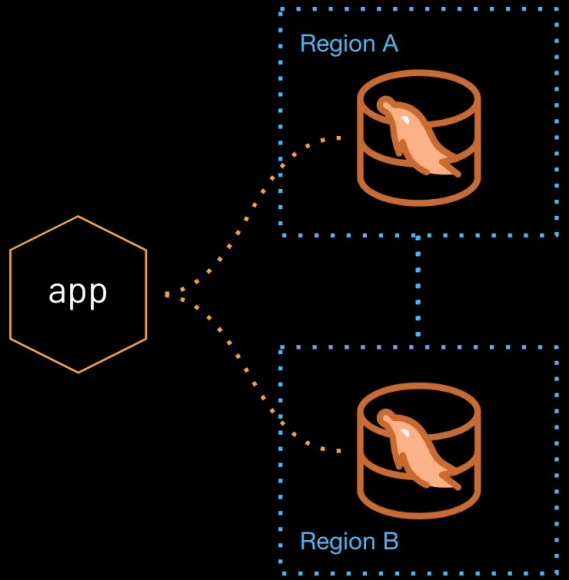
- DDL
  - MySQL instant ddl
  - pt-online-schema-change
  - gh-ost
  - spirit
- Resume on failure
- Revert the change



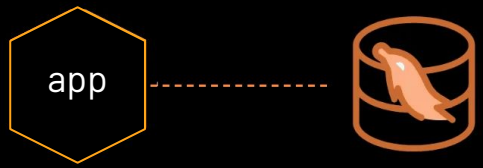
# MySQL

- Zero downtime migration
- Move/Copy a subset of the tables
- Checksum the data
- Switch reads from A to B
- Switch writes from A to B
- Revert reads/writes from B to A



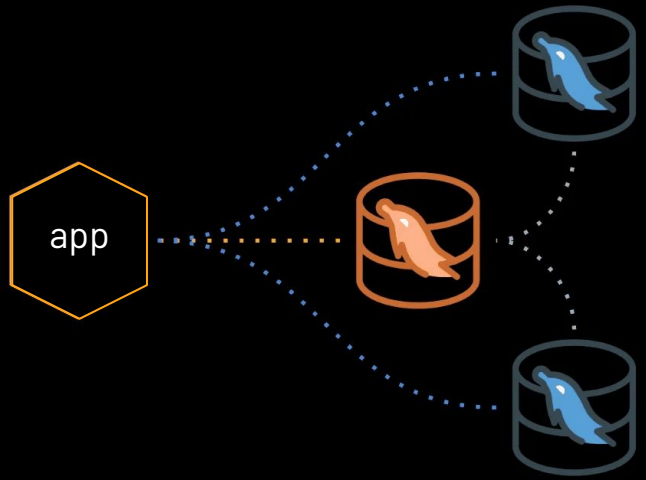

# MySQL - HA

- Only use the Primary
- Always read from a Replica
- Skip reading from replica if lag > $X
- Min serving replicas
- Disaster recovery (A,B,C)



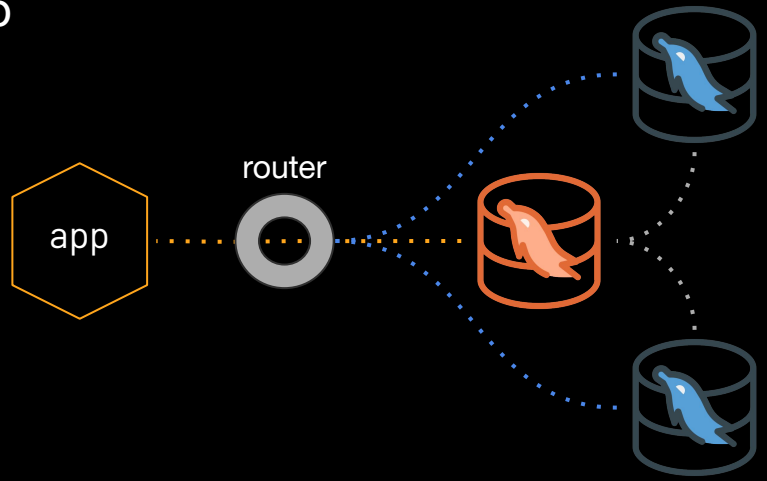

# MySQL - HA

- Primary failure
- Lost transactions
- Replica failure
- App errors
- Replica clone and warm up



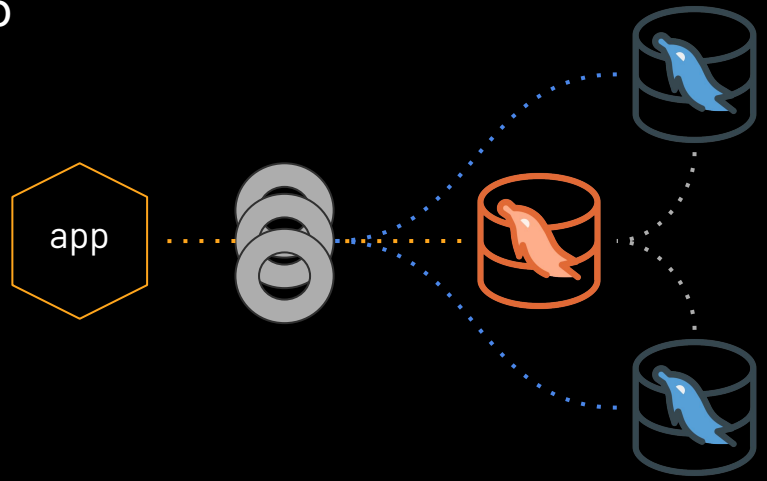

# MySQL HA

- Remove the complexity from the app
- Router as a middle layer
- Semi sync replication
- Synchronous replication

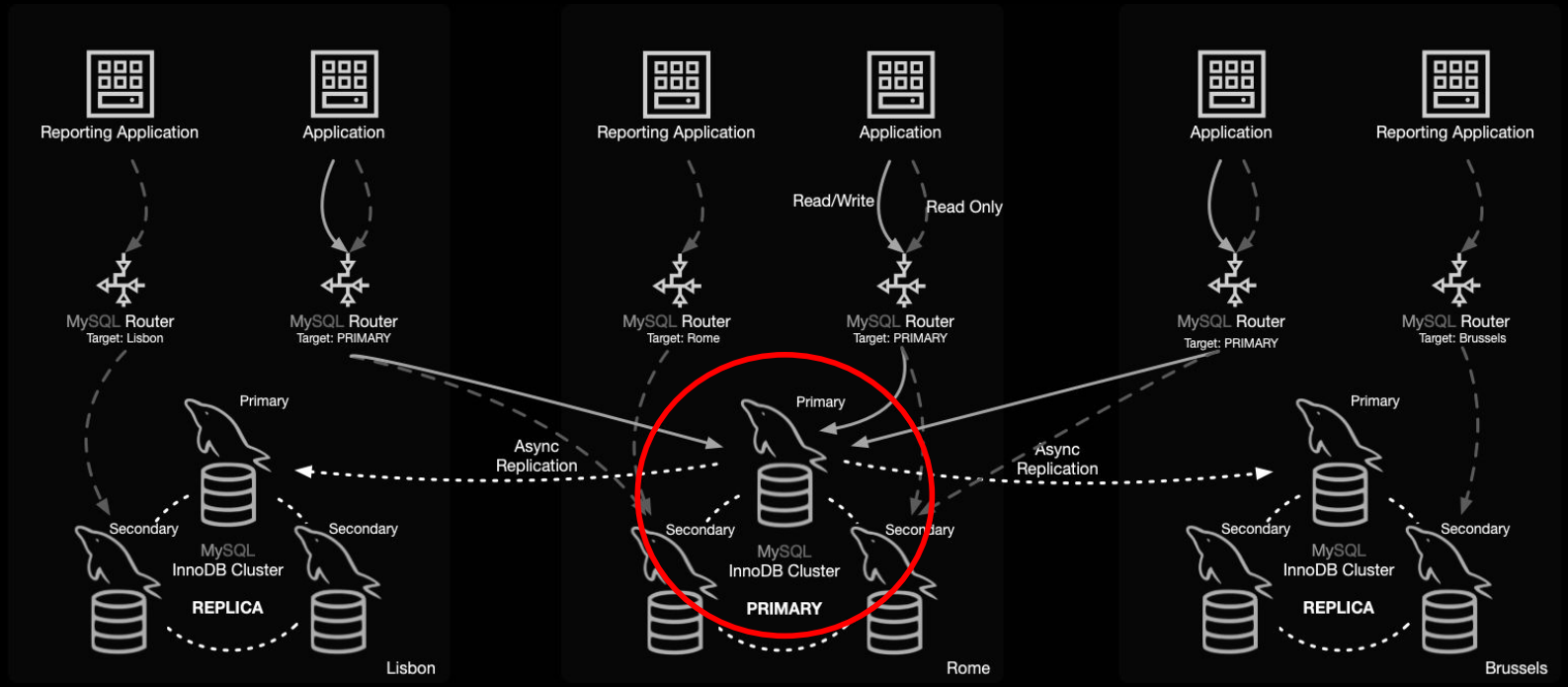



# MySQL HA

- Remove the complexity from the app
- Router as a middle layer
- Everything as HA

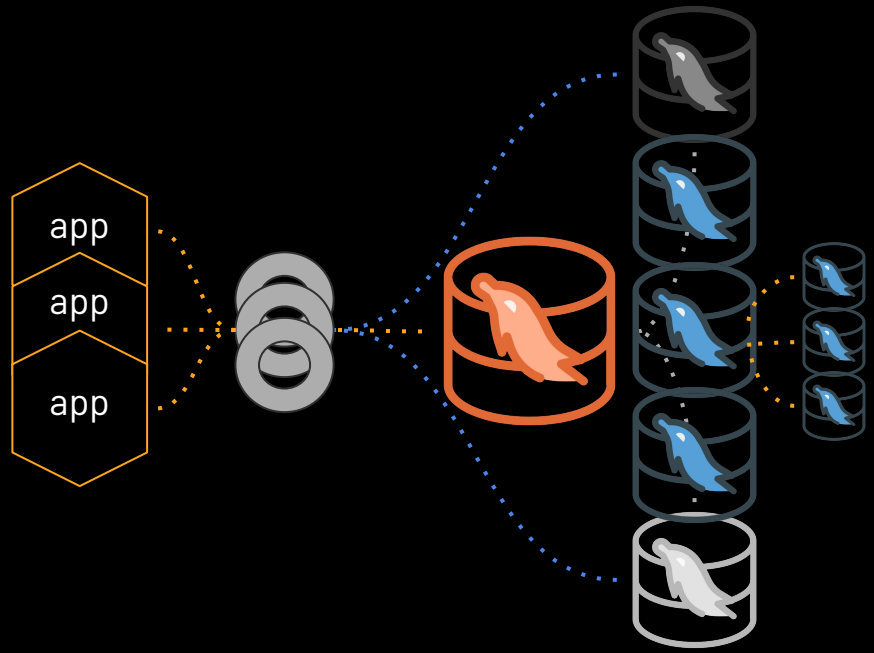



# MySQL HA and DR



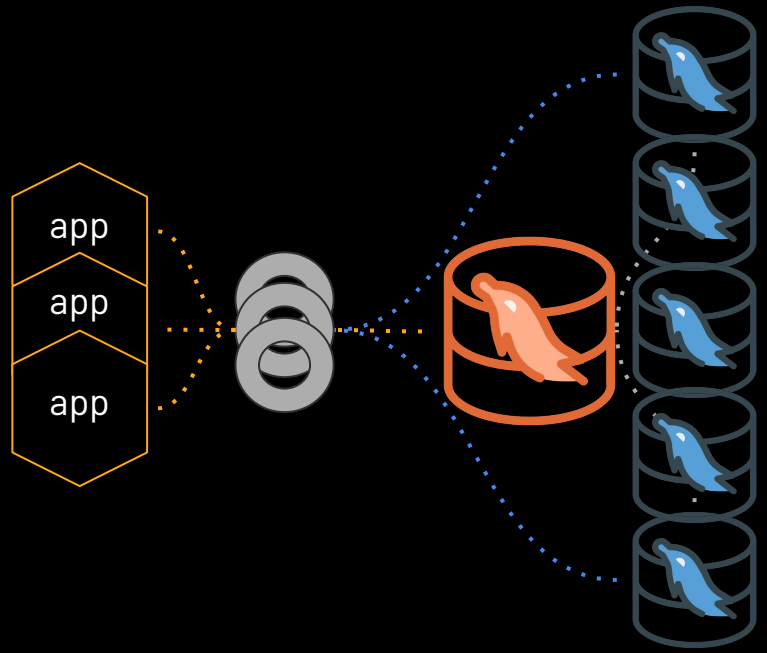
# MySQL - Vertical scaling

- Increased workload
- Reporting replicas
- Dedicated backup replica
- Delayed replica
- Intermediary primary/replica



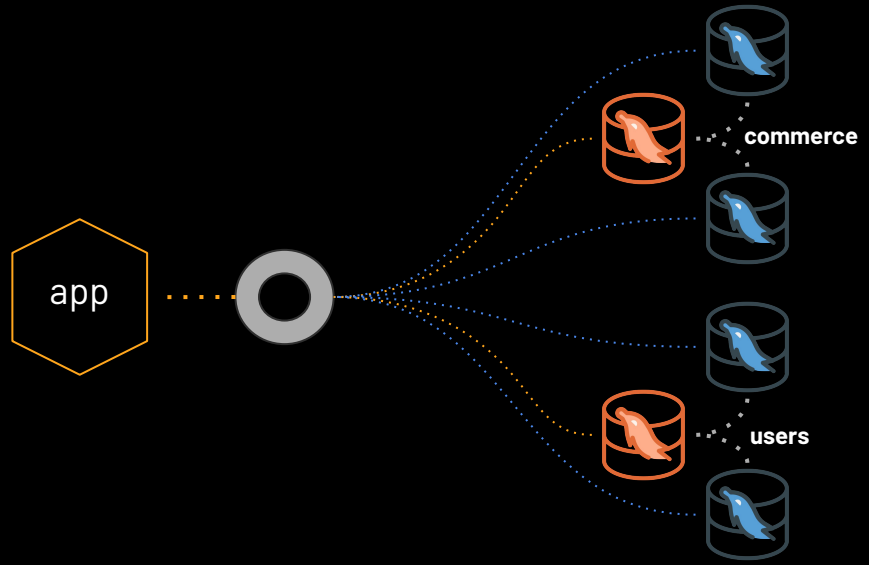

# MySQL - Vertical scaling

- Long backups
- Long restores
- Long DDL
- Lock contention
- Exponential costs
- Performance is not linear



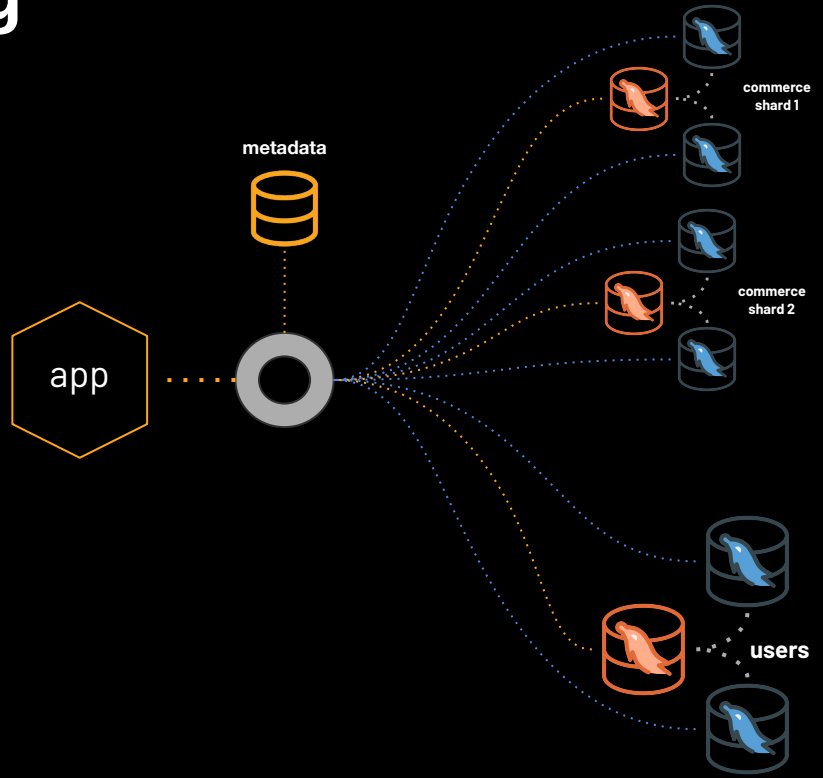

# MySQL - Vertical scaling

- Divide and conquer
- Scale to largest machines
- Scale read capacity
- Normalization
- Referential integrity

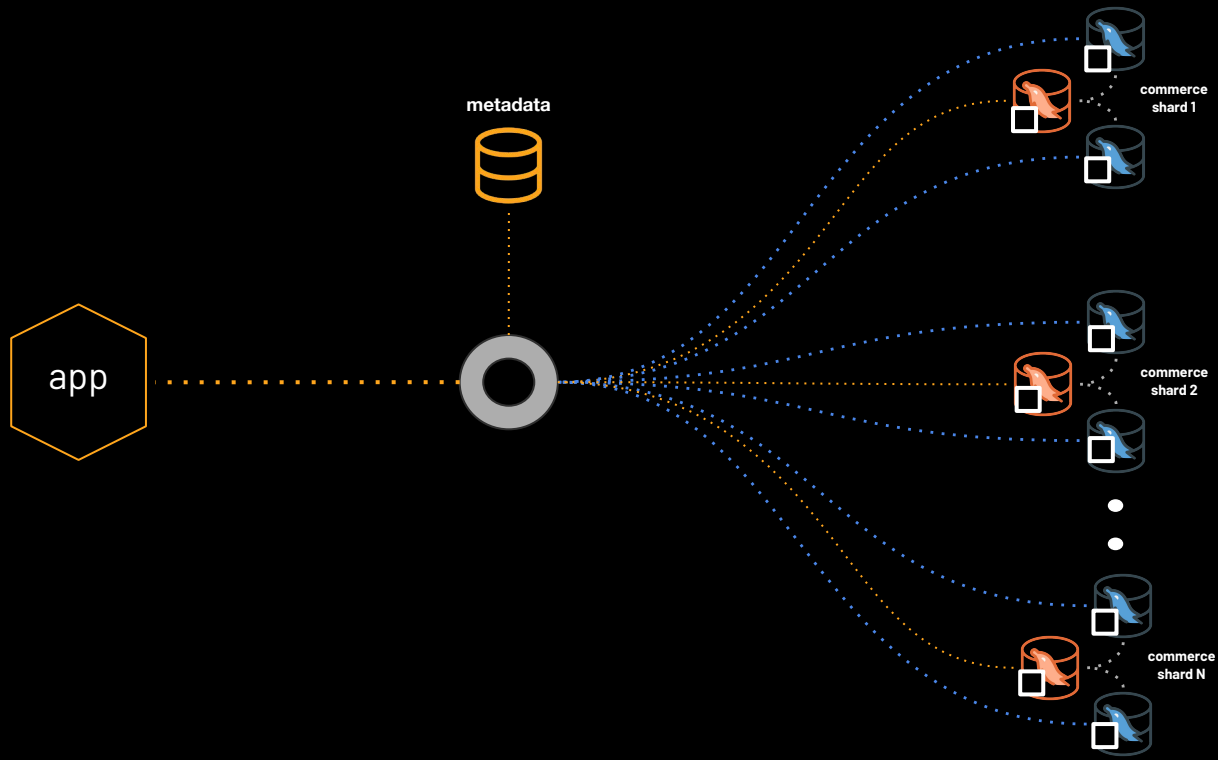



# MySQL - Horizontal scaling

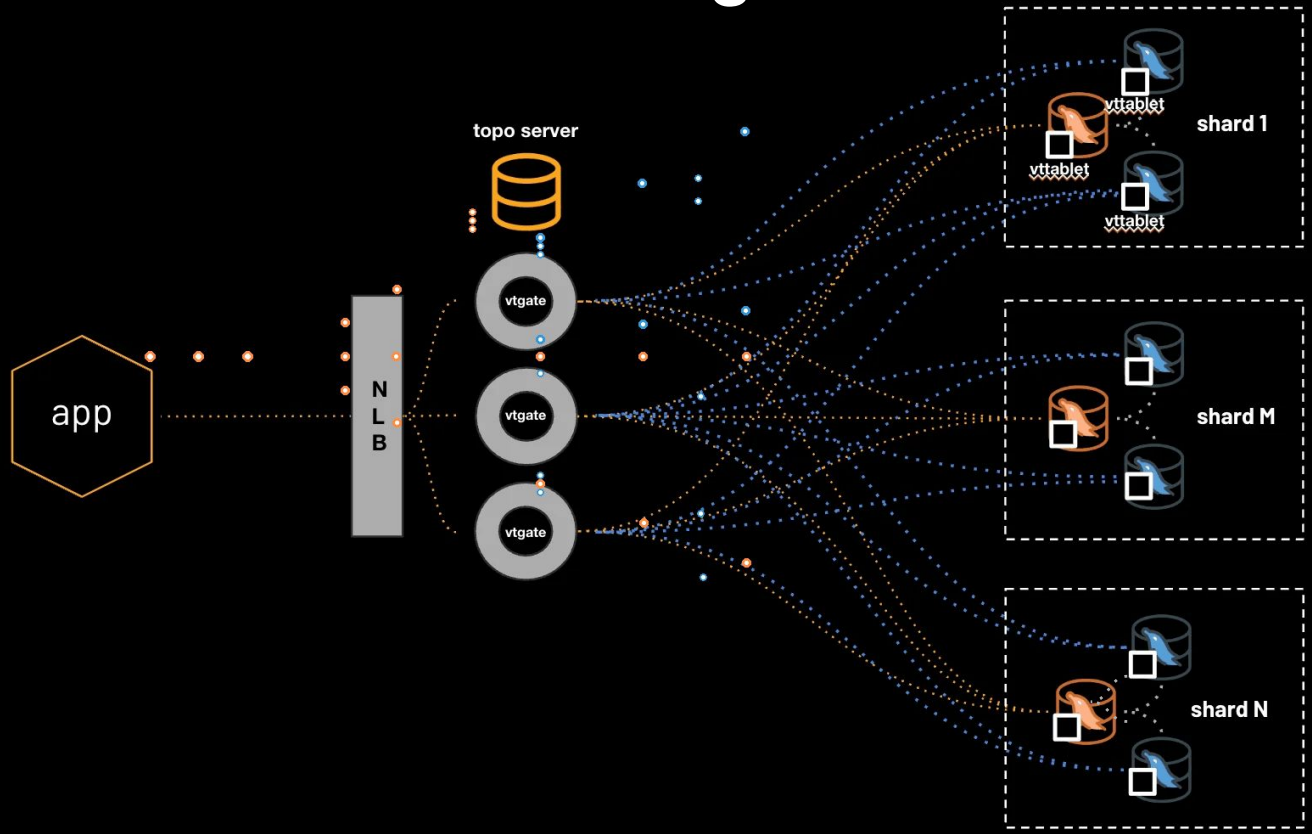
- App logic where to send the query
- Primary failure
- Limiting the result sets
- Resharding
- Updating the metadata
- Restoring a node



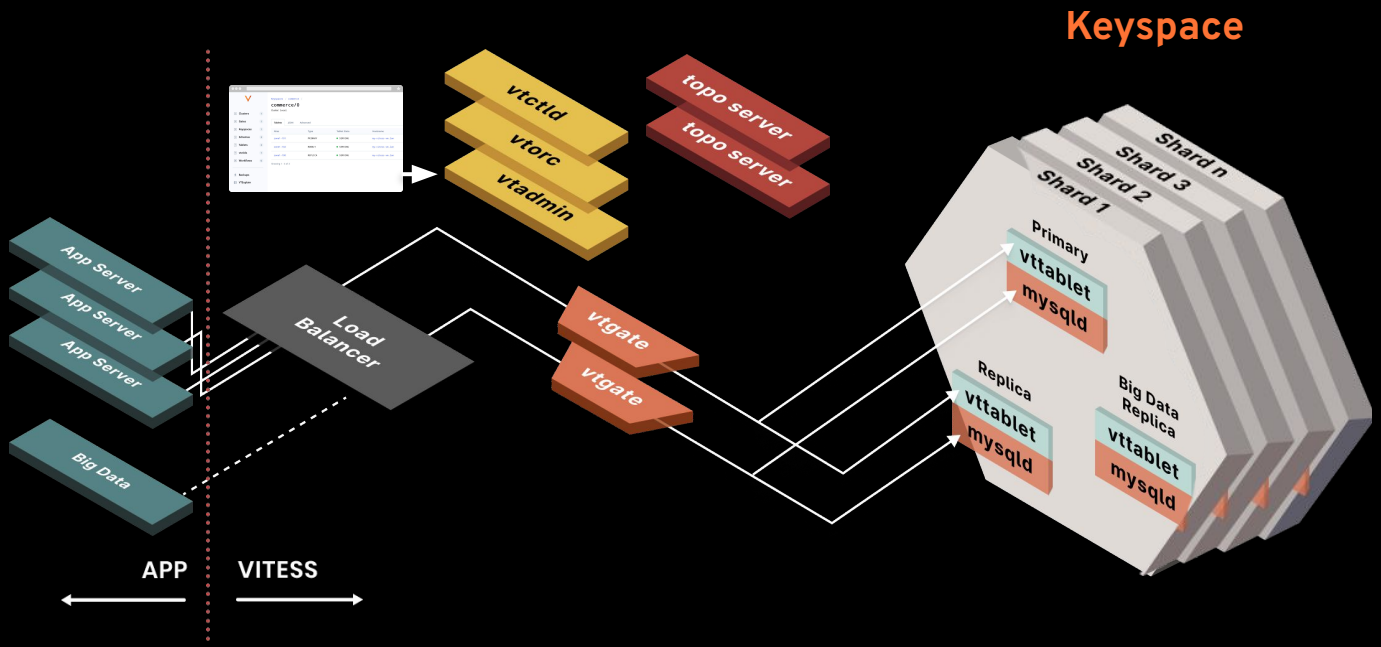
# MySQL - Horizontal  Scaling



# MySQL - Horizontal  Scaling

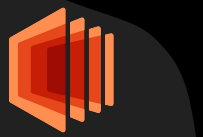


# Vitess Architecture




# Vitess serves millions of QPS in production

slack
New Relic
Square
Flipkart
HubSpot
peak
Pinterest
shopify
nozzle
weave
GitHub
JD.COM 京东
Quiz of Kings
stitchlabs
PlanetScale



# Concepts

- Keyspace
- Shard
- Topology server

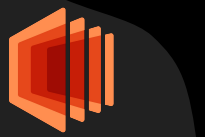
# Vitess provides an illusion

- A single database (server)
- MySQL 8.0 and above
- A single, dedicated connection

# Vitess needs to deal with

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

# VReplication



# Use cases

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

# How it works?

- Copy phase: `SELECT * FROM table [WHERE id > ?]`
  - Streams rows in PK order
  - Keeps track of last copied PK
  - Default duration 1h (configurable)
- Replication phase
  - Catch up on binary logs
  - Keeps track of replication position (GTID preferred)

# 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

# ALTER TABLE issues

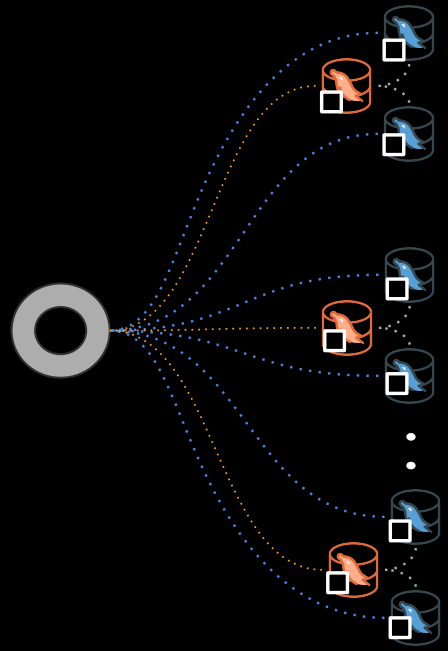
- Blocking
- Resource greedy
- Not auditable
- Not interruptible

# 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!



# 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

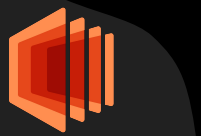
```
mysql> REVERT vitess_migration '4d8246f2_9801_11ed_a6ae_c87f5403e983';
+--------------------------------------+
| uuid                                 |
+--------------------------------------+
| 20f5337f_9826_11ed_a6ae_c87f5403e983 |
+--------------------------------------+
1 row in set (0.04 sec)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

# 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

# Thank you!