VTOrc

How Vitess achieves Consensus with Replicated MySQL

Deepthi Sigireddi

Vitess

@vitessio

Deepthi Sigireddi

Technical Lead
Vitess

@ATechGirl
@atechgirl.hachyderm.io

Vitess

@vitessio

Vitess Overview

@vitessio

What is Vitess?

MySQL compatible

Massively Scalable

Highly Available

Data Durability

Vitess

@vitessio

Features

MySQL compatibility

- Compatible with popular frameworks

Management features

- Connection Pooling
- Online schema changes
- Query consolidation

Features

High Availability

- Failure detection and failover

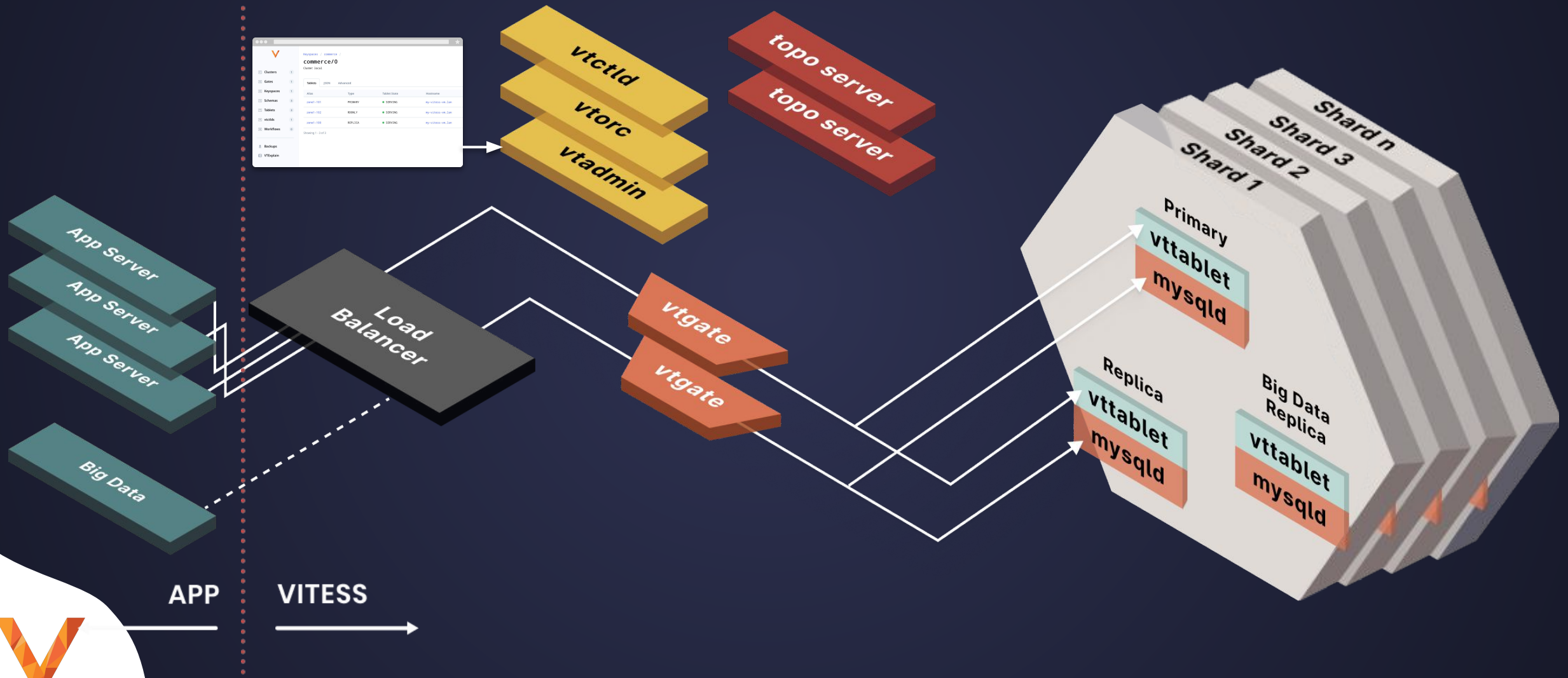
Data Durability

- Replication

Scalability through Sharding

Data migrations, views, roll-ups, CDC

Architecture Summary



Vitess

Consensus in Vitess

Problem Statement

- How to recover from MySQL failures
- While guaranteeing
  - High availability
  - Data Durability
  - Minimal downtime / recovery time

Design Principles

- Engineering approach
- Single leader system
- Fulfill requests while respecting durability policy
- Leader election process
  - Planned versus unplanned
- Forward Progress
- Race conditions

Concepts

Keyspace

- Logical database

Shard

- Slice of data

Cell

- Failure Domain

Shard topology

A replicated database cluster with primary and replicas

Vitess

VTTablet

Each MySQL server is assigned a **vttablet**

- A daemon/sidecar
- Controls the **mysqld** process
- Interacts with the **mysqld** server
- Typically on same host as **mysqld**

Vitess

VTOrc

- Rewrite of openark/orchestrator
- Agent that detects and repairs failures
- Durability through Replication
  - Policies allow trade-offs
- High availability through failover
  - Planned / unplanned leader election

Vitess

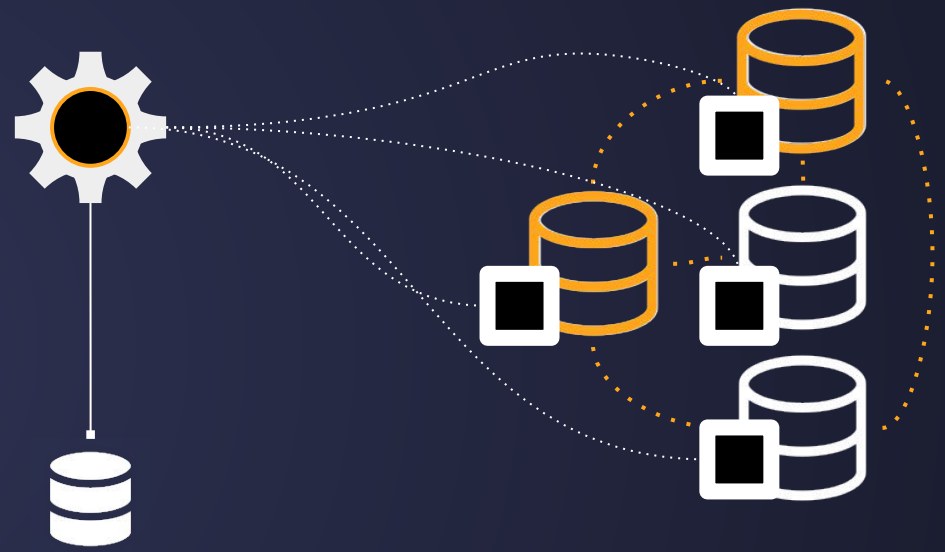
@vitessio

Leader Election

- Revocation
- Election
- Propagation

Planned Leader Election

- Revocation
  - Current leader is asked to step down
- Leader selection
  - A new leader is chosen
- Propagation
  - Completed requests

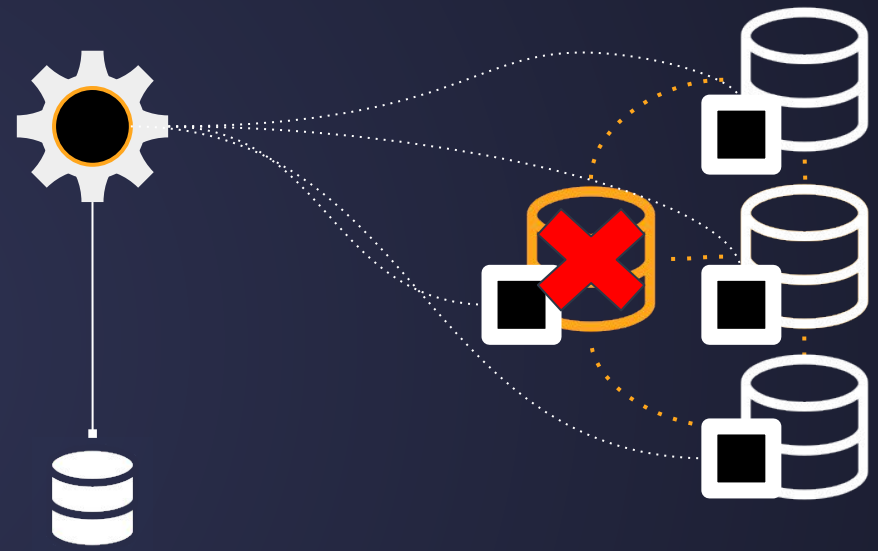


Vitess

@vitessio

Unplanned Leader Election

- Revocation
  - Reach “m” followers
- Leader selection
  - A new leader is chosen
  - Based on durability policy
- Propagation
  - Completed requests



Vitess

@vitessio

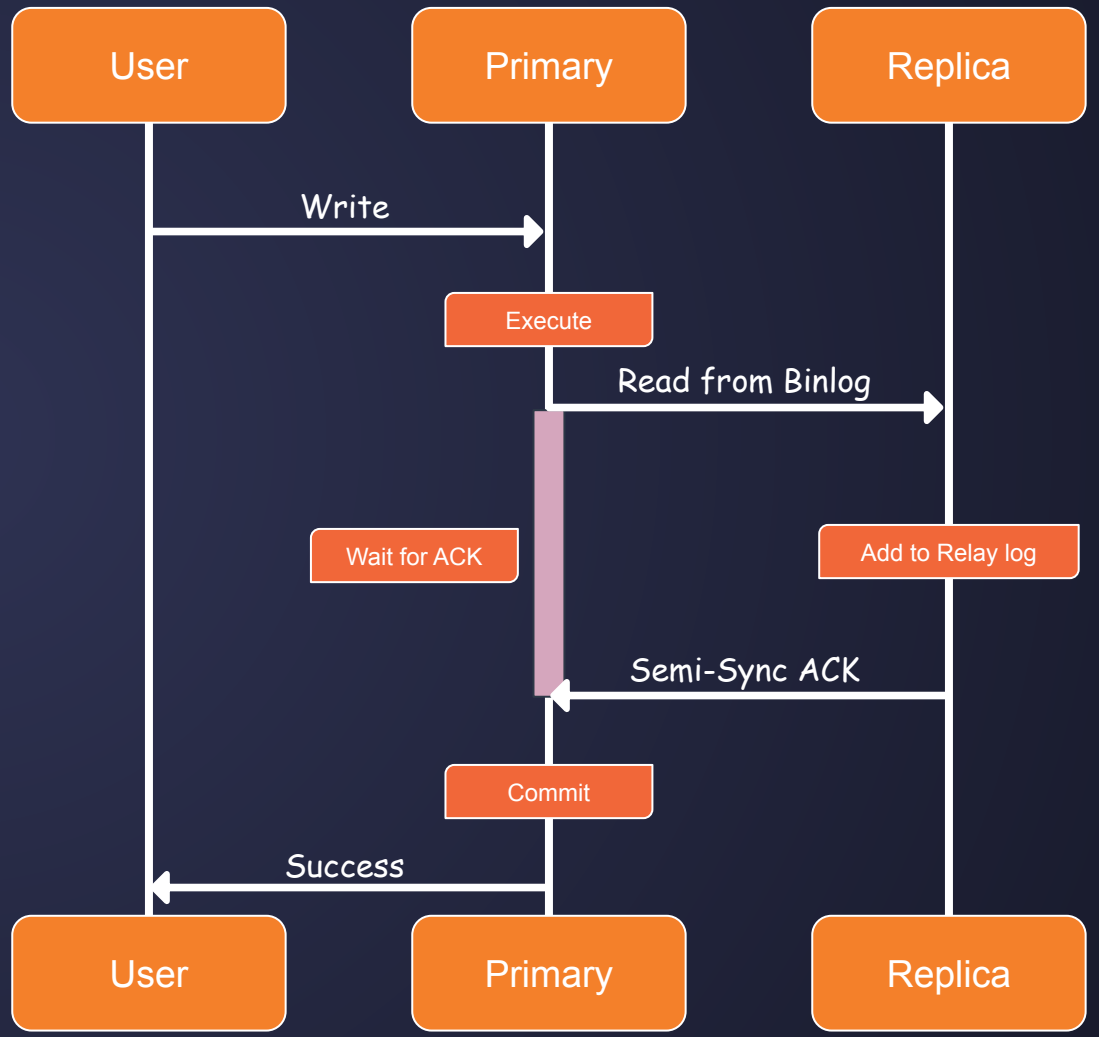
Revocation and Quorum

- What is “m”?
- How do we know we have reached sufficient tablets to guarantee safety?

- Intersecting Quorum
- Quorum for accepting transactions
- Quorum for revocation

Durability Policies & Semi-Sync

- Durability Policy
  - Who can be the primary?
  - How many semi-sync ACKs required for each primary?
  - Who can send these ACKs?
- Increased Flexibility
  - None - Default policy
  - Semi-Sync
  - Cross-Cell
  - Custom

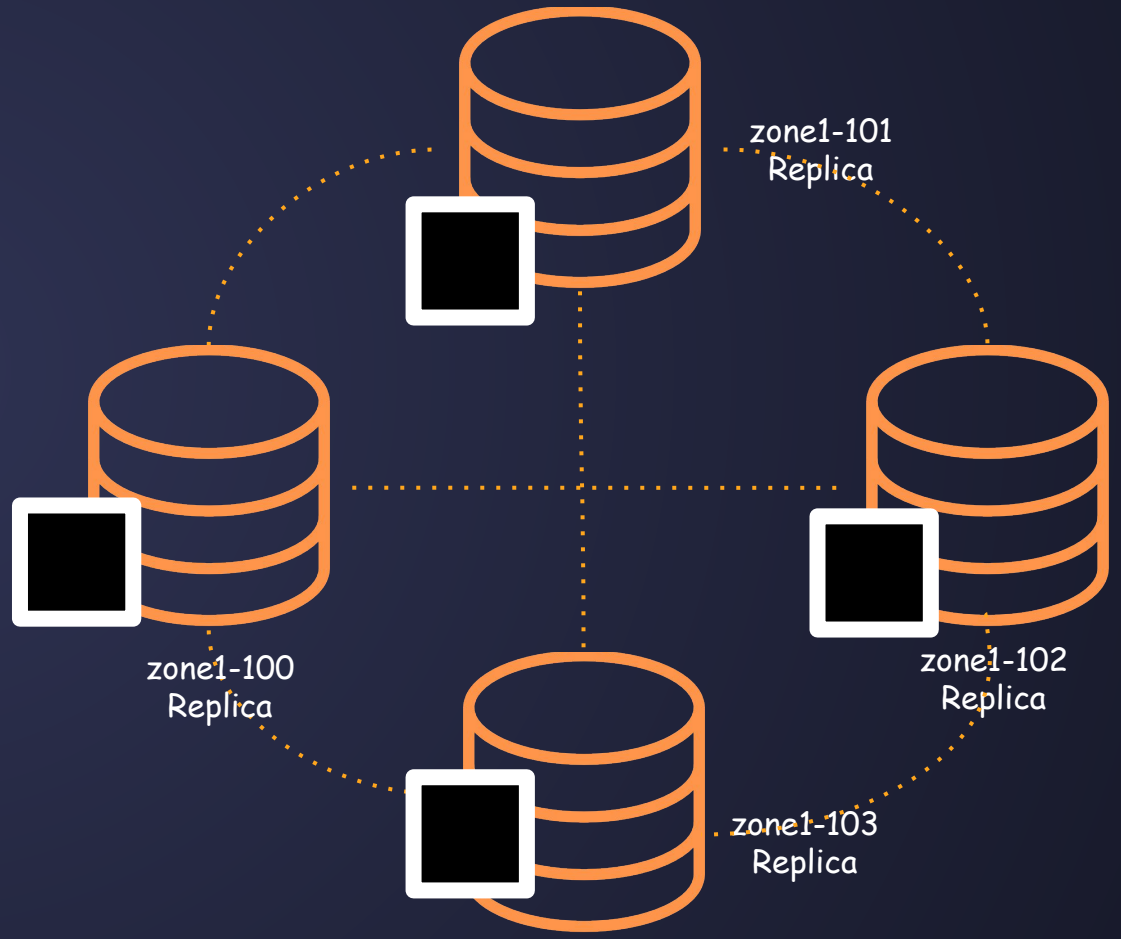


Vitess

@vitessio

Semi-Sync Durability

- Durability Policy - semi-sync
  - Any replica can be the primary
  - 1 semi-sync ACK required
  - Any replica can send the ACK

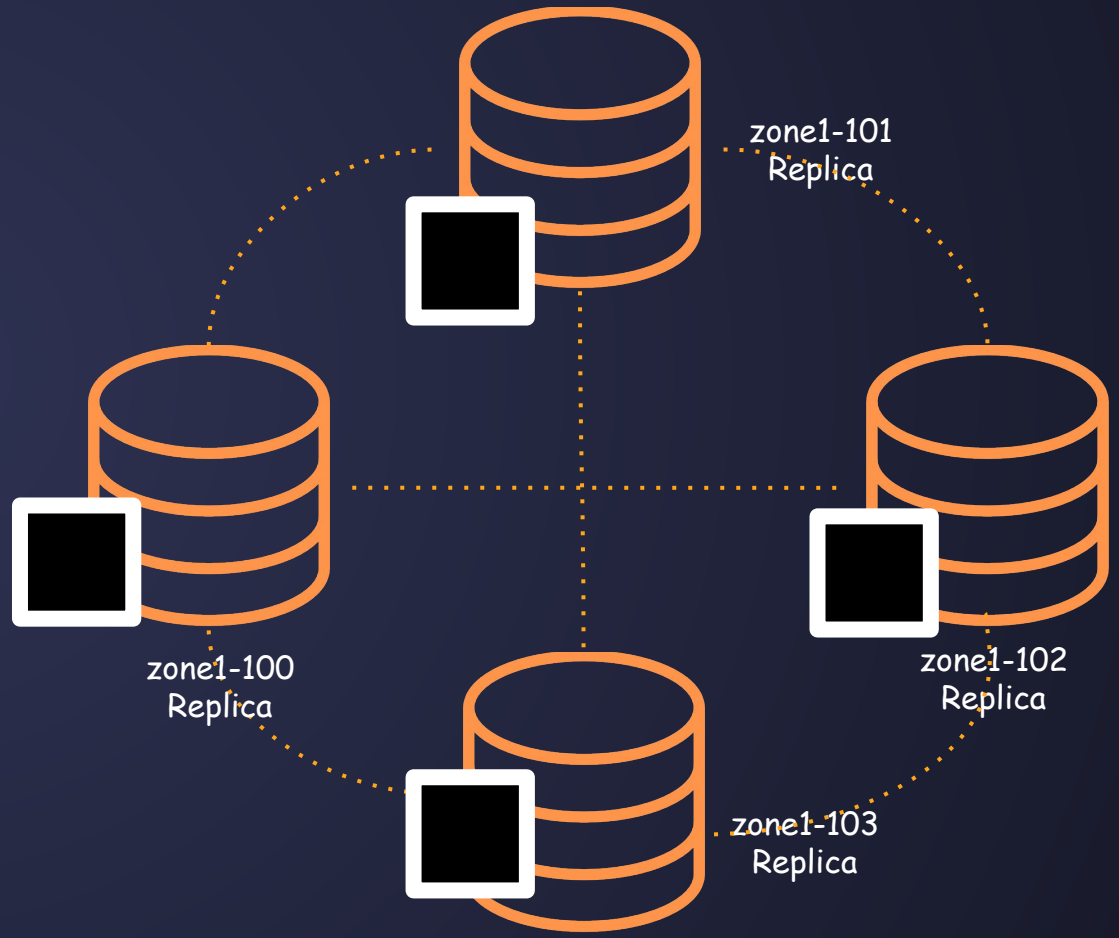


Vitess

@vitessio

Revocation

- **Quorums for Accepting Transactions**
  - [(**100**, 101), (**100**, 102), (**100**, 103)]
  - [(**101**, 100), (**101**, 102), (**101**, 103)]
  - [(**102**, 100), (**102**, 101), (**102**, 103)]
  - [(**103**, 100), (**103**, 101), (**103**, 102)]
- **Quorums for Revocations -**
  - [100, 103] ❌
  - [100, 102, 103] ✅
  - [100, 101, 102, 103] ✅
  - [101] ❌



Vitess

@vitessio

Revocation and Quorum

Vitess

@vitessio

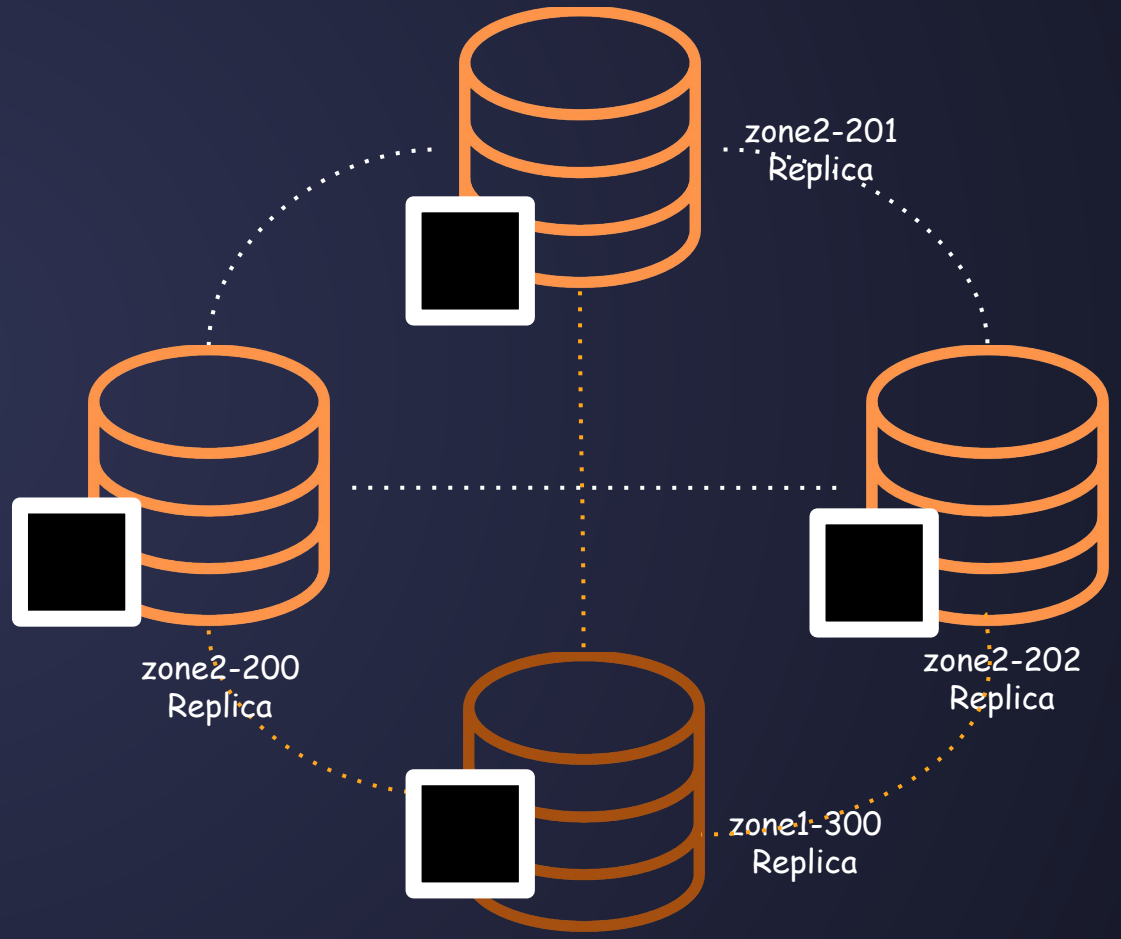
Revocation and Quorum

Vitess

@vitessio

Cross-Cell Durability

- Cell = Failure Domain
- Durability Policy - cross-cell
  - Any replica can be the primary
  - 1 semi-sync ACK required
  - Only a replica from a different cell can send an ACK



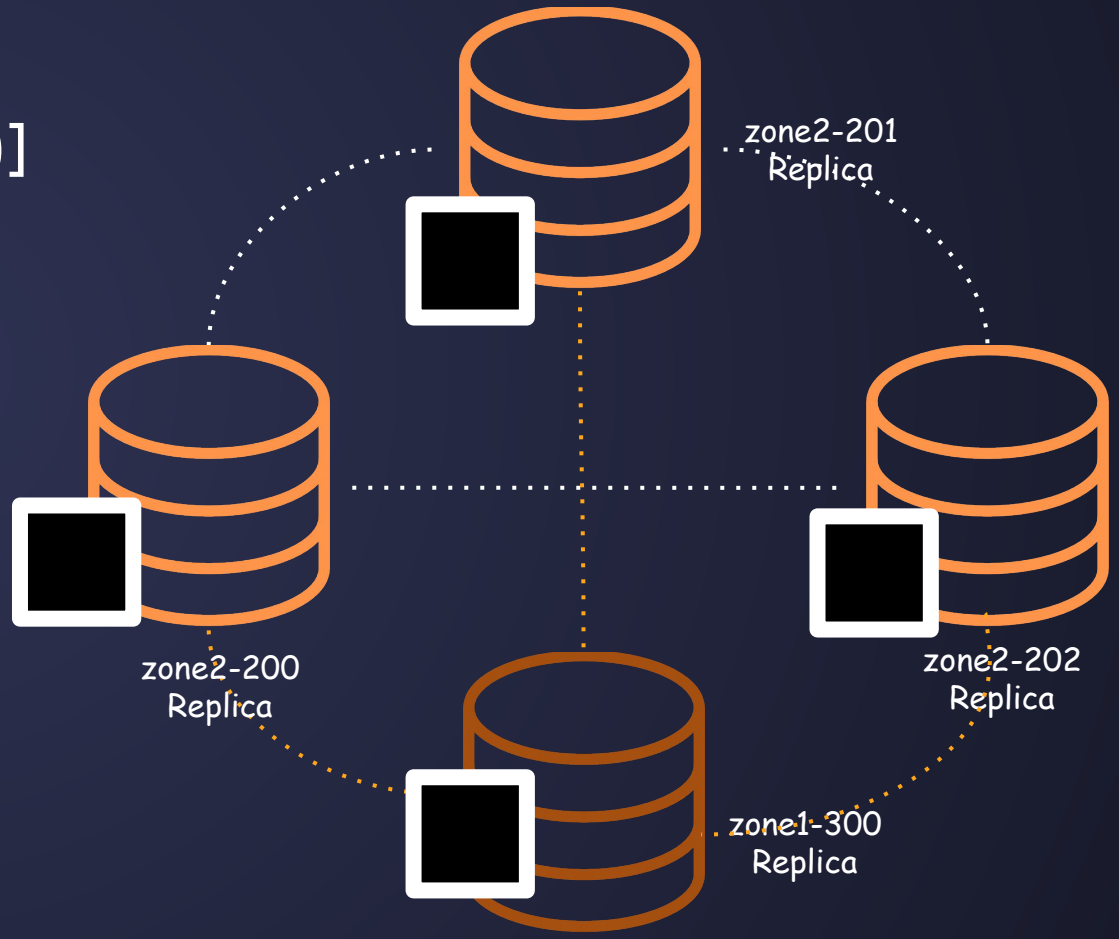
Vitess

@vitessio

Revocation

- **Quorums for Accepting Transactions**
  - [(**300**, 200), (**300**, 201), (**300**, 202)]
  - [(**200**, 300)]
  - [(**201**, 300)]
  - [(**202**, 300)]

- **Quorums for Revocations -**
  - [200, 201] ❌
  - [300, 202, 201] ✅
  - [300, 201] ✅
  - [300, 200, 201, 202] ✅



Vitess

@vitessio

More Failure Scenarios

- Primary is Read-Only
- Replica's replication is stopped
- Replica is writable
- Semi-sync settings are incorrect
- Shard has no primary
- Primary is replicating from a different tablet
- ErrantGTID detection

Future Work

- ErrantGTID handling
  - Uncommitted transactions when server went down
  - Committed when server comes back
  - Propagate with new GTID??
  - Rewind??
- Reduce / remove dependency on external locks

Resources

Blog Post Series

- https://planetscale.com/blog/blog-series-consensus-algorithms-at-scale-part-1

Documentation

- https://vitess.io
- https://vitess.io/docs/16.0/reference/vtorc/
- https://vitess.io/docs/16.0/user-guides/configuration-basic/vtorc/

Community

- https://github.com/vitessio/vitess
- https://vitess.io/slack

Vitess

@vitessio

Q & A

@vitessio

Thank you!

@vitessio