Smooth Scaling

Slack's journey to a new database

Smooth Scaling

Slack's journey to a new database

@ameetkotian

Slack is growing

2014

Launch Year

8M+

Daily Active Users in 100+ countries

65%

of the Fortune 100 companies

3.5B

HTTP requests per day

42B

database queries per day

2.1 PB

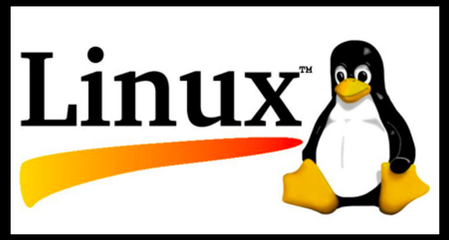
of database storage

Agenda

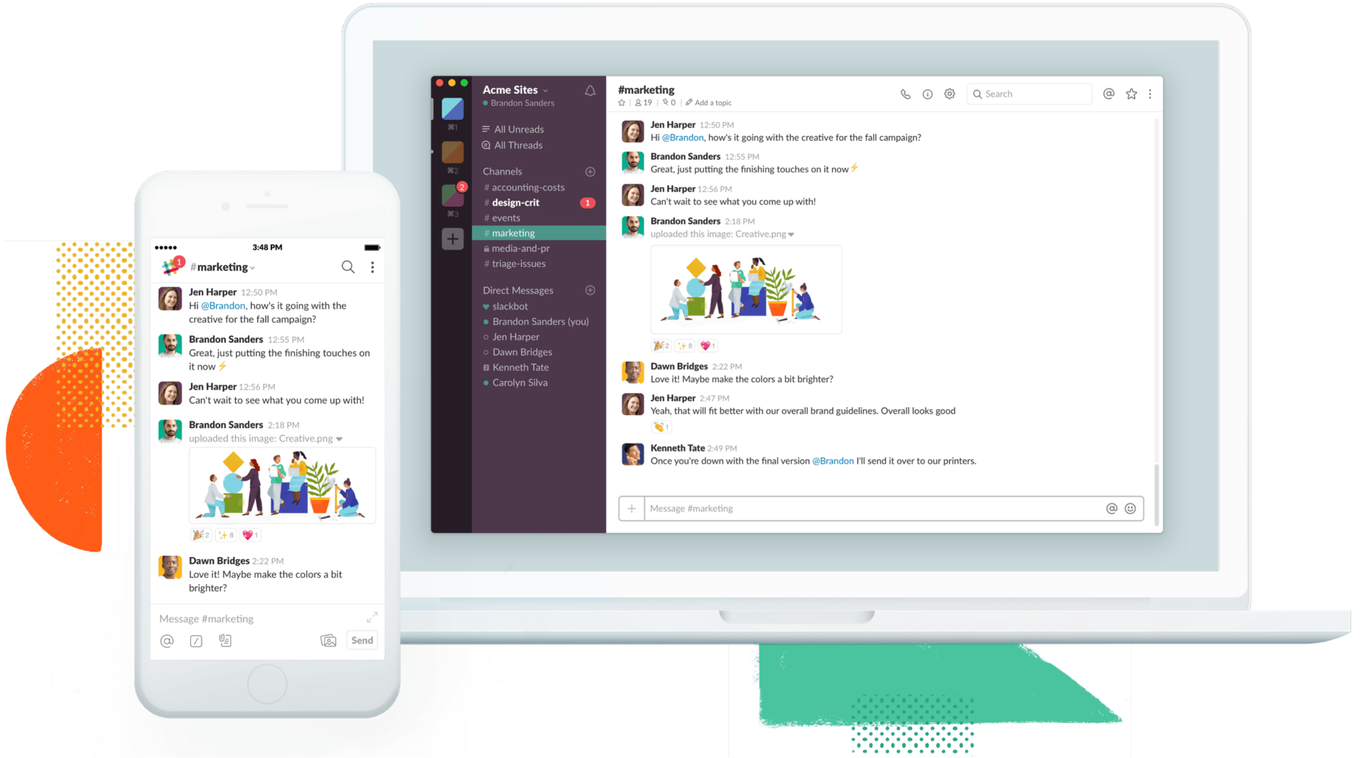
- Original database setup with challenges
- What is Vitess?
- Current state of migration and future

Slack's stack is simple

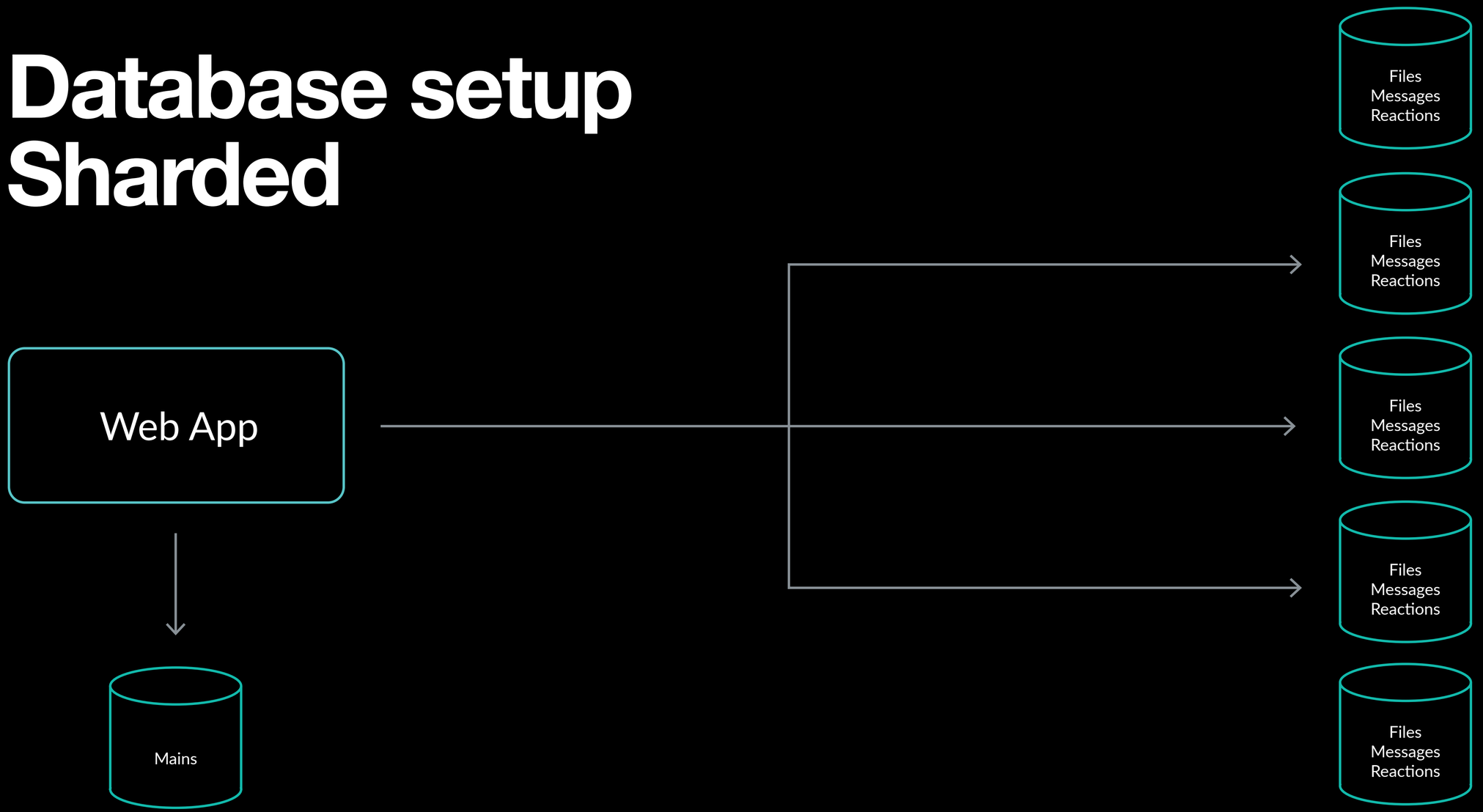
But simple is good.



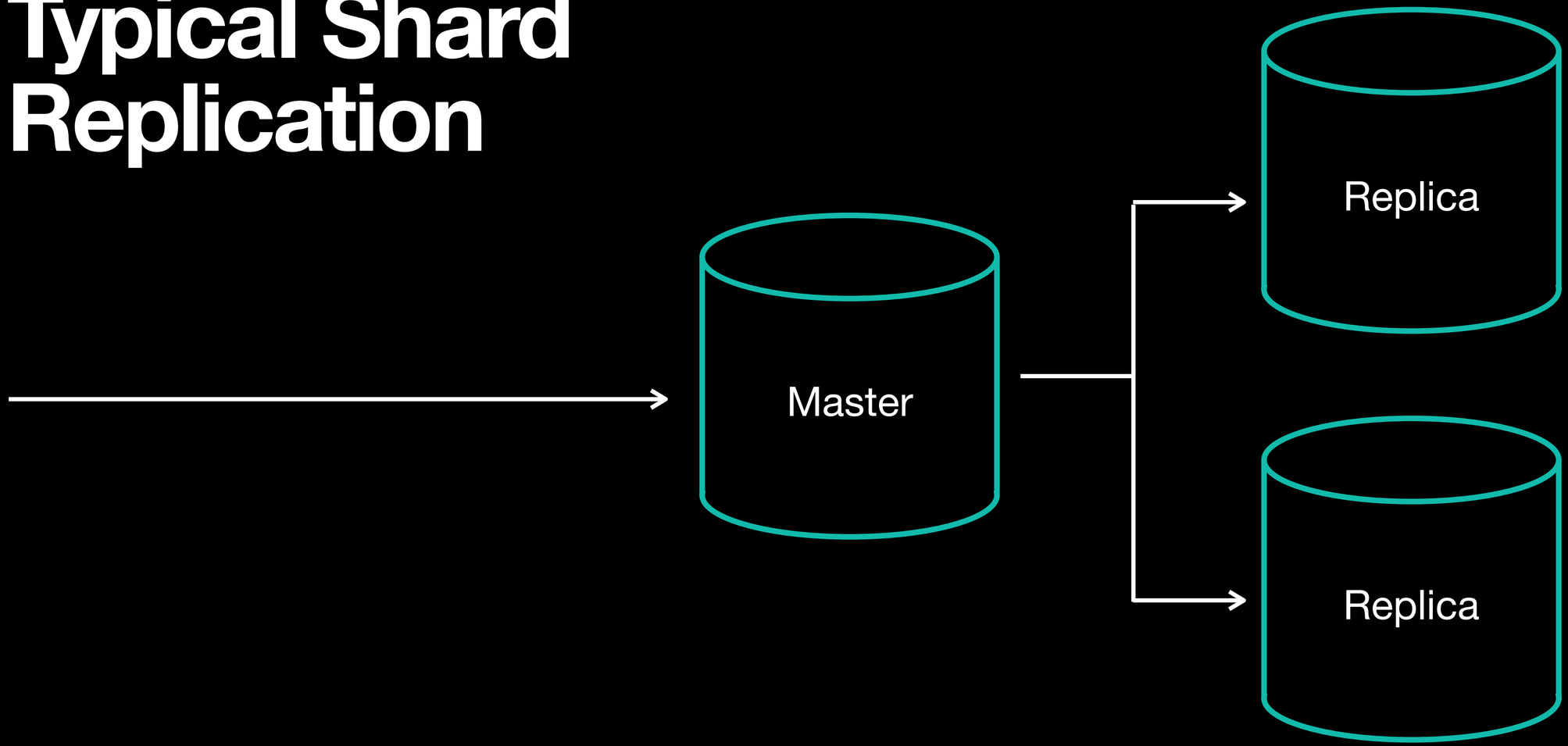
Workspaces <-> Teams



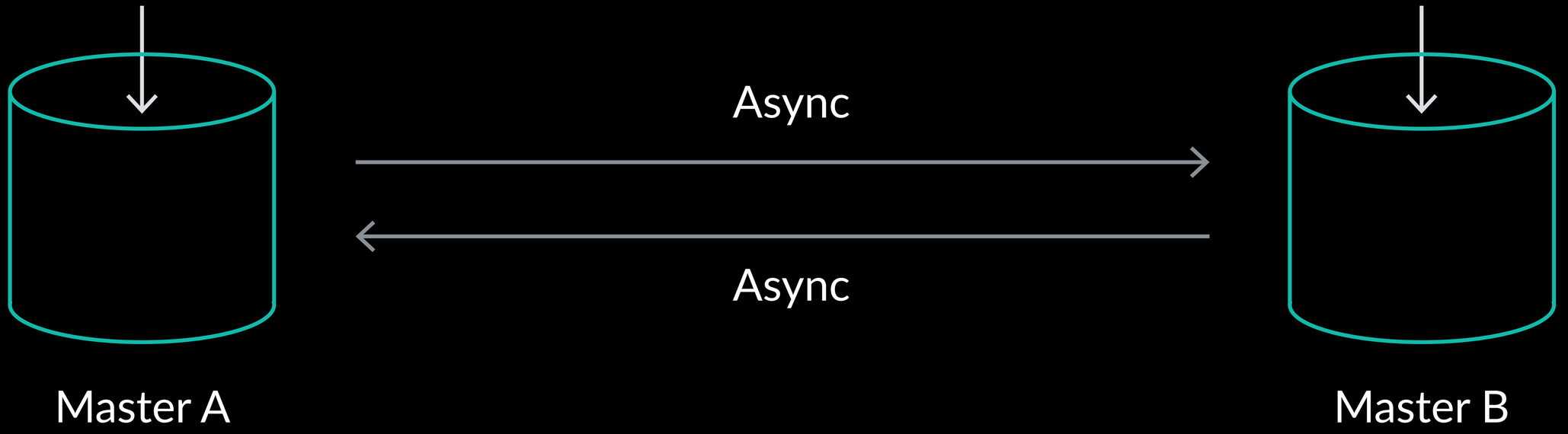
Database setup
Sharded



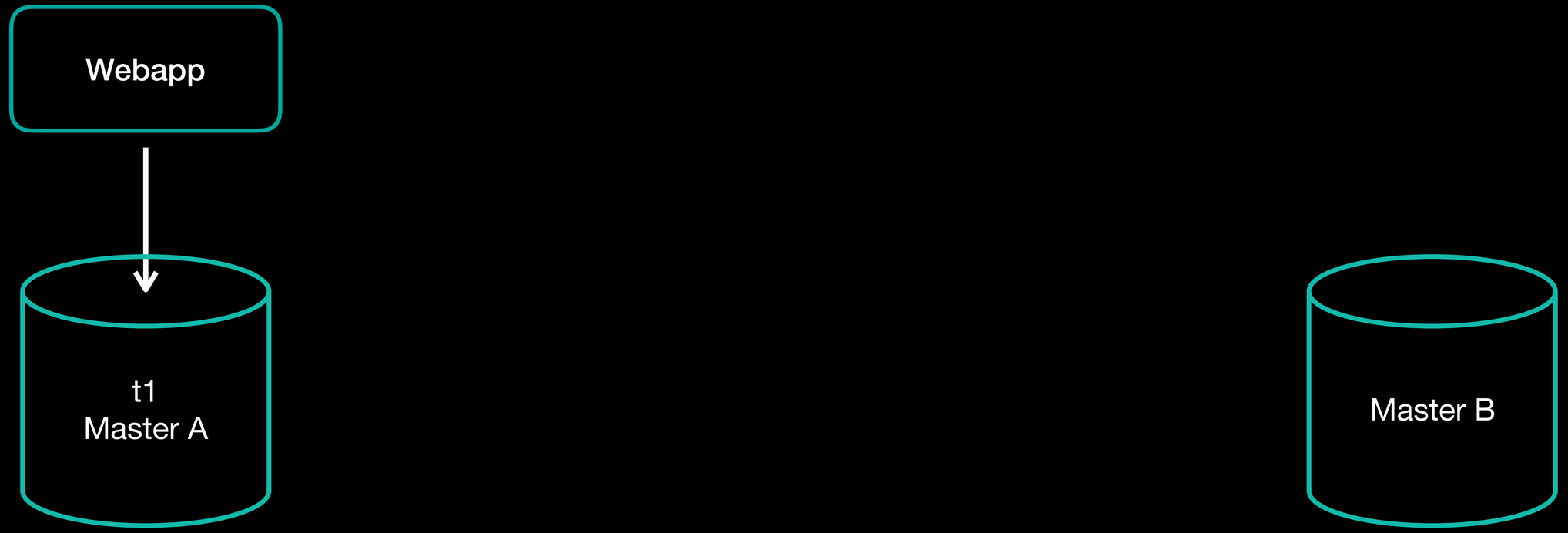
Typical Shard Replication



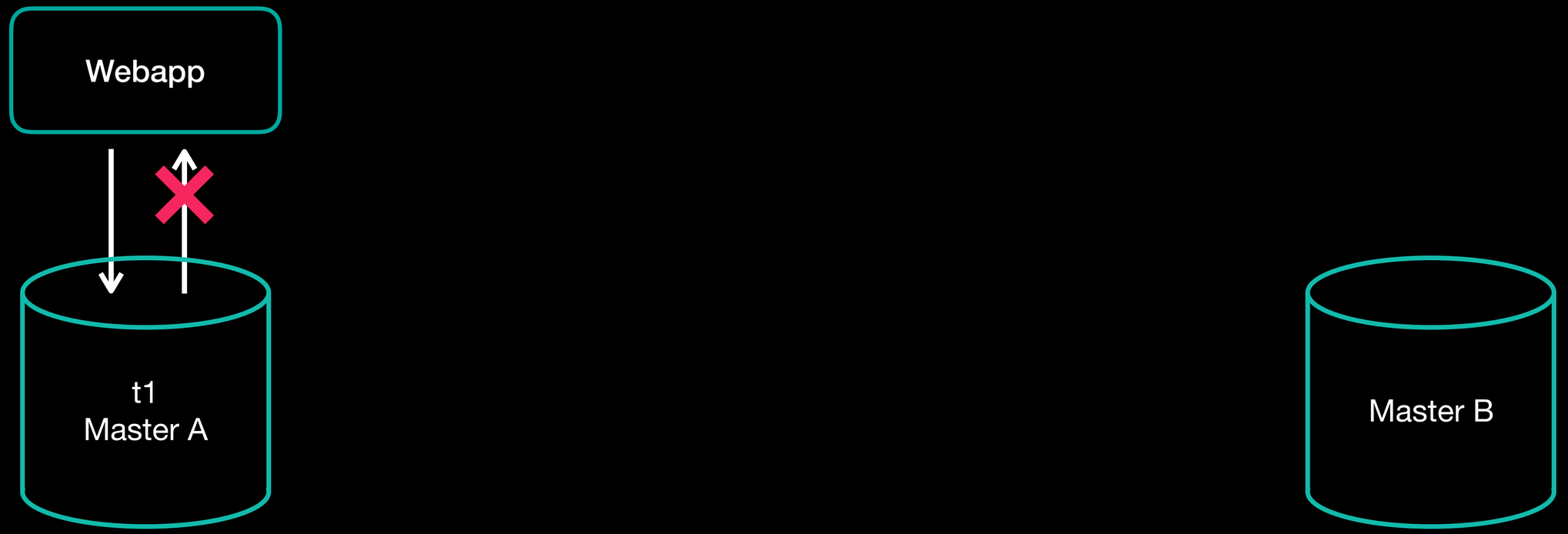
Slack shard replication
Active master-master



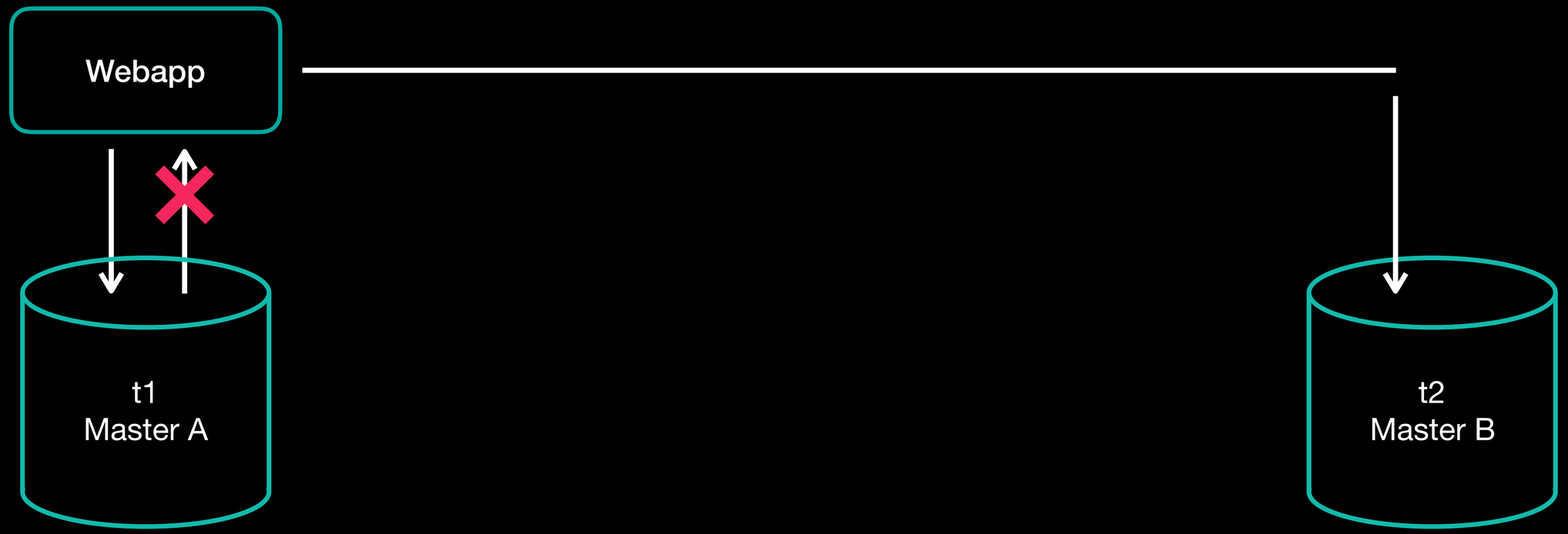
Slack shard replication
Active master-master



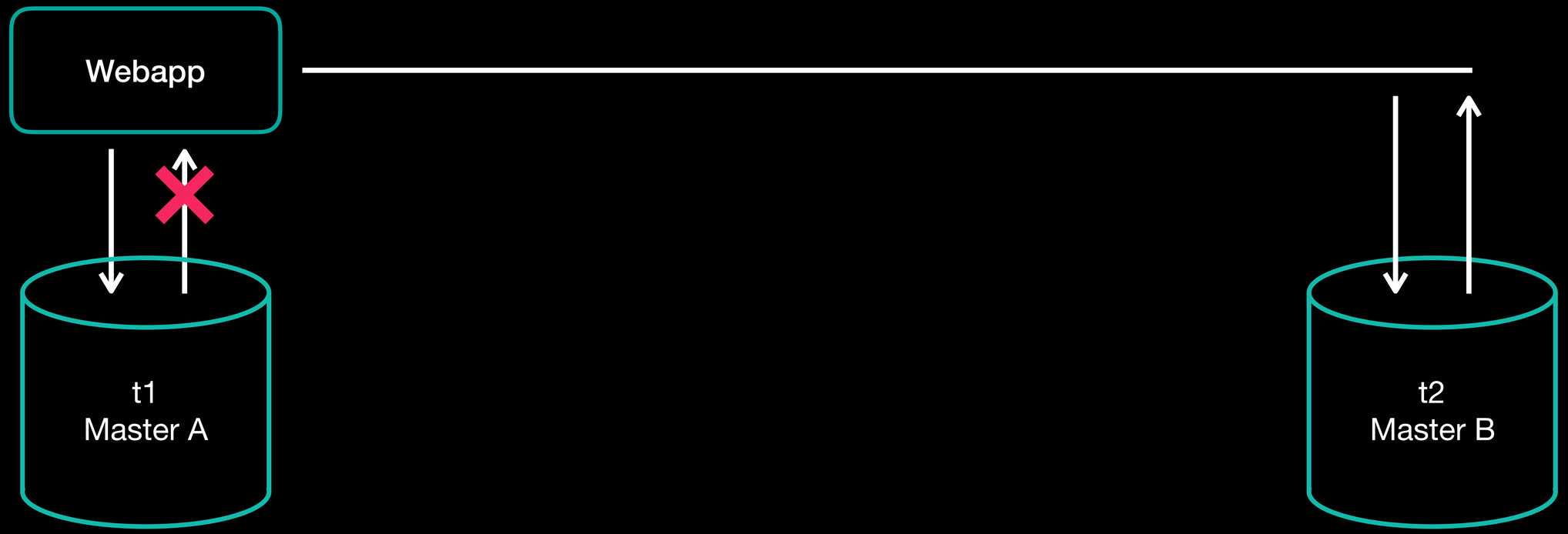
Slack shard replication
Active master-master



Slack shard replication
Active master-master

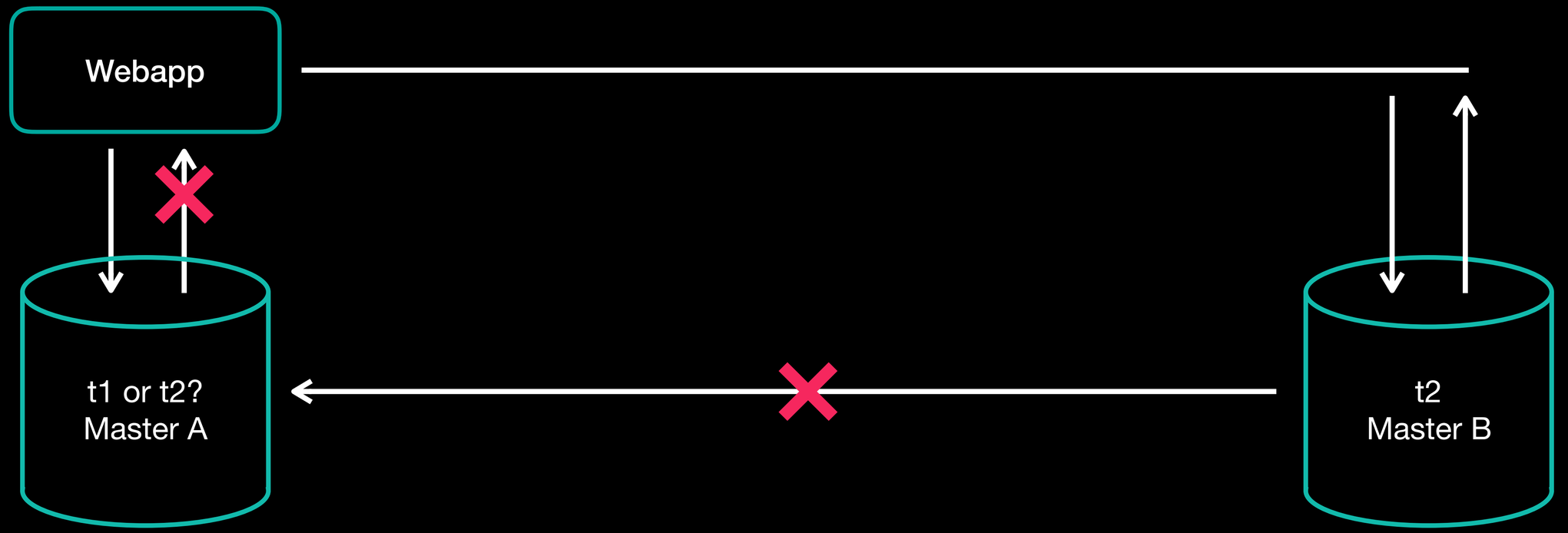


Slack shard replication
Active master-master



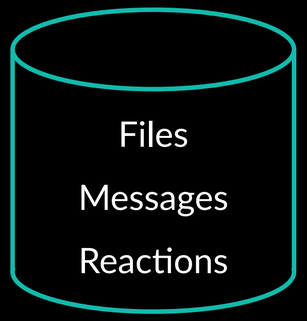
Slack shard replication
Active master-master

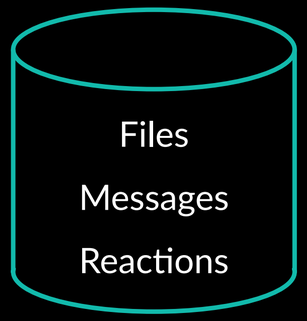
Slack shard replication
Active master-master



Types of database clusters

Shards





Mains

Aux

Problems with Slack's shards

1. Static sharding scheme
2. Inefficient usage
3. Operational overhead
4. Tight coupling between infrastructure and app

1
Static sharding scheme

Slack's database sharding scheme is only capable of sharding by teams. A team cannot grow beyond a MySQL shard.

1
Static sharding scheme

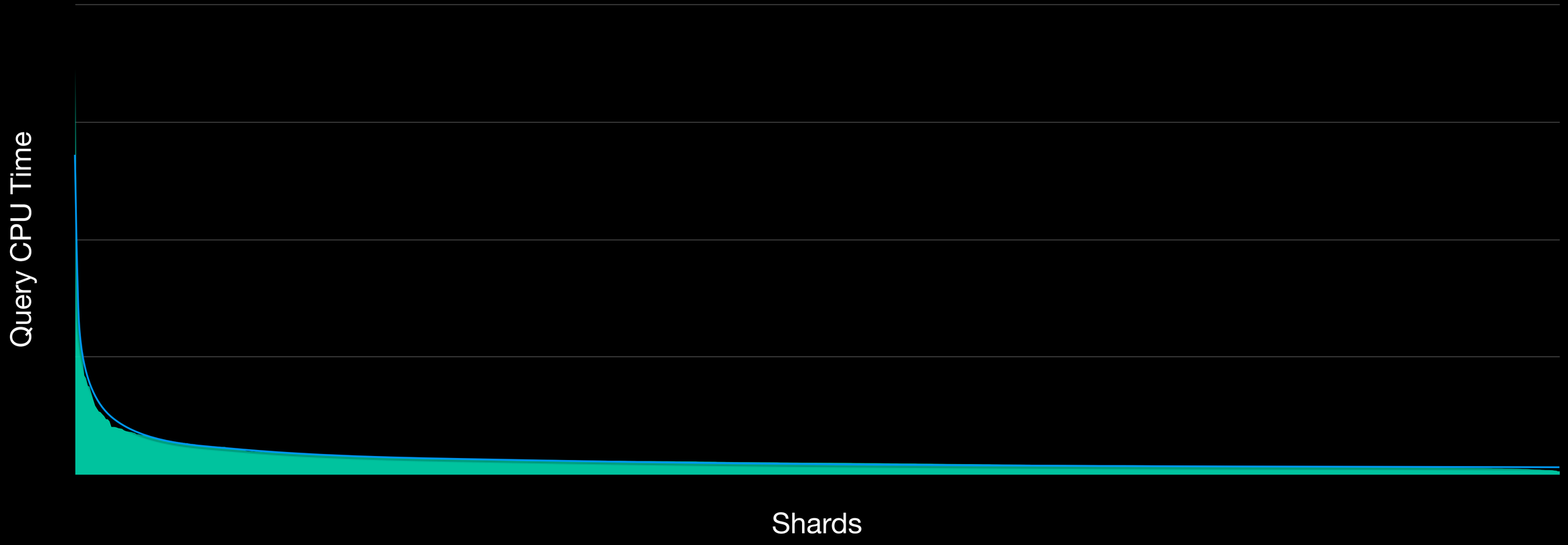
Slack's database sharding scheme is only capable of sharding by teams. A team cannot grow beyond a MySQL shard.

Requirement: We need a flexible sharding scheme

2
Inefficient usage

Not all shards are equal. This results in hot spots and inefficient usage of resources across the cluster.

Cluster Utilization



2
Inefficient usage

Not all shards are equal. This results in hot spots and inefficient usage of resources across the cluster.

2
Inefficient usage

Not all shards are equal. This results in hot spots and inefficient usage of resources across the cluster.

Requirement: We need to improve cluster utilization and avoid hot spots.

3
Operational overhead

Active master-master setup is an anti-pattern which can result in duplicate primary key errors. We cannot use things like row-based replication, global transaction identifier (GTID), orchestrator, gh-ost schema management, pt-online-schema-change, etc.

3
Operational overhead

Active master-master setup is an anti-pattern which can result in duplicate primary key errors. We cannot use things like row-based replication, global transaction identifier (GTID), orchestrator, gh-ost schema managedment, pt-online-schema-change, etc.

Requirement: The next database, at Slack, should be easy to operate.

4
Tight coupling between infrastructure and app

Application developers need to worry about databases physical layout. It means adding/removing/replacing shards require a production deploy.

4
Tight coupling between infrastructure and app

Application developers need to worry about databases physical layout. It means adding/removing/replacing shards require a production deploy.

Requirement: We need to make sharding transparent to the application.

Problems with Slack's shards

1. Static sharding scheme
2. Inefficient usage
3. Operational overhead
4. Tight coupling between infrastructure and app

Enter Vitess

What is Vitess?

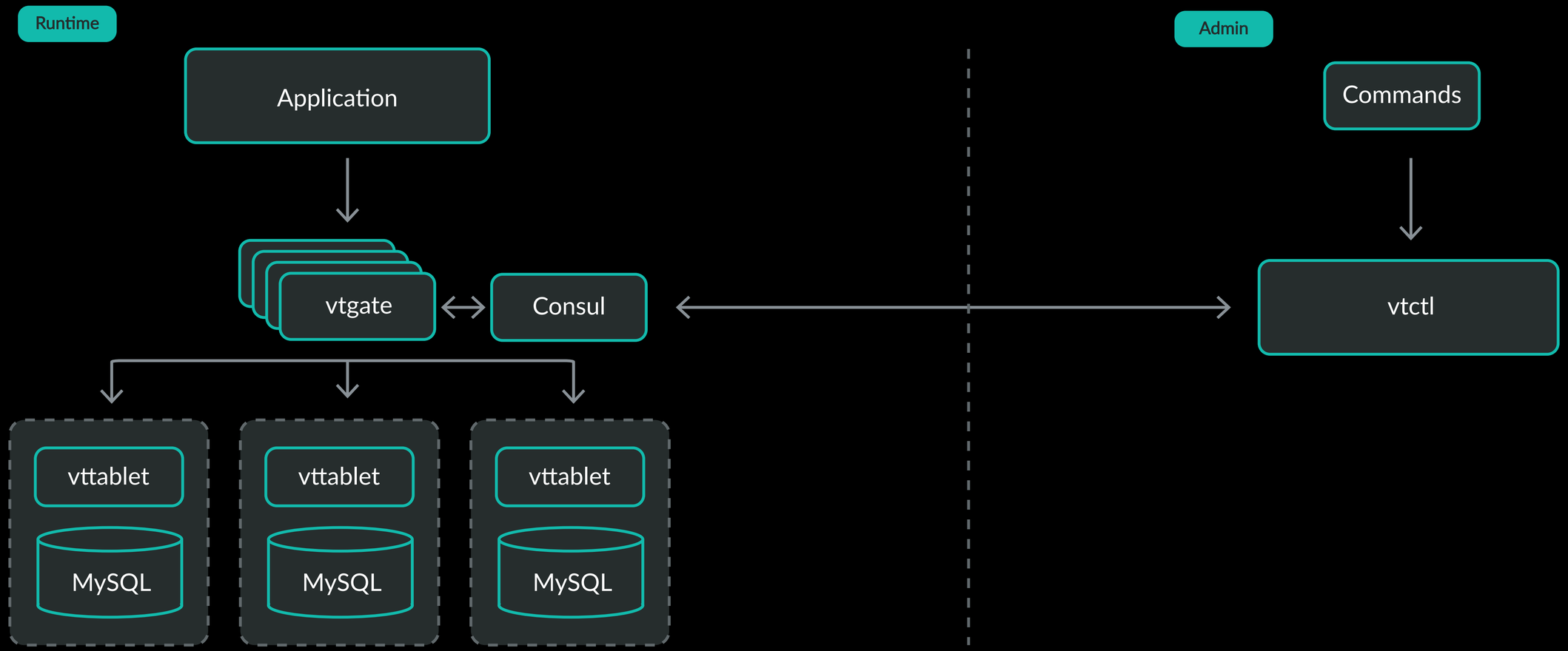
1. Database solution for MySQL
Deploy, scale and manage large MySQL cluster

2. Built on top of MySQL replication and InnoDB
MySQL features + scalability of a NoSQL database

3. Open source project by YouTube (Google)
Started in 2010

4. Cloud Native Computing Foundation endorsed project
Ability to run each component in a container

Vitess components



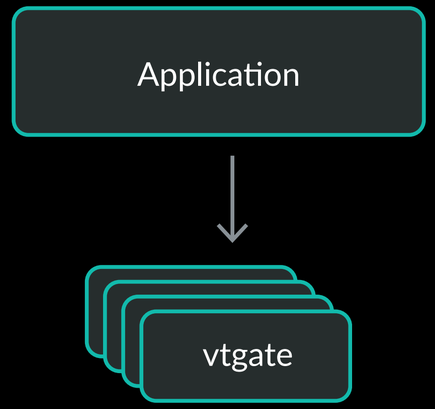
Vitess components

Runtime

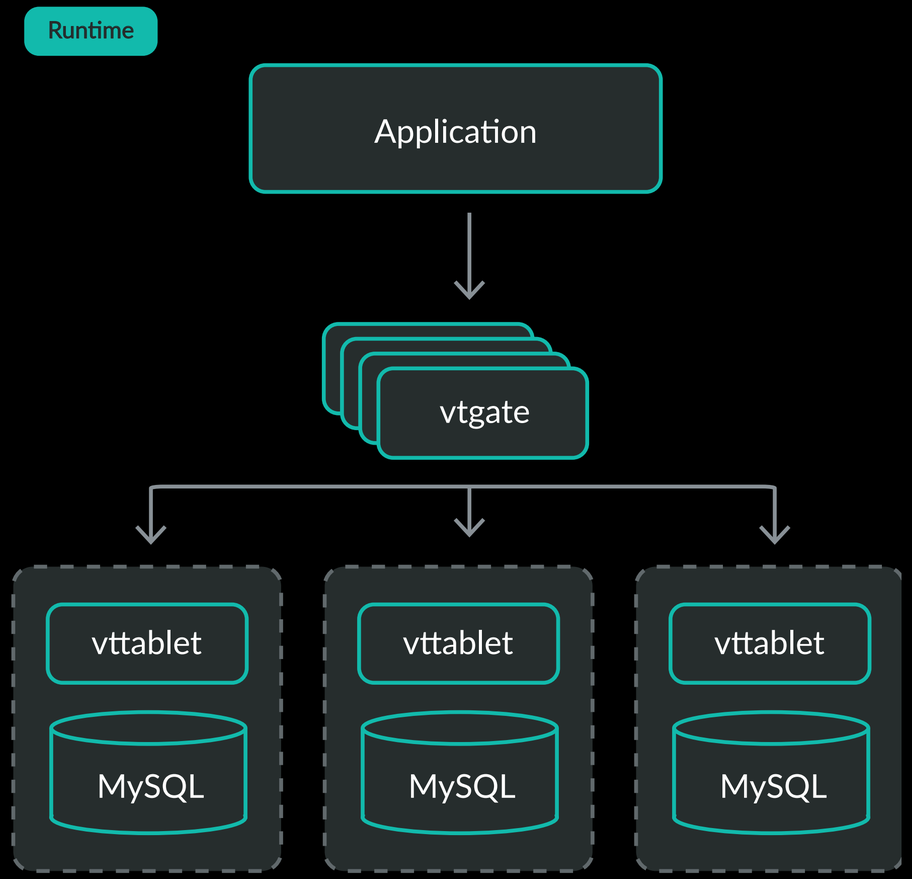
Application

Vitess components

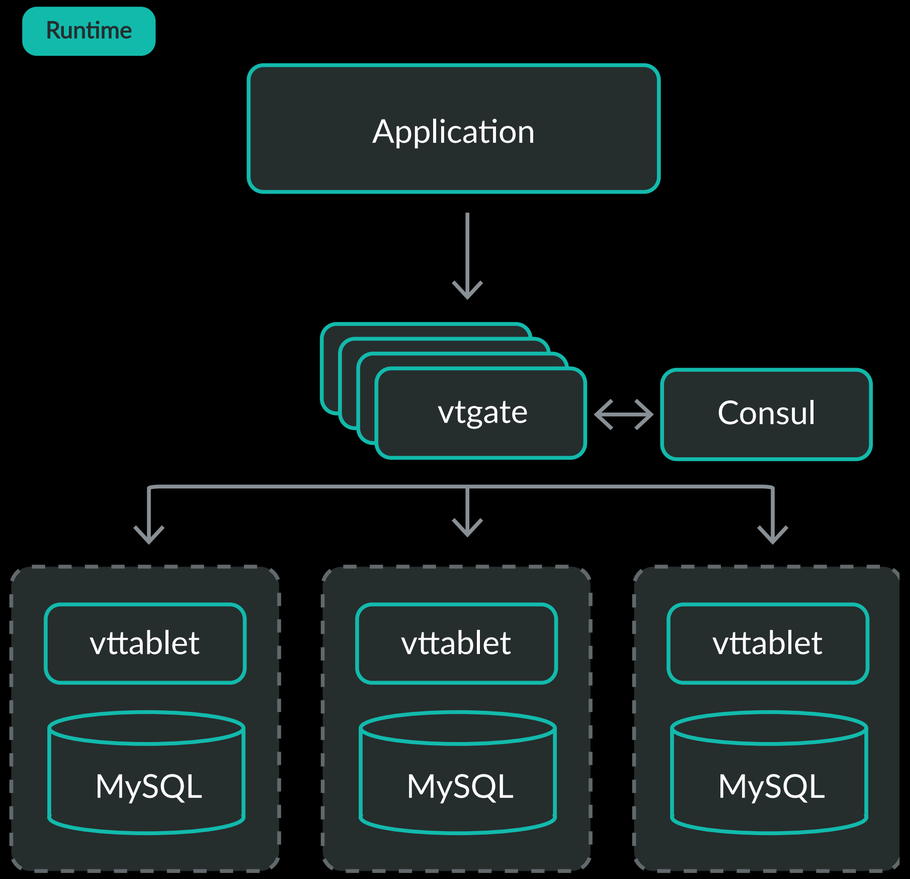
Runtime



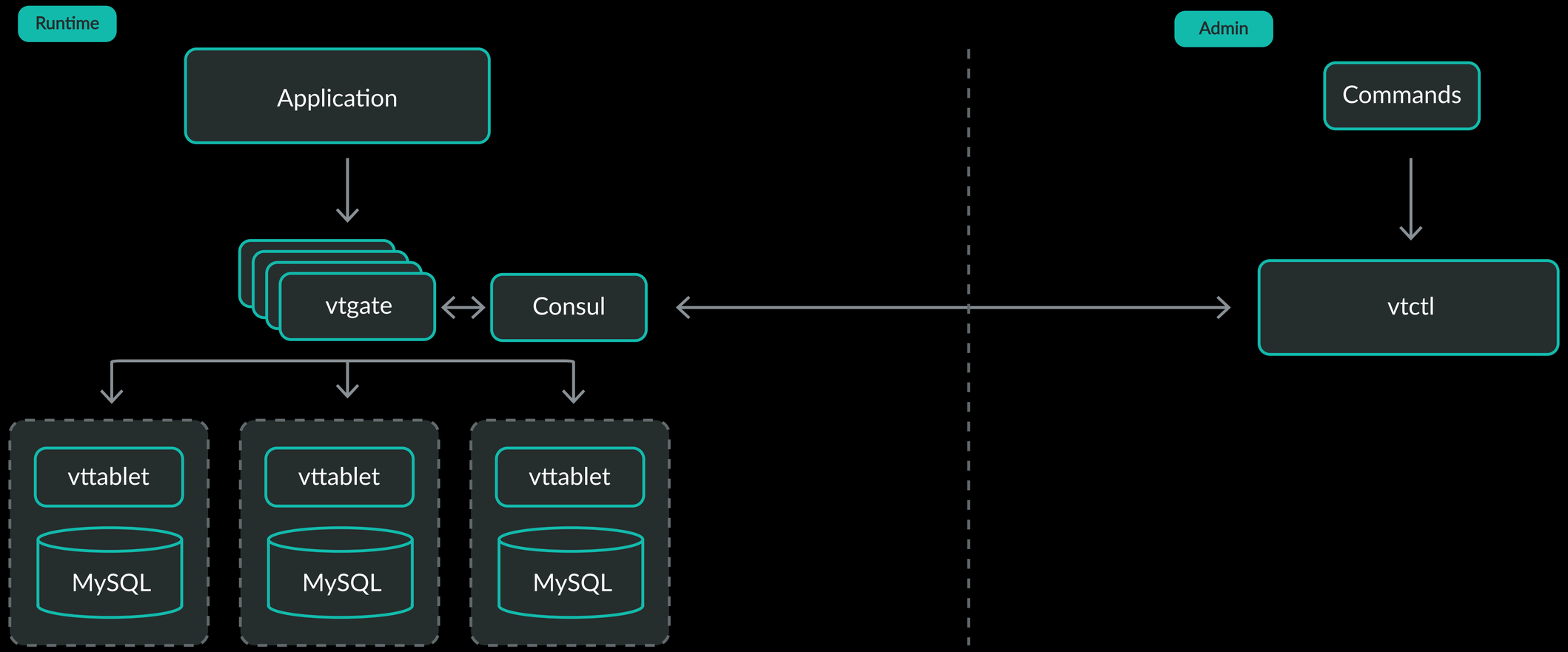
Vitess components



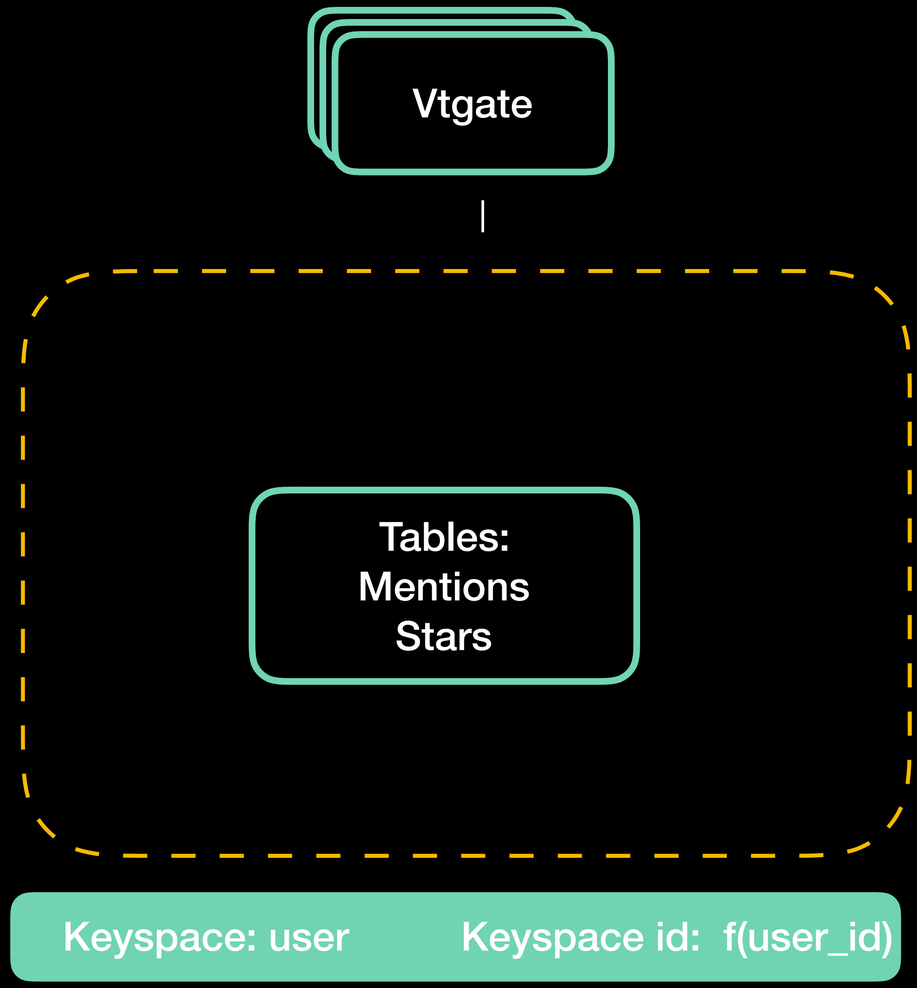
Vitess components



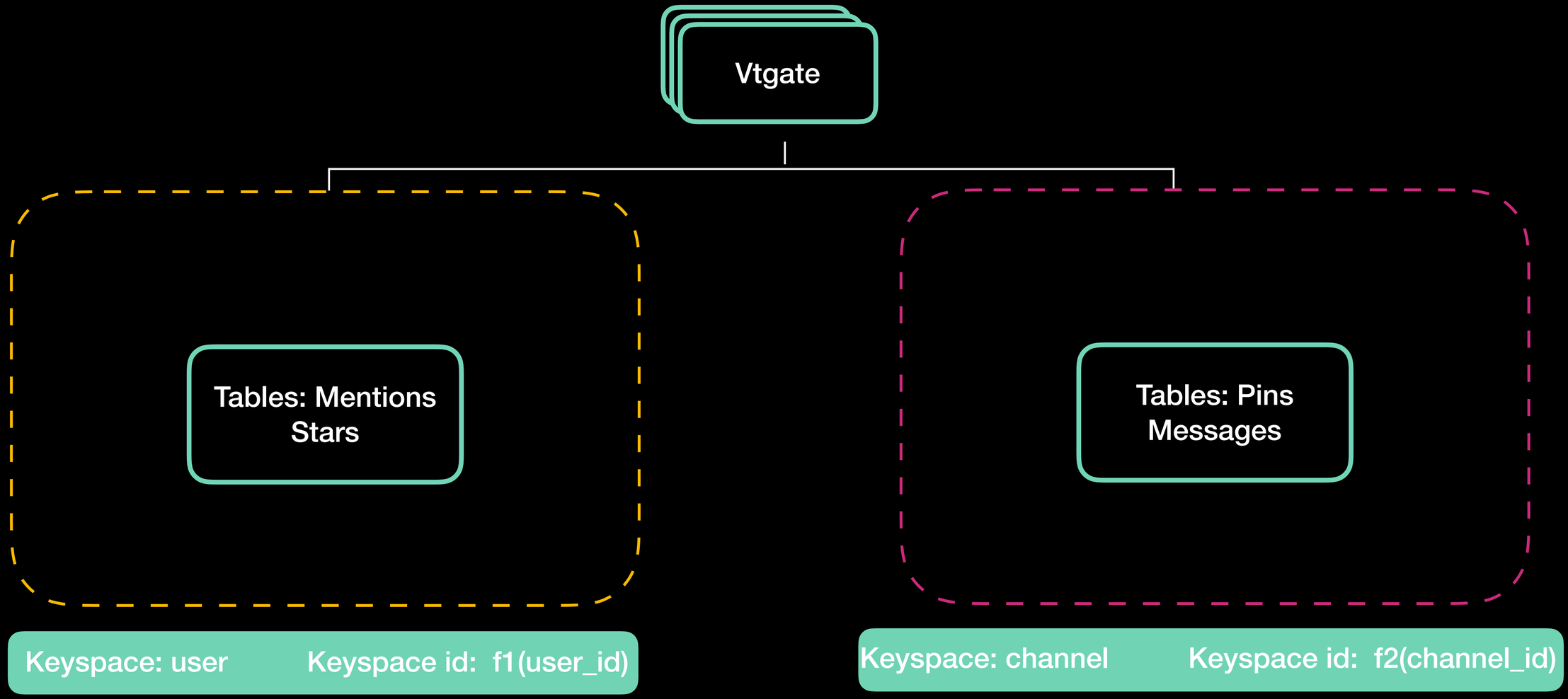
Vitess components



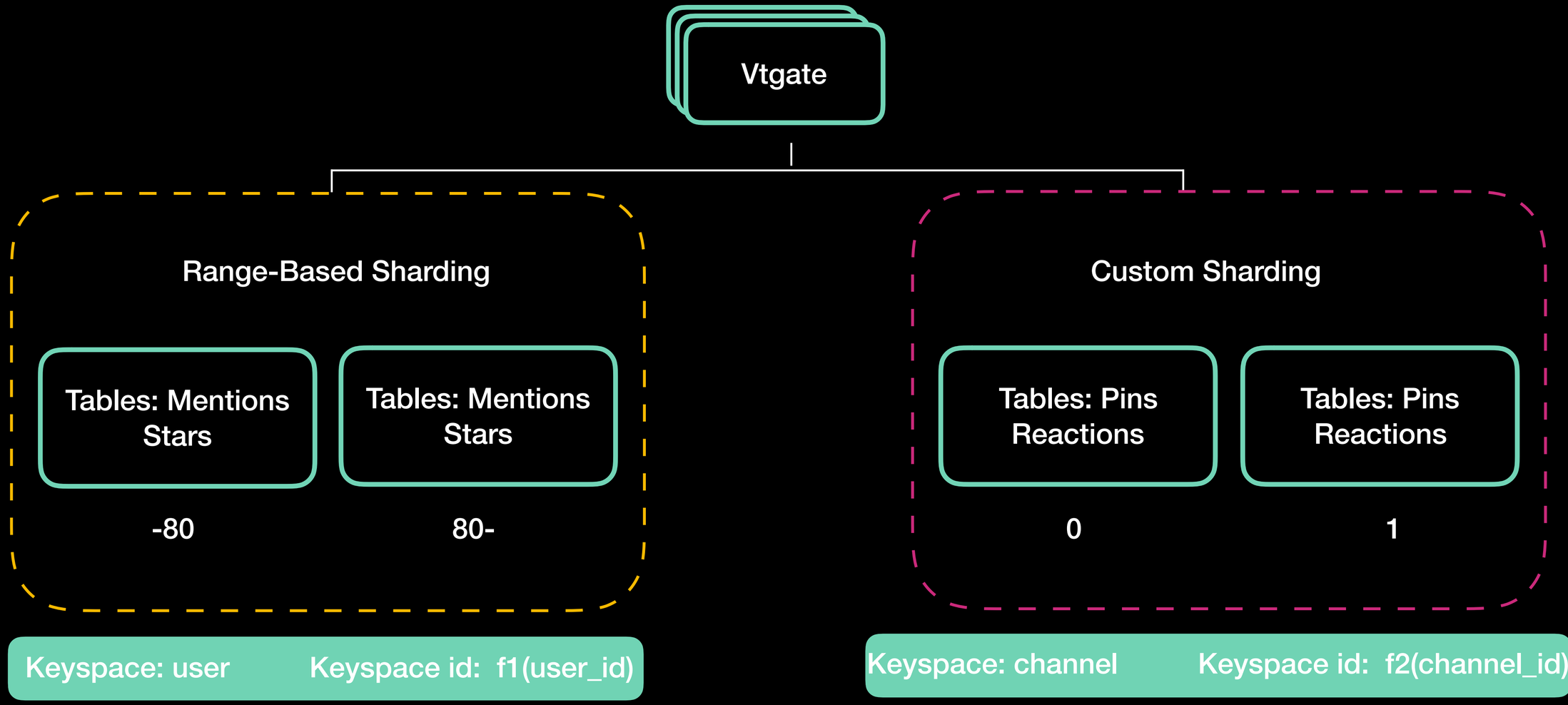
Vitess Keyspaces



Vitess Keyspaces



Vitess Sharding



Why Vitess?

Built on top of MySQL

Flexible sharding

Easy development model

Separation of code and infrastructure

Out of the box operational tools

Powers YouTube storage

Performance

Active friendly community

Slack's contribution to Vitess project

- vtexplain
- gRPC Auth Plugins
- Prometheus monitoring support
- Query logging improvements
- Vtqueryserver (experimental)
- Resilient topology caching
- Multi-shard delete and updates statements
- MySQL query compatibility
- And many more...

Vitess at Slack

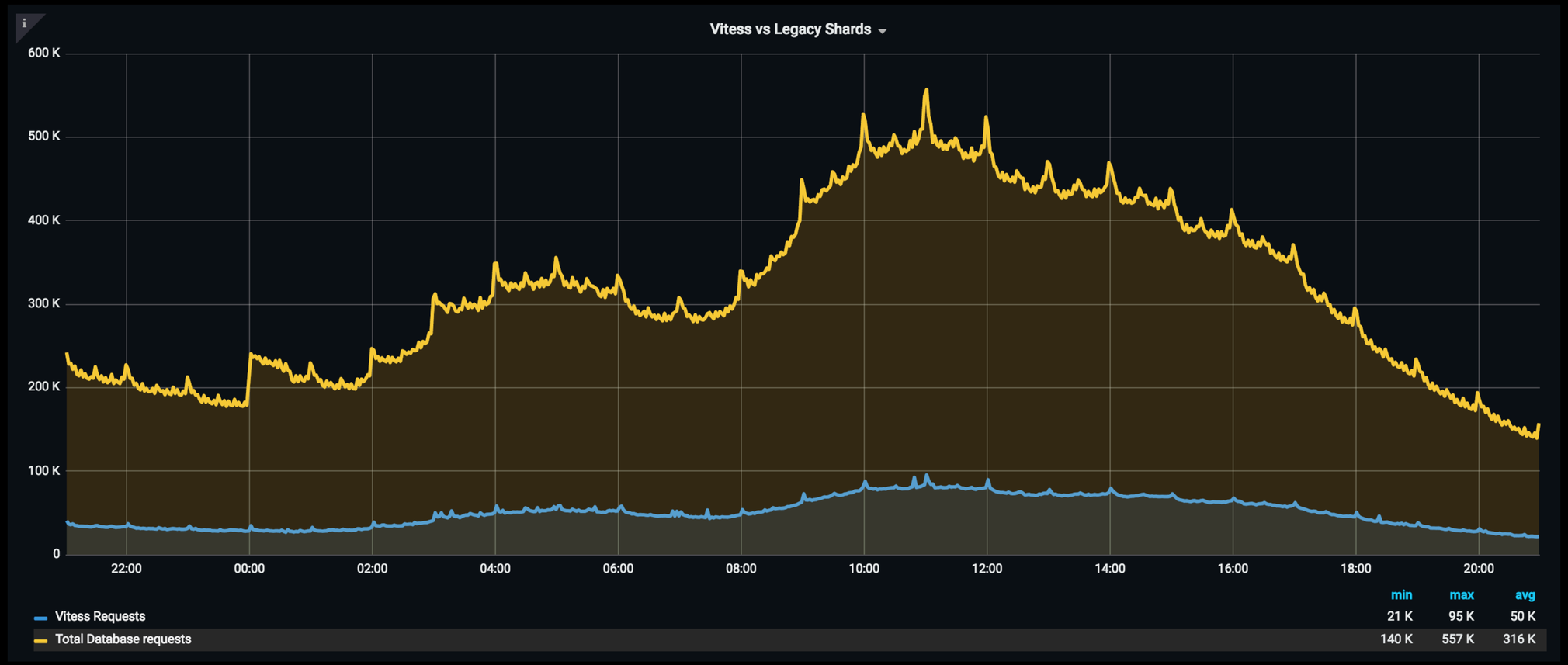
As of Oct 2018

- Tables moved - **teams**, teams_preference, mentions, sessions, rss_feeds, audit logs, stars, payments
- **117** shards across **9** keyspaces

- MySQL replication using **row based replication** and **semi-sync, GTIDs**
- Automated failover using github.com/github/orchestrator
- Schema changes - github.com/github/gh-ost

Vitess at Slack

As of Oct 2018



Future of Vitess at Slack

- Migrations - channels, stars, pins, files, messages?
- Setting up regression testing environment
- Vtsg - auto scaling/ replacement

- Point in time recovery
- Partial results scatter query
- Adopt VTReplication

Thank you

Join #Vitess on Slack
Visit https://vitess.io/ for more information
Slack is hiring: https://slack.com/careers