

Scaling resilient systems: a journey into Slack's database service

Rafael Chacón
Guido Iaquinti



SPEAKERS



Rafael Chacon

he/him/his

 twitter.com/rafaelchacon

Staff Software Engineer - Slack



Guido Iaquinti

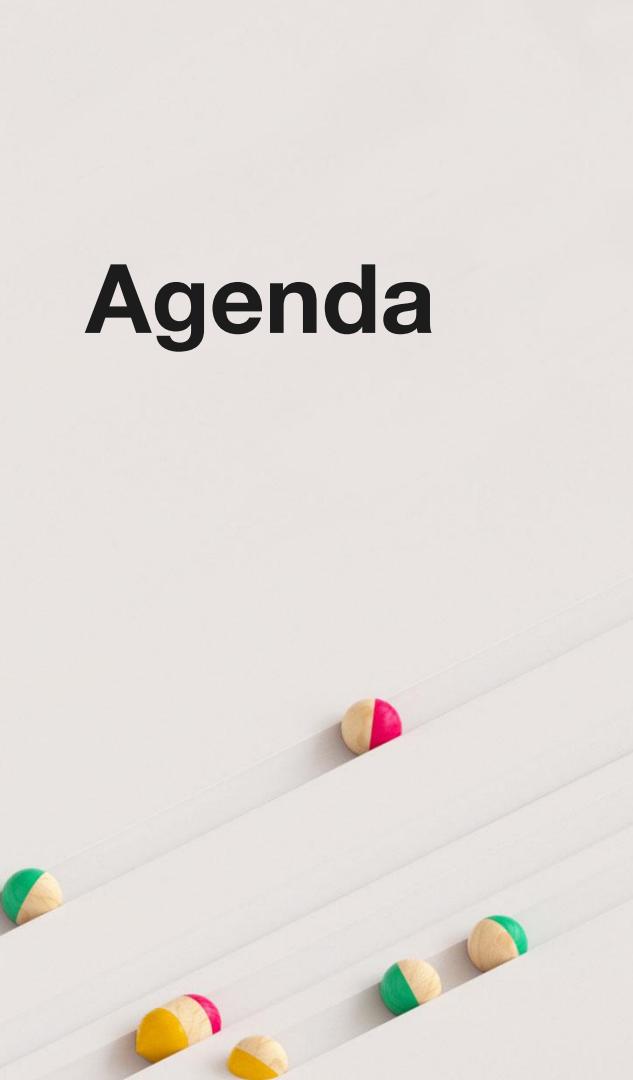
he/him/his

 twitter.com/guidoiaquinti

 guido.iaquinti.com

Site Reliability Engineer - Freelance

Agenda

- 
- A decorative background image featuring several small, colorful wooden beads (yellow, pink, green) arranged on a light-colored, ribbed surface, possibly a book or a folder.
- 1. Databases at Slack
 - 2. Running databases in the cloud
 - 3. Fault tolerance & Isolation
 - 4. Key Lessons
 - 5. Q&A

MISSION STATEMENT

Slack's mission is to make people's working lives **simpler**, more **pleasant**, and more **productive**.

The screenshot shows the Slack desktop application interface. On the left is the sidebar with the team name "Acme Inc." and a user profile for "Alex Saunders". The sidebar lists "All Unreads", "Starred" channels (# announcements, # design-team, # social-media, # helpdesk), and "Channels" including # accounting, # design-crit, # help-design, # media-and-pr, # triage-issues, and # design-team-sf. Below that is the "Direct Messages" section with entries for slackbot, Zoe Maxwell, Leland..., Florence Garret, and Liza Zhang.

The main area displays the "#social-media" channel feed. The channel header includes a star icon, 21 messages, 1 thread, and a description "Track and coordinate social med...". It shows messages from Sara Parker, Zoe Maxwell, Acme Team BOT, Harry Boone, and Jeremy Stevens. A message from Jeremy Stevens is highlighted, showing a file attachment for "1/9 Meeting Notes" which was last edited just now. A message input field at the bottom is ready for the user to type a message to the channel.

On the right, there is a sidebar titled "About #social-media" with links to "Channel Details", "Highlights", "1 Pinned Item", "21 Members", "Shared Files", and "Notification Preferences".

Databases at Slack

Current status

In progress
migration of our
entire dataset to
Vitess.

Two main types of clusters:

- Legacy shards
- Vitess shards

Why are we migrating?

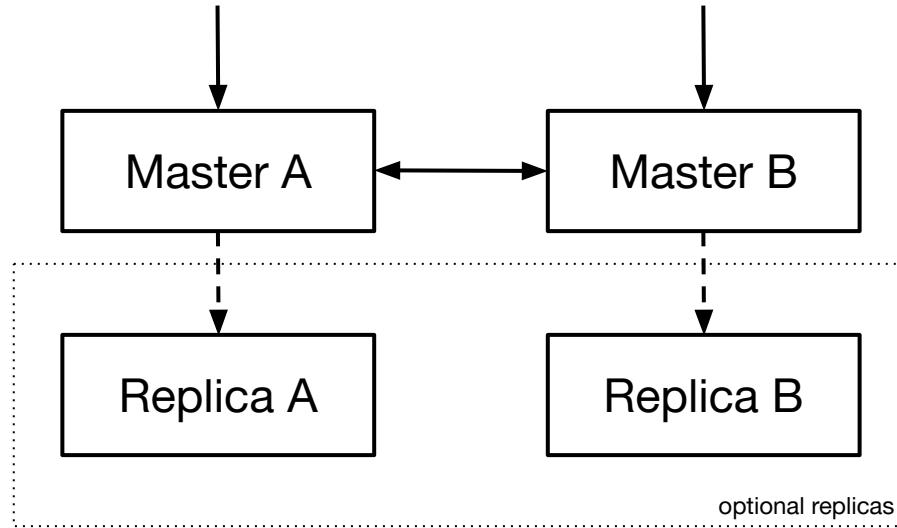
For more details
please see the
presentations on
the side.

tl;dr; shard size limits, inefficient resource distribution, operational overhead, single sharding model

- “*Migrating to Vitess at (Slack) Scale*” - Mike Demmer
- “*Designing and launching the next-generation database system at Slack: from whiteboard to production*” - Guido Iaquinti
- “*Smooth scaling: Slack’s journey toward a new database*” - Ameet Kotian

Legacy shards

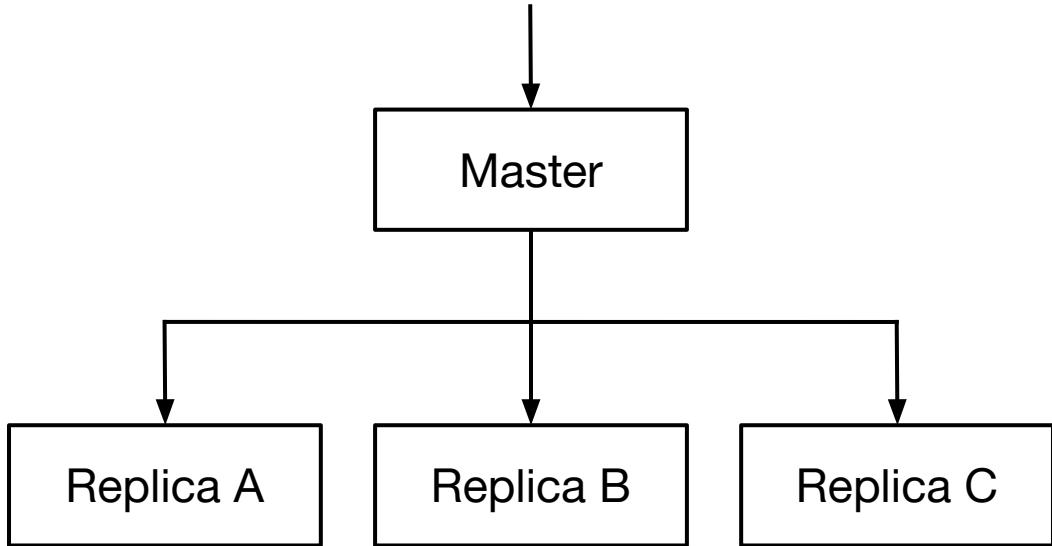
Application level
team-sharded **active master-master**
MySQL setup.



Vitess shards

Master-replica

MySQL setup fully managed by Vitess.



Stats

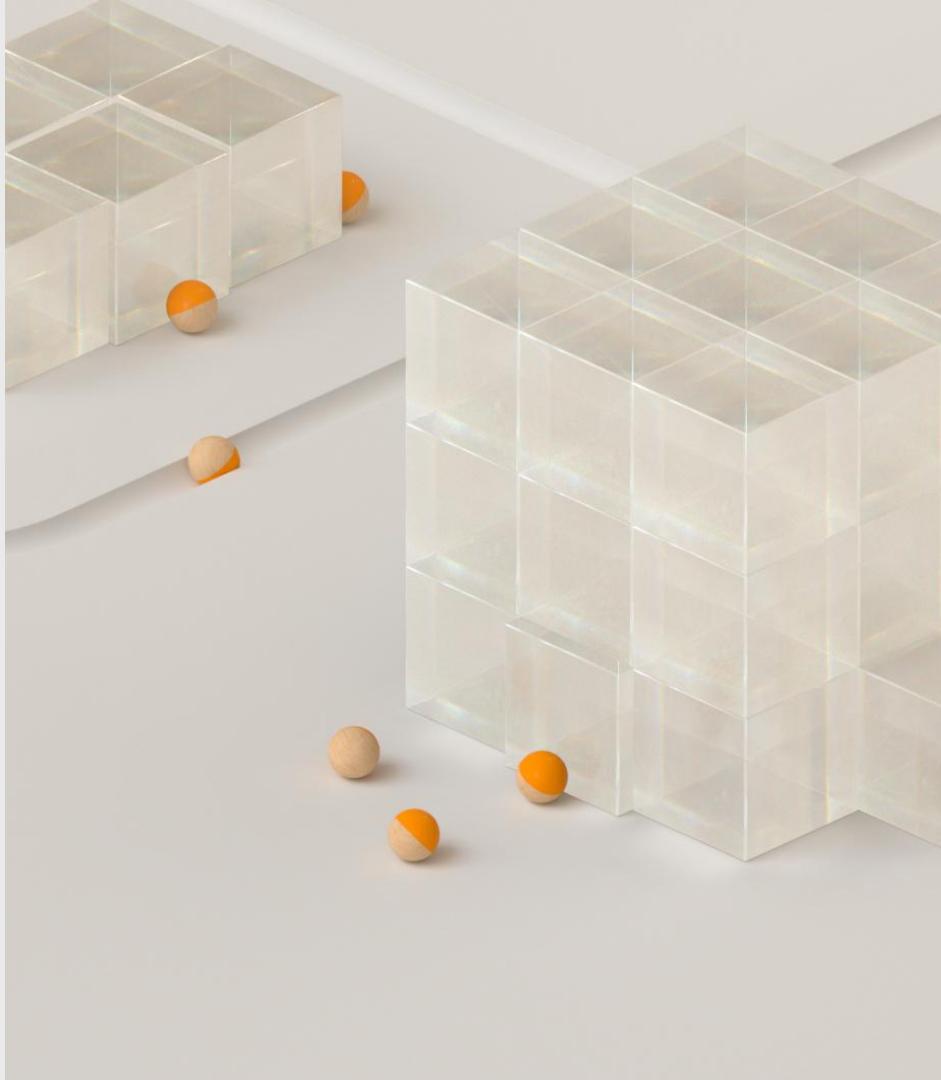
Queries per day: **53+ billion**

Storage provisioned: **7.5+ PB**

Served by legacy infrastructure: ~**60%**

Served by Vitess: ~**40%**

Target: **70%** served by Vitess by EOY



Running databases in the cloud



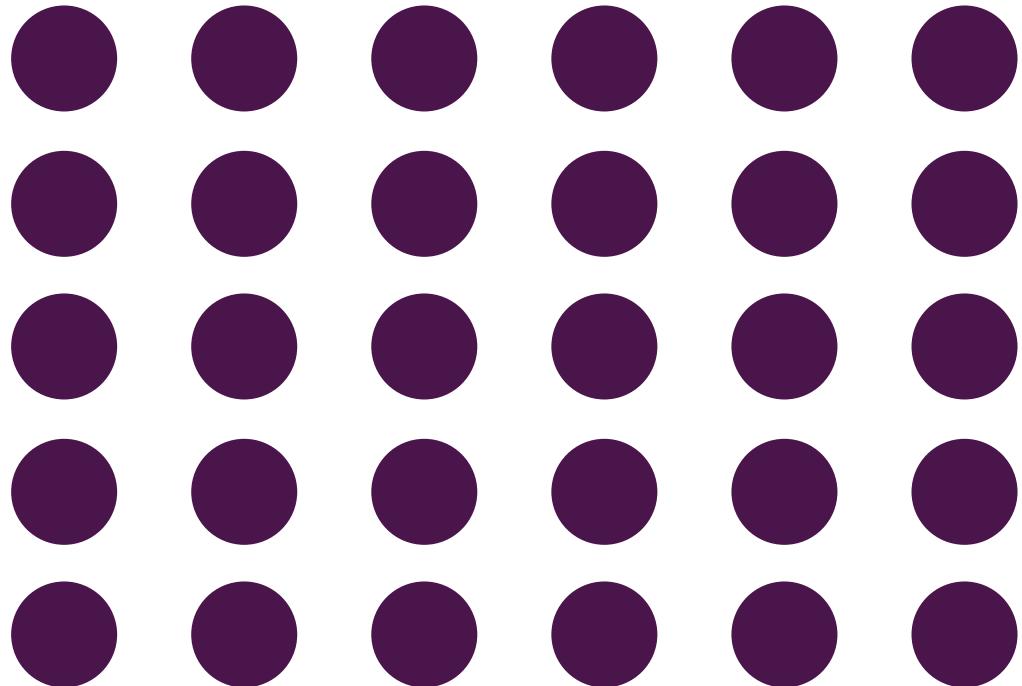
Variable infrastructure



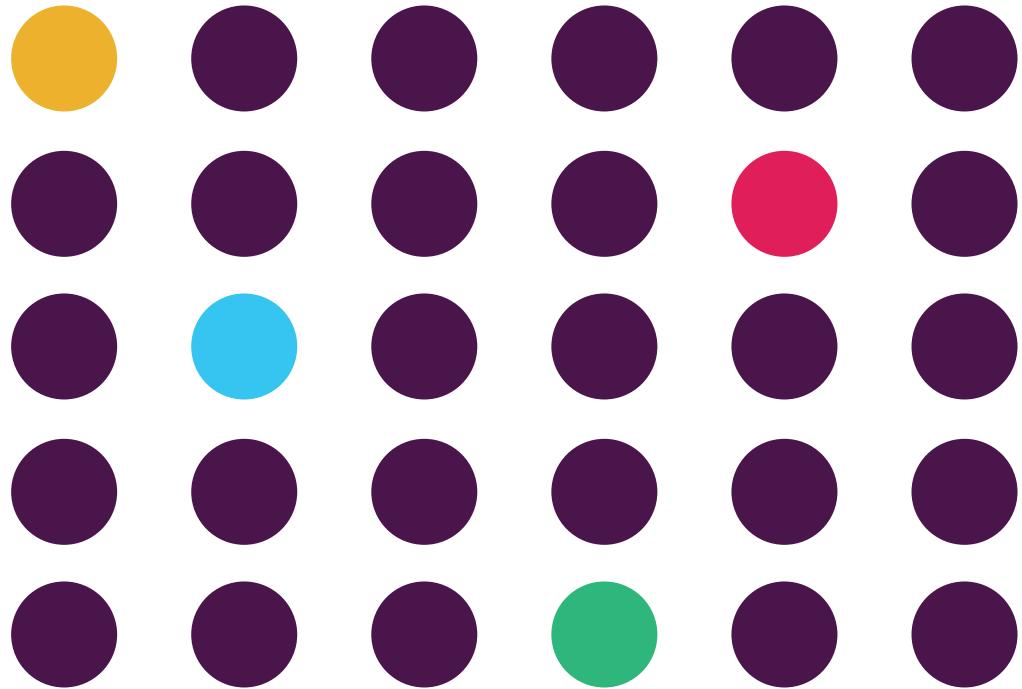
Variable infrastructure



Variable infrastructure



Variable infrastructure



Immutable infrastructure

- Instances are untouched after provisioning
- Configuration changes happen only through reprovisioning
- No in-place patching allowed



Instance failure

The airplane analogy



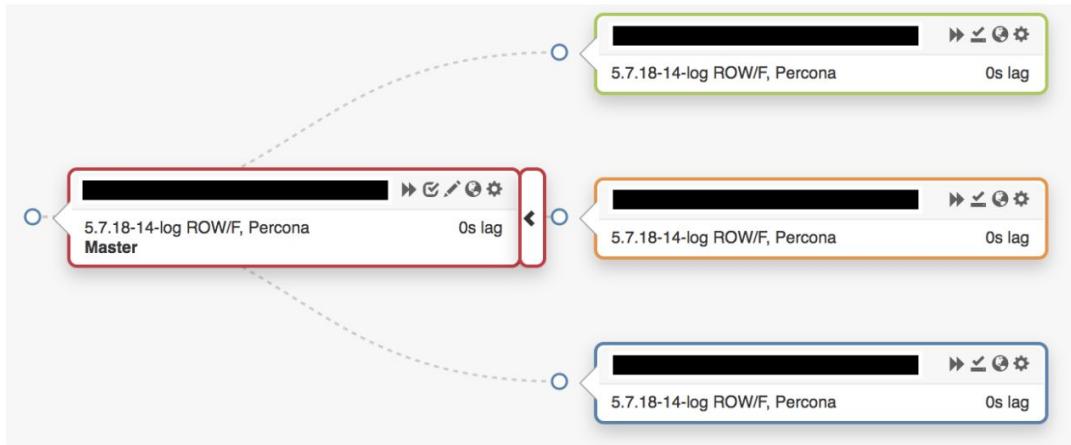
Instance failure

Always reprovision: challenges

- Network storage VS ephemeral instance storage
 - Network storage: stop & start instance
 - Instance storage: download the latest backup (NIC is the bottleneck)
- Small shards VS big shards
 - Recovery time (if you don't use network storage)
 - Blast radius
 - Distributed workload VS centralized workload
 - Less contention

Durability through replication

via semi-sync



Durability through replication

via semi-sync

```
rpl_semi_sync_master_timeout = 999999999999
```

```
rpl_semi_sync_master_wait_no_slave = 1
```

```
sync_binlog = OFF
```

```
innodb_flush_log_at_trx_commit = 2
```

How we run Vitess

- AWS



How we run Vitess

- AWS
- EC2 not k8s



Kelsey Hightower

@kelseyhightower

Follow

Kubernetes has made huge improvements in the ability to run stateful workloads including databases and message queues, but I still prefer not to run them on Kubernetes.

6:04 AM - 13 Feb 2018

304 Retweets 705 Likes



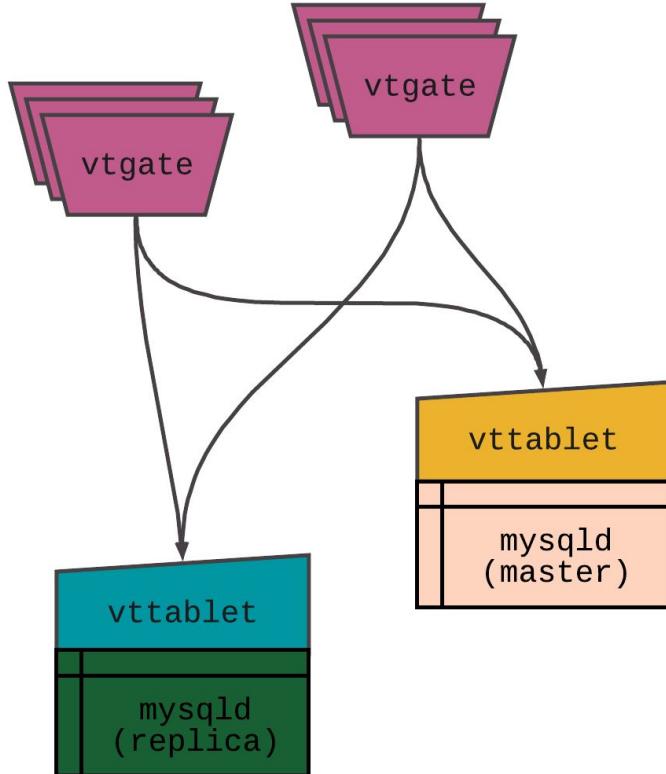
24

304

705

How we run Vitess

- AWS
- EC2 not k8s
- ASG for stateless components



How we run Vitess

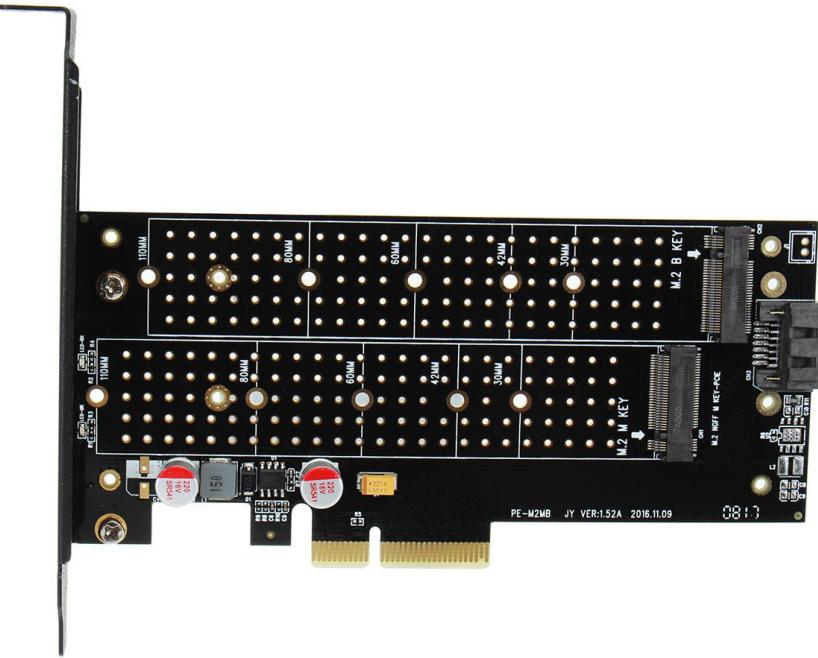
- AWS
- EC2 not k8s
- ASG for stateless components
- MySQL 5.7 (Percona)



PERCONA

How we run Vitess

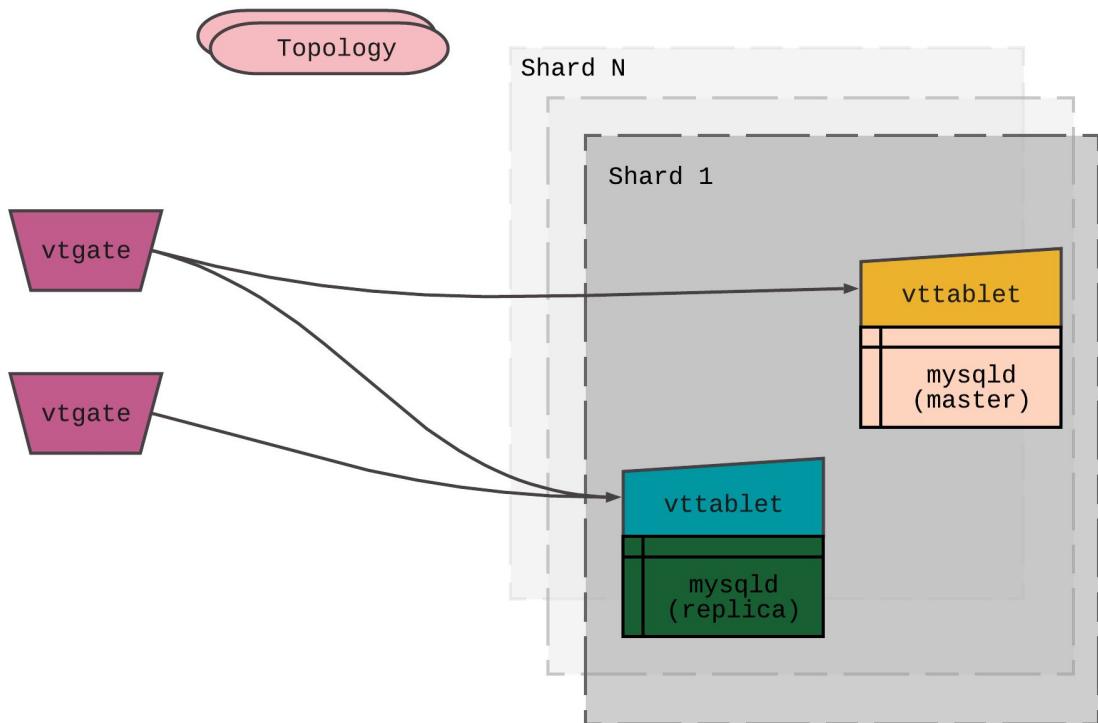
- AWS
- EC2 not k8s
- ASG for stateless components
- MySQL 5.7 (Percona)
- Ephemeral NVMe (no EBS)



Fault tolerance & isolation

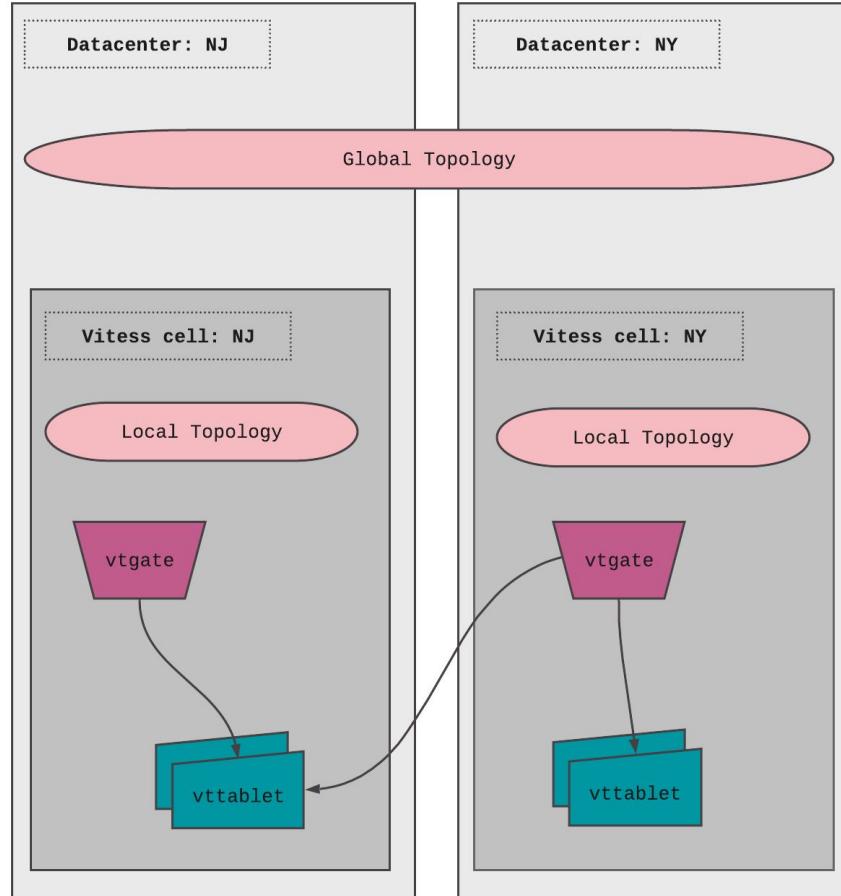


Vitess architecture



FAULT TOLERANCE & ISOLATION

Vitess standard deployment



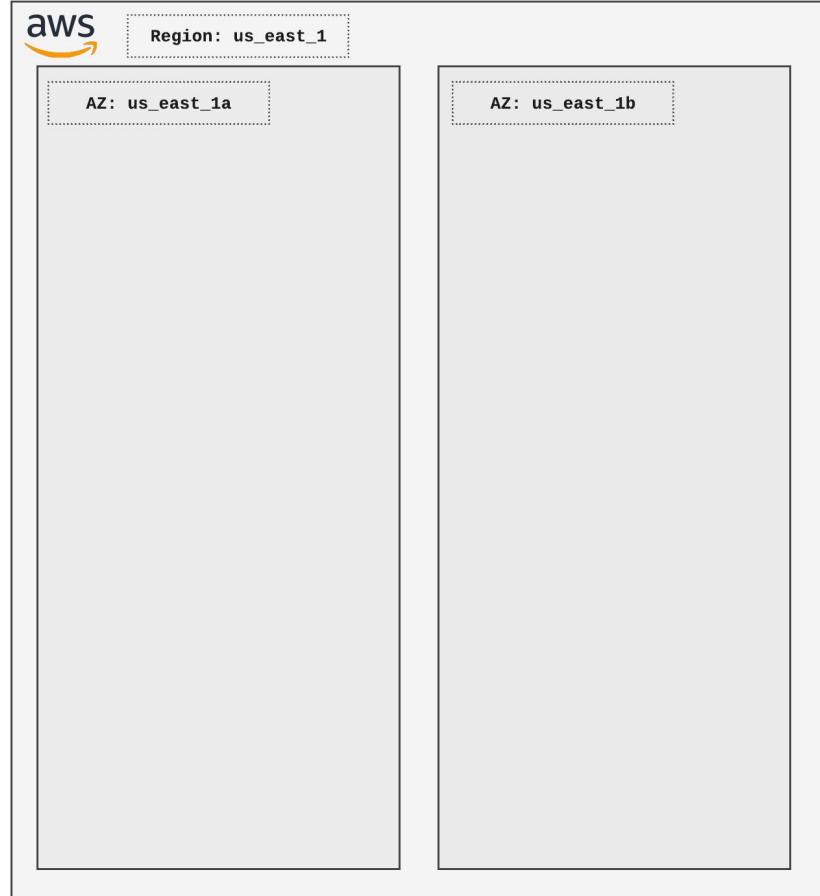
Slack cloud infrastructure

- Amazon EC2 is hosted in multiple locations world-wide.
- These locations are composed of Regions and Availability Zones (AZ's).
- Each Region is a separate geographic area.
- AZ's in a Region are connected through low-latency links.



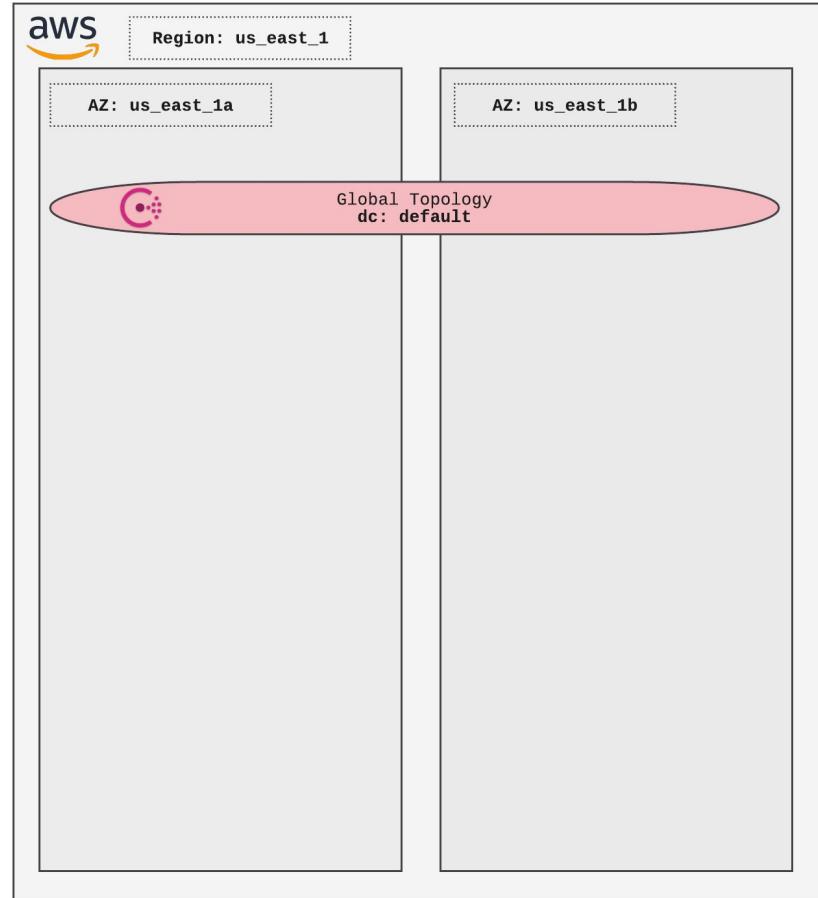
Vitess initial deployment

- We now have multiple clusters in different regions.



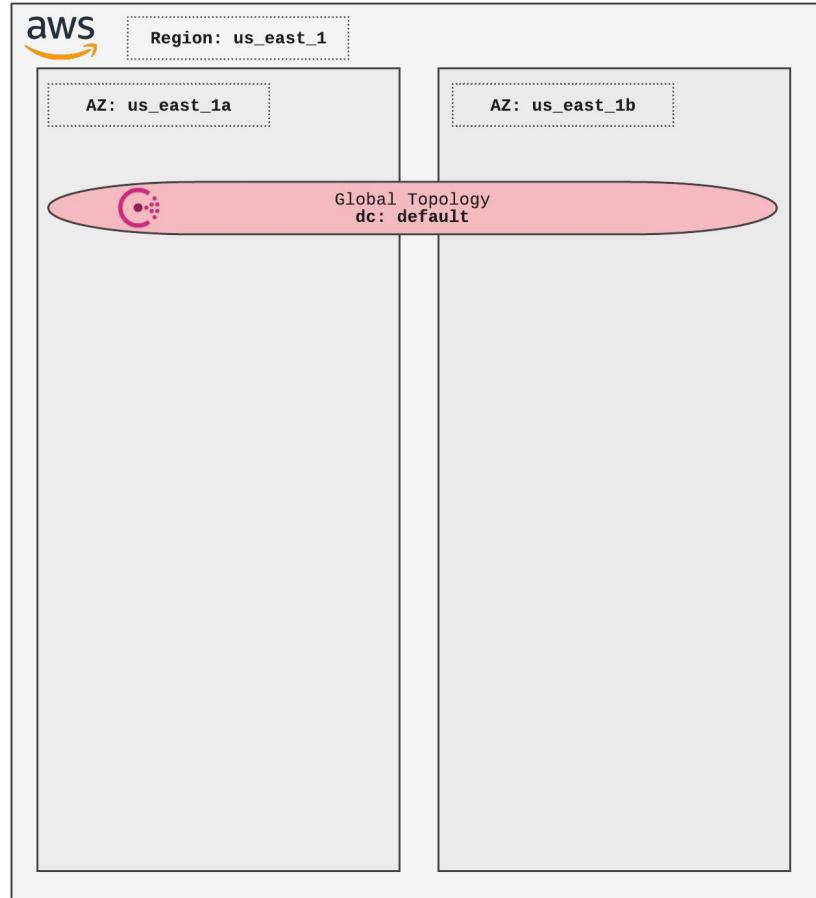
Vitess initial deployment

- We use Consul



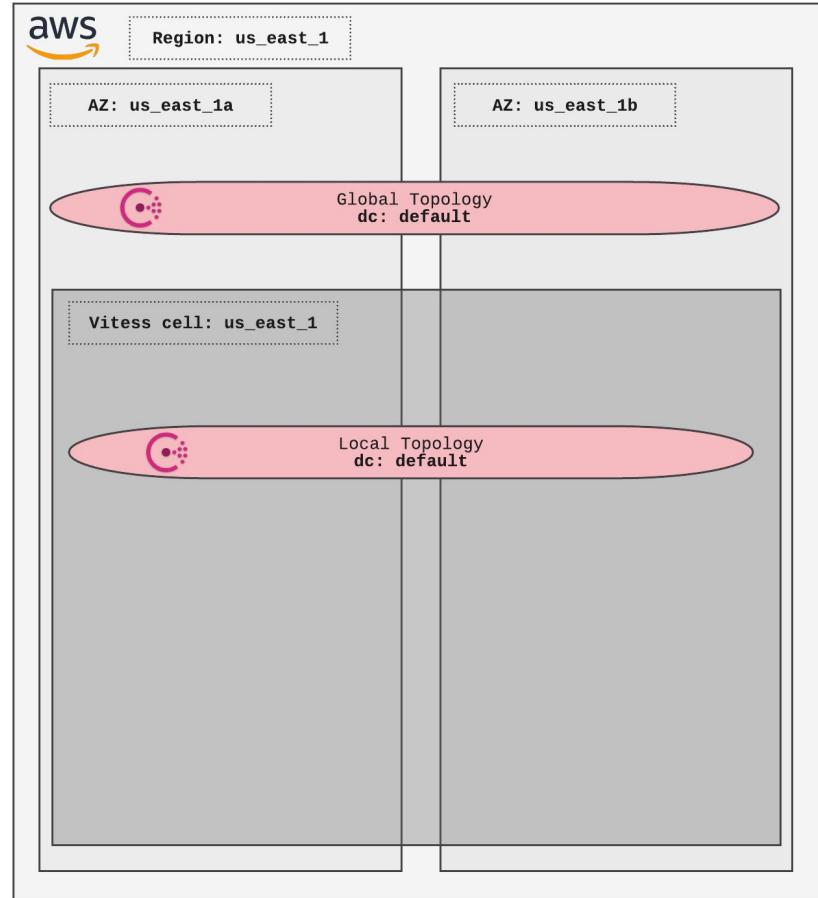
Vitess initial deployment

- We use Consul
- Notice **default** datacenter (dc) in Consul.



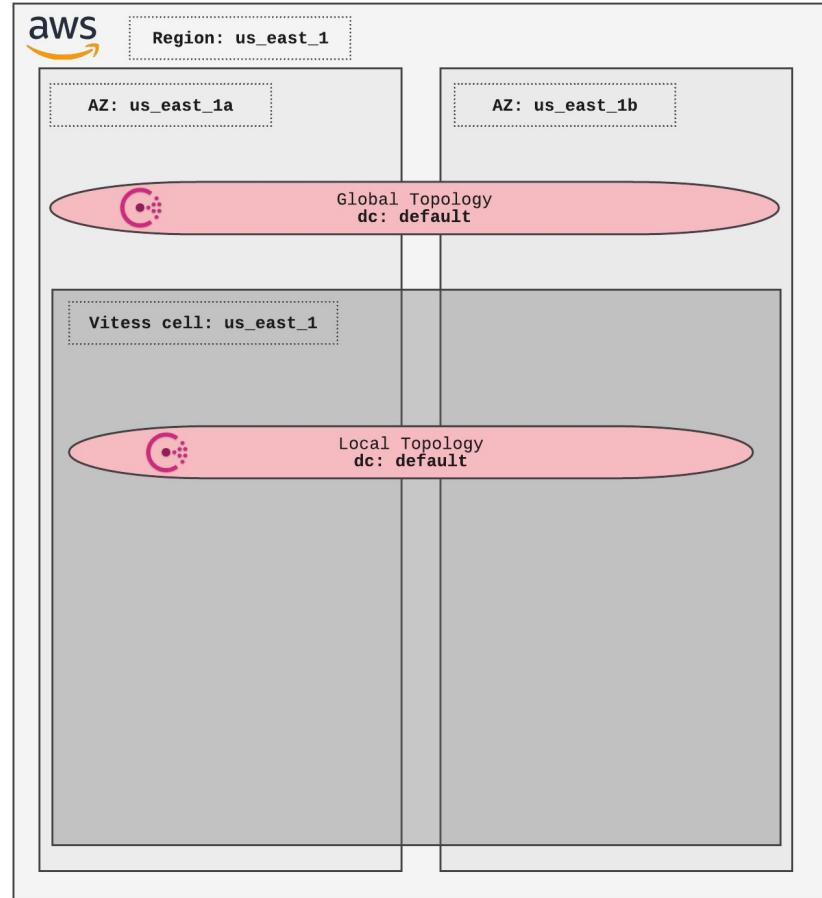
Vitess initial deployment

- Single **cell**.



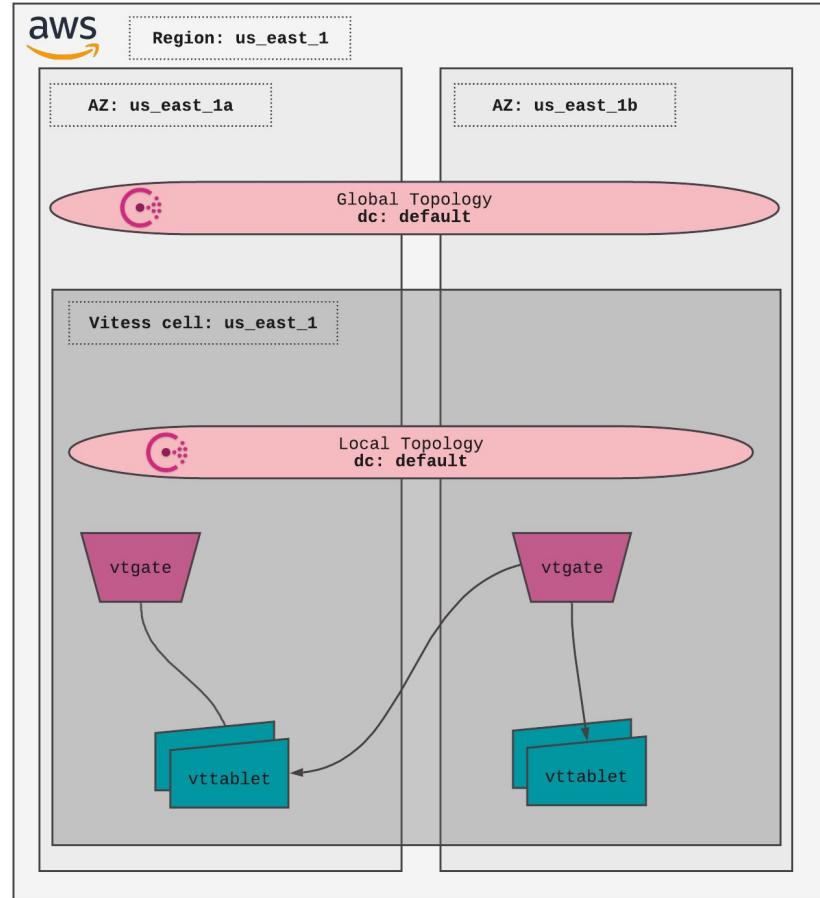
Vitess initial deployment

- Single **cell**.
- Same Consul **dc: default**.



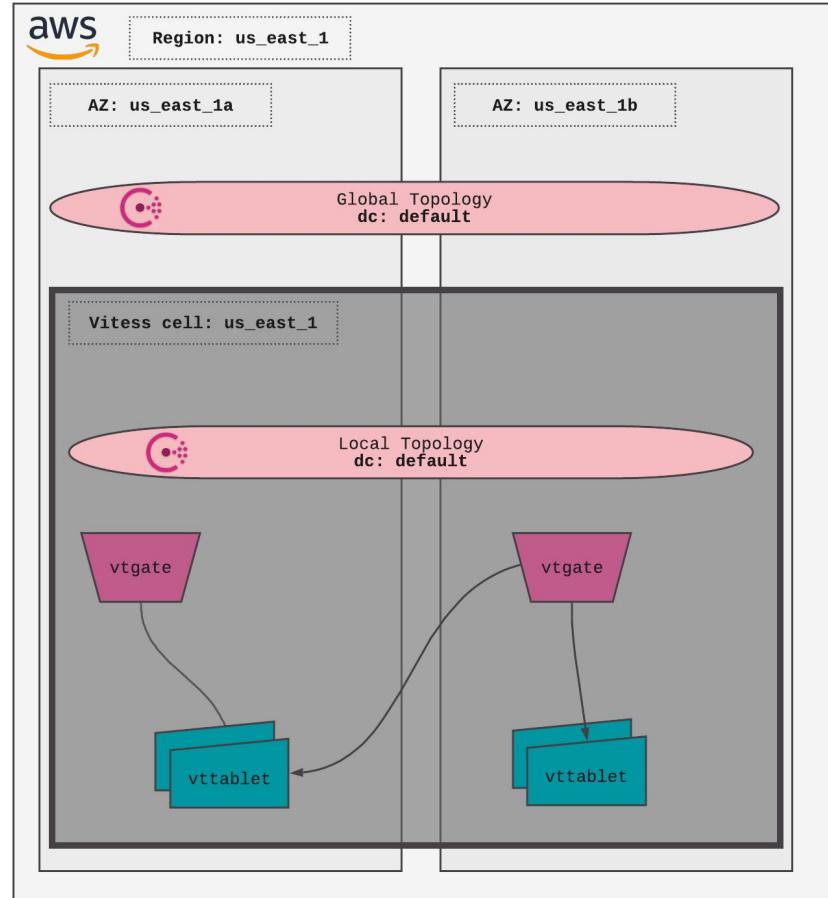
Vitess initial deployment

- Single **cell**.
- Same Consul dc: **default**.
- vtgates/tablets in different AZ's.

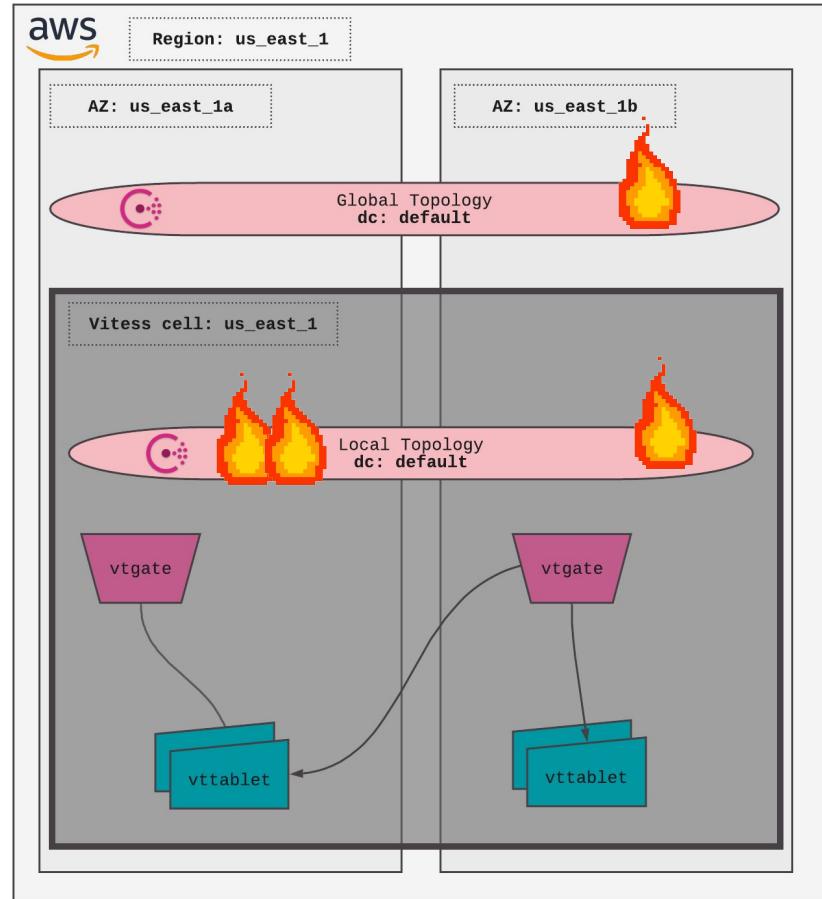


Vitess initial deployment

- A **single** cell across multiple AZ's (fundamental).
- Global and local topology using the same Consul cluster (circumstantial).



Vitess initial deployment



Vitess initial deployment



We fixed things!

- Defensive programming.
- Fixing bugs.

↗ **make the resilient topo cache even more resilient and informative** ●

#3641 by demmer was merged on Feb 14, 2018 • Review required

↗ **add resilient topo server caching for the full srv keyspace object** ●

#3610 by demmer was merged on Feb 5, 2018 • Review required

Problem

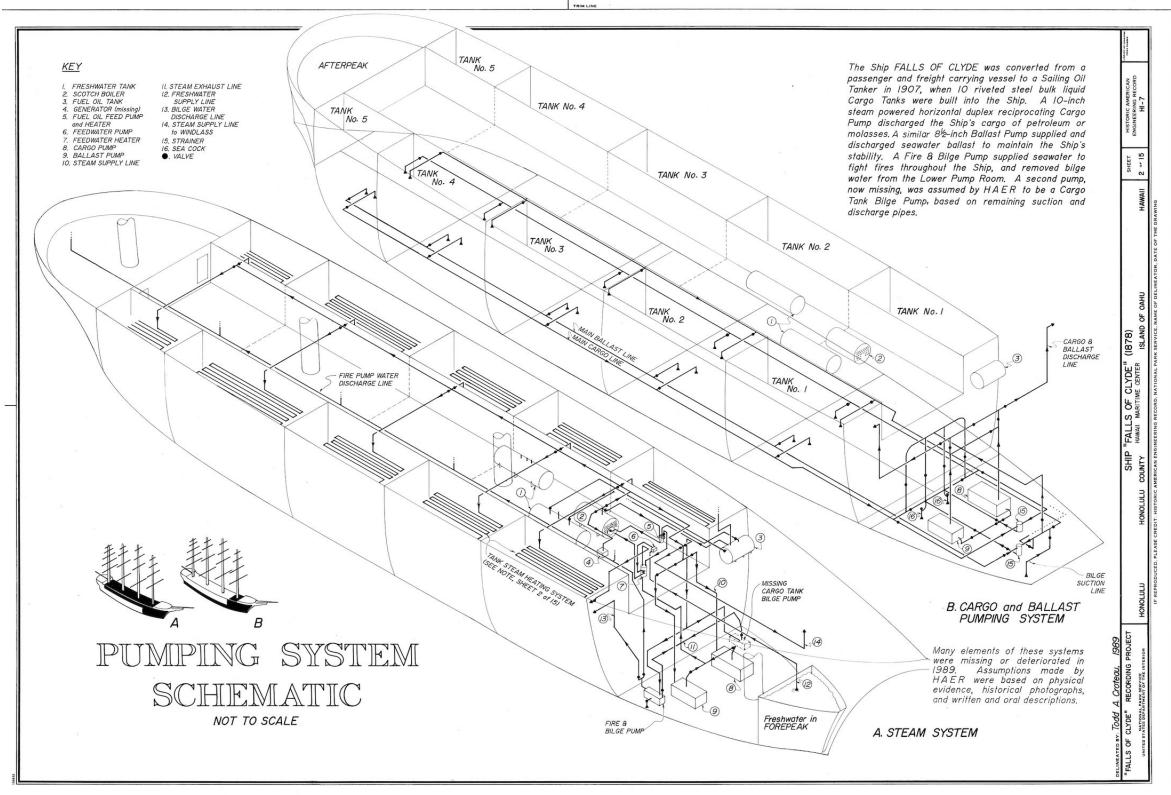
Surfaced a fundamental issue
in our deployment.



FAULT TOLERANCE & ISOLATION

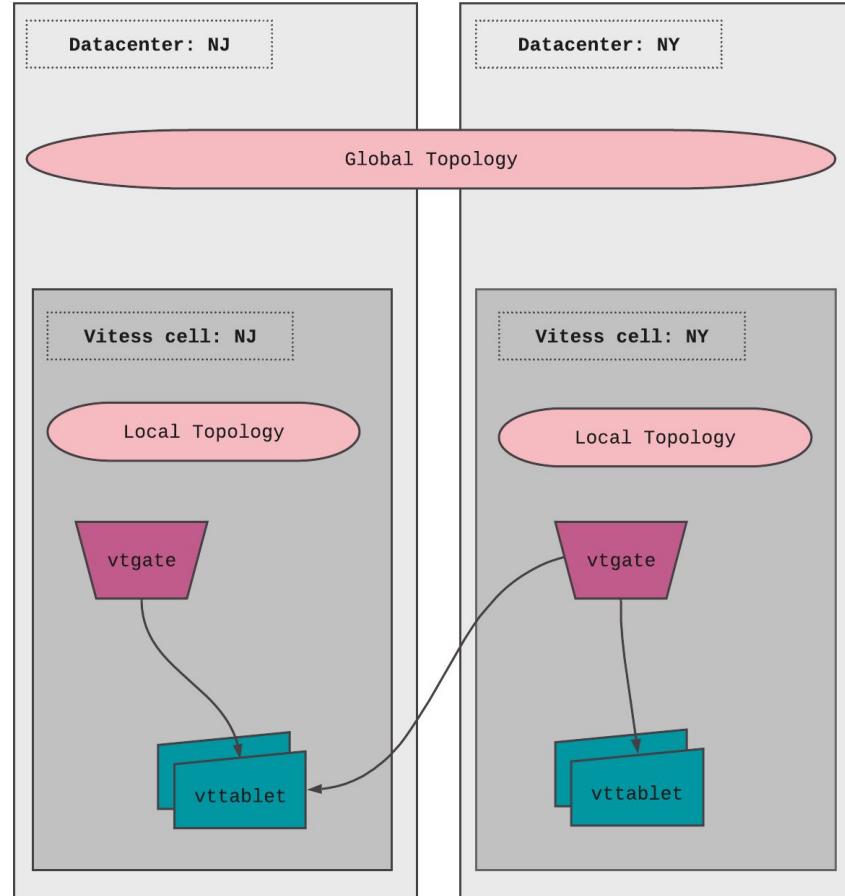
Resilient systems

- Minimize the blast radius.
 - Isolation is key.
 - Understand your dependencies.



Current deployment

Easy right?



Current deployment

Easy right?

[topology] Global Topology Serving Shards refactor #4496

 **Closed** rafael opened this issue on Jan 2 · 3 comments

 rafael commented on Jan 2 • edited

Member + ...

Feature Description and motivation

In order to route to a given shard, Vitess needs to know whether or not a shard is Serving. Currently, there is no canonical way to determine this. The following is a proposal to simplify Global topology and serving keyspace graph generation that will allow us to have a canonical representation of serving shards.

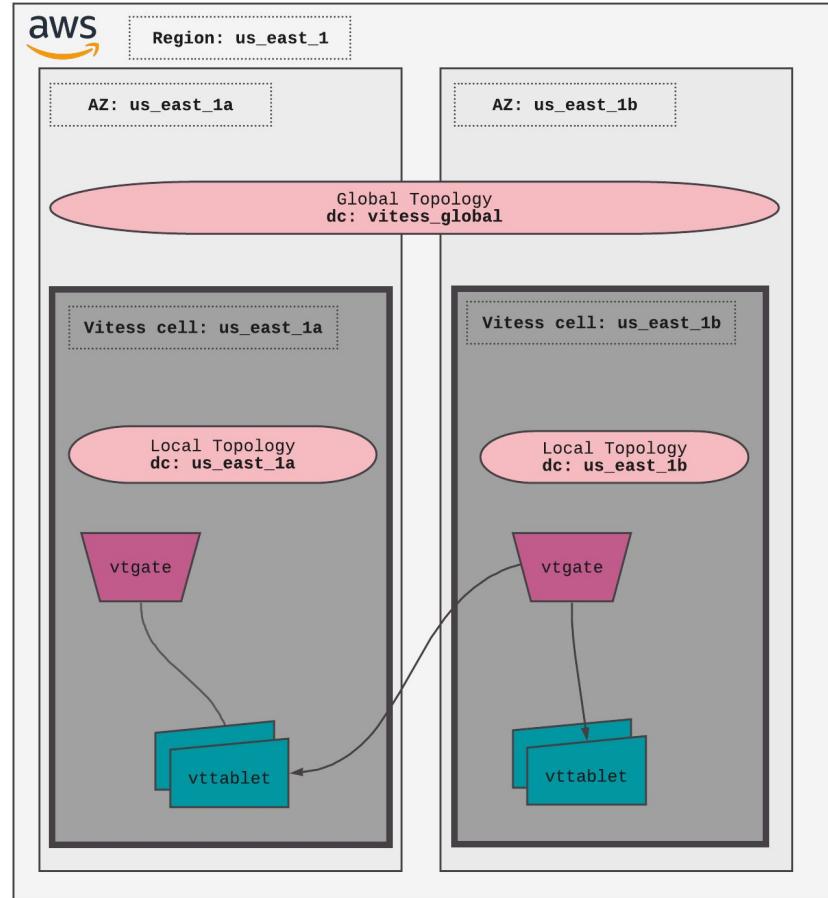
I see two main reasons to go in this refactor right now:

- Not having this canonical way to know if a shard is serving creates problems with multi cell regions that bleed to different parts of the cluster.
- Current design has the side effect that it creates problems with multi cell regions where it is possible to have vtgates in cells that don't have tablets. A workaround was proposed in #4482, but as part of that exploratory work, it seems like a higher level refactor like the one proposed here is a better approach.

 rafael closed this on Mar 21

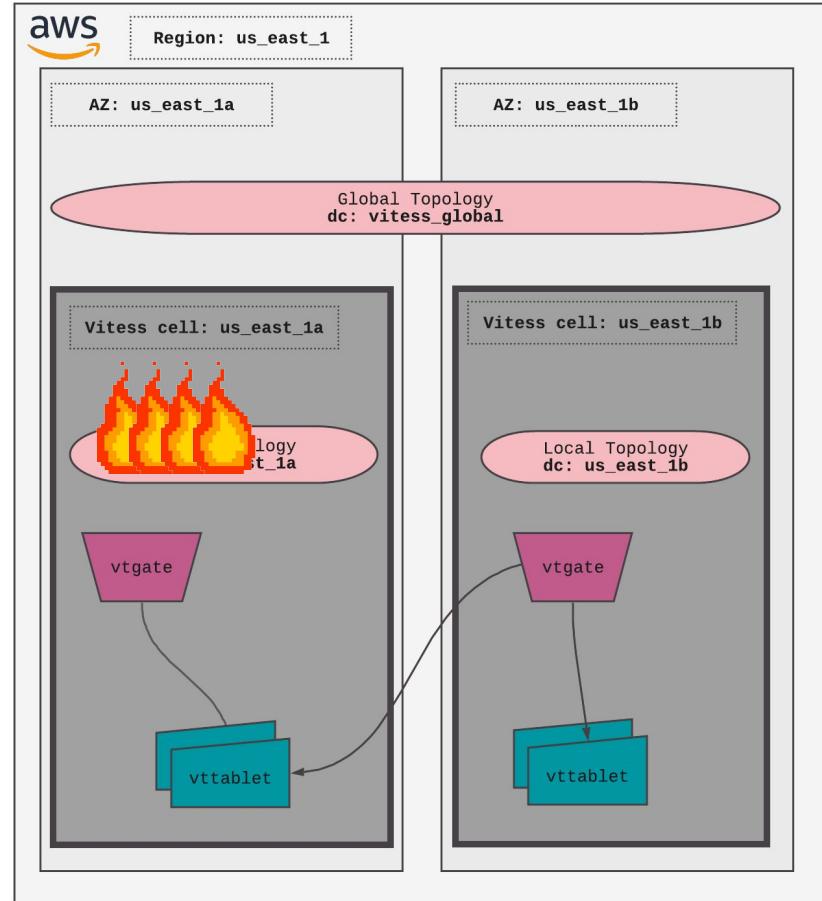
Current deployment

- Isolated topologies (one dc for each AZ and one for the global topo).
- Blast radius is mapped to physical infrastructure.

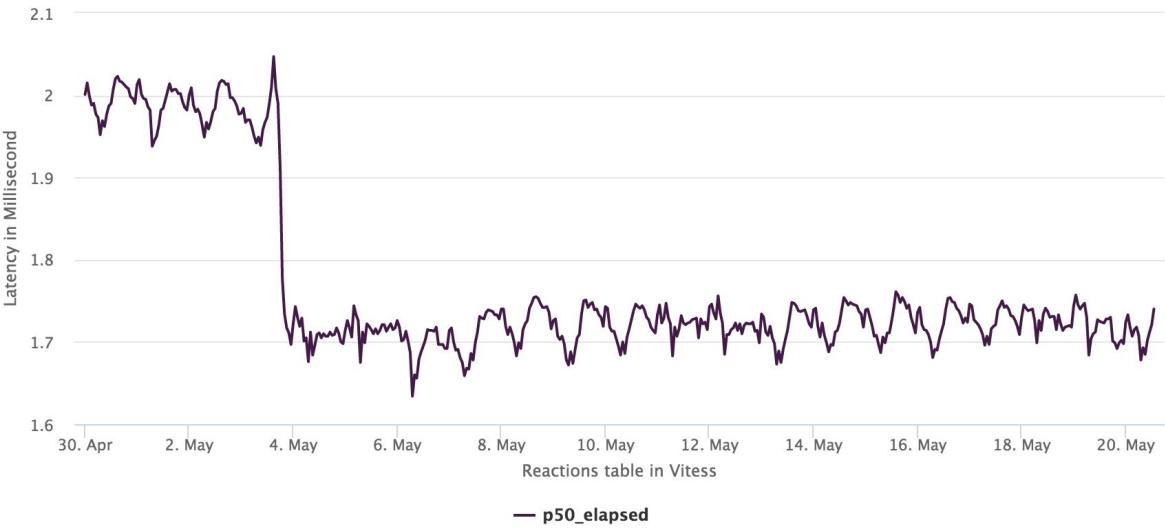


We have benefited already

- AZ failure during backup time.
- Single cell was affected!



Performance wins



Key Lessons

Complex system failures

- Complex systems are intrinsically dangerous systems.
- Complex systems are heavily and successfully defended against failure.
- Catastrophe is always just around the corner.
- Complex systems contain changing mixtures of failures latent within them.

A Short Treatise on the Nature of Failure; How Failure is Evaluated; How Failure is Attributed to Proximate Cause; and the Resulting New Understanding of Patient Safety - Richard I. Cook, MD (2000)

KEY LESSONS

Complex system failures

Humility towards complexity.

Reach out to other fields and learn from their experience.



Thank you!

P.S. We are hiring!



Q&A



mistermysql
@mistermysql

Follow ▾

AMA. All questions must be in the form of valid SQL queries.

5:21 PM - 17 Nov 2019

Q ↗ ♥



Thank you!

P.S. We are hiring!



Appendix

Vitess configuration

Here are some of the configuration settings that we use in our setup.

They are the result of several years of tuning. We are sharing them so that the community can benefit as well.

```
./vtgate
-buffer_size 10000
-discovery_high_replication_lag_minimum_serving 5m
-discovery_low_replication_lag 30s
-enable_buffer
-gateway_implementation discoverygateway
-gateway_initial_tablet_timeout 120s
-grpc_initial_conn_window_size 1073741824
-grpc_initial_window_size 1073741824
-grpc_keepalive_time 10s
-grpc_keepalive_timeout 10s
-grpc_server_initial_conn_window_size 1073741824
-grpc_server_initial_window_size 1073741824
-min_number_serving_vttables 2
-mysql_server_query_timeout 60s
-mysql_server_read_timeout 60s
-mysql_server_write_timeout 60s
-normalize_queries
-service_map grpc-vtgateservice
-srv_topo_cache_refresh 5s
-srv_topo_cache_ttl 8760h
-tablet_refresh_interval 60s
-tablet_refresh_known_tablets=false
-tablet_types_to_wait MASTER,REPLICA
-topo_read_concurrency 1
-transaction_mode MULTI
```

Vitess configuration

Here are some of the configuration settings that we use in our setup.

They are the result of several years of tuning. We are sharing them so that the community can benefit as well.

```
./vttablet
-binlog_use_v3_resharding_mode
-degraded_threshold 30s
-enable-autocommit
-enable_replication_reporter
-enable_semi_sync
-grpc_initial_conn_window_size 1073741824
-grpc_initial_window_size 1073741824
-grpc_server_initial_conn_window_size 1073741824
-grpc_server_initial_window_size 1073741824
-grpc_server_keepalive_enforcement_policy_min_time 2s
-health_check_interval 1s
-queryserver-config-idle-timeout 1200
-queryserver-config-passthrough-dmls
-queryserver-config-pool-size 150
-queryserver-config-schema-reload-time 300
-queryserver-config-transaction-cap 150
-queryserver-config-txpool-timeout 3
-unhealthy_threshold 1h
```