

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

Rafael Chacón
Guido Iaquinti



SPEAKERS



Rafael Chaon

he/him/his

 twitter.com/rafaelchaon

Staff Software Engineer - Slack



Guido Iaquinti

he/him/his

 twitter.com/guidoiaquinti

 guido.iaquinti.com

Site Reliability Engineer - Freelance

Agenda

A decorative background on the left side of the slide. It features a white surface with several parallel white lines. Small wooden spheres, some with colored segments (red, green, yellow), are scattered across these lines, creating a sense of depth and movement.

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 displays the Slack interface for the #social-media channel. On the left is a dark purple sidebar with the 'Acme Inc.' workspace name and a list of channels including #social-media (highlighted in blue), #design-team, #helpdesk, #accounting, #design-crit, #help-design, #media-and-pr, #triage-issues, #design-team-sf, and #slackbot. Below the channels are direct messages with Zoe Maxwell, Leland..., Florence Garret, and Liza Zhang. The main channel view shows a header with the channel name, member count (21), and a search bar. The message history includes a post from Sara Parker praising @zoe, a post from Zoe Maxwell expressing excitement, a bot announcement for a 'Team Status Meeting' starting in 15 minutes, a post from Harry Boone about a team sync, and a post from Jeremy Stevens about meeting notes. A '1/9 Meeting Notes' document is attached to the last message. On the right, a sidebar for the channel shows options like '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?

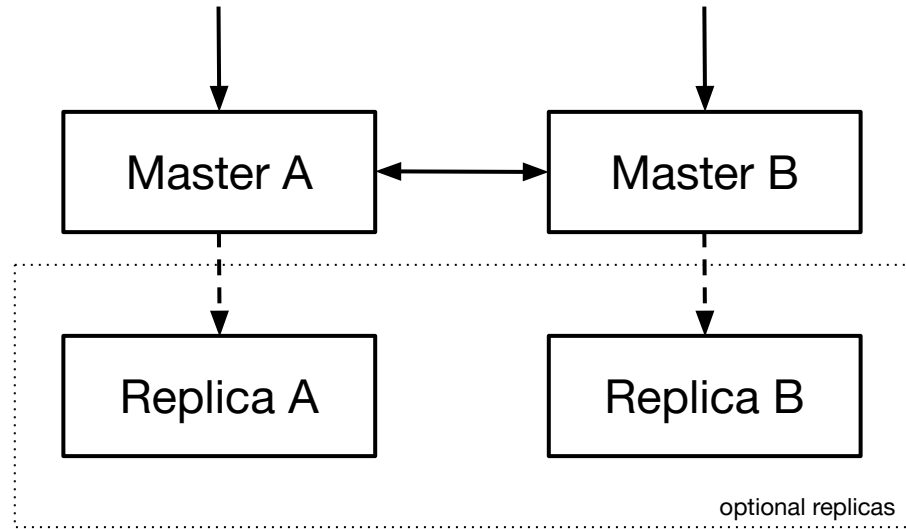
For more details please see the **presentations** on the side.

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 laquinti*
- *“Smooth scaling: Slack’s journey toward a new database” - Ameet Kotian*

Legacy shards

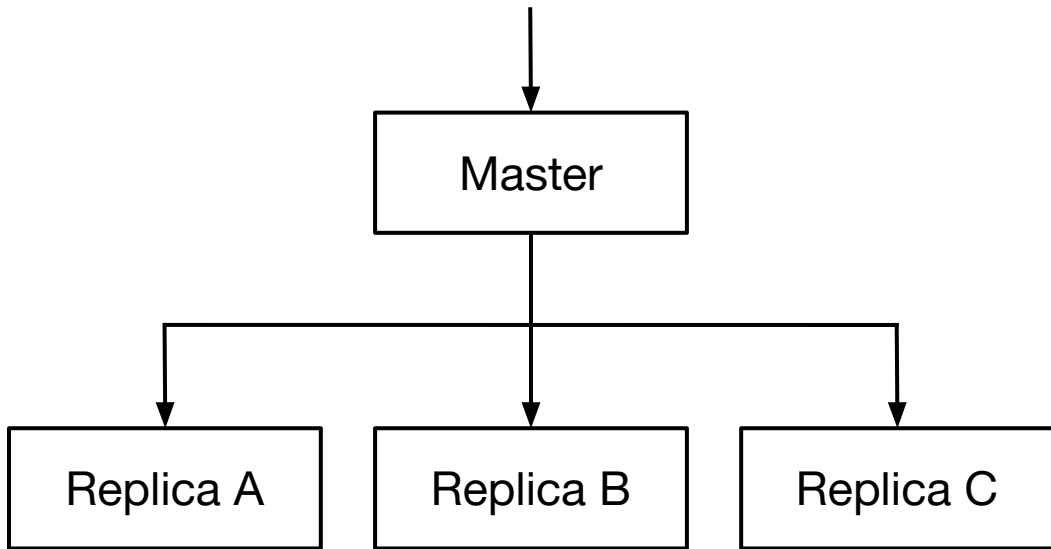
Application level
team-sharded **active
master-master**
MySQL setup.



Vitess shards

Master-replica

MySQL setup fully managed by Vitess.



DATABASES AT SLACK

Stats

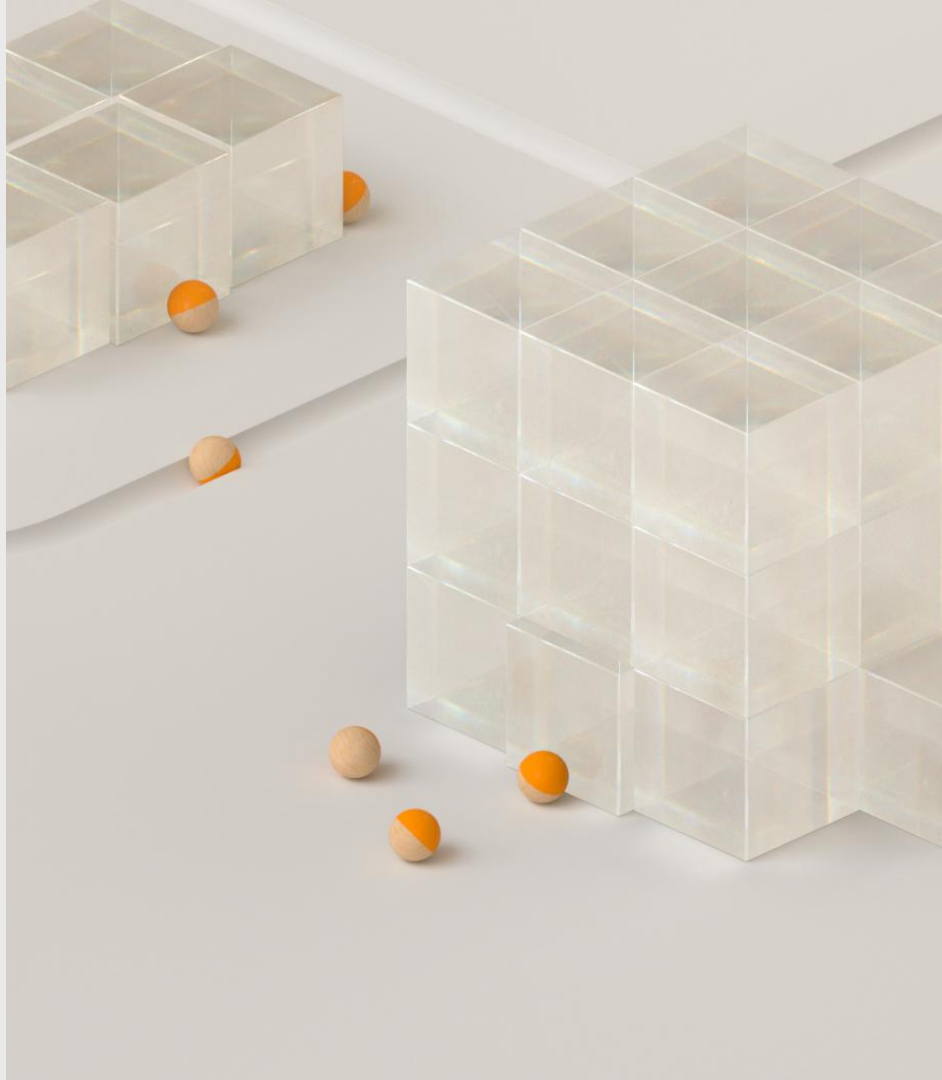
Queries per day: **53+ billion**

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



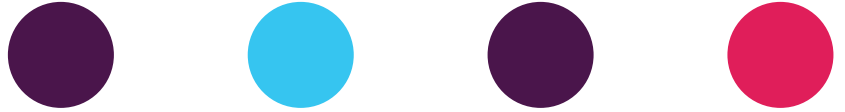
RUNNING DATABASES IN THE CLOUD

Variable infrastructure



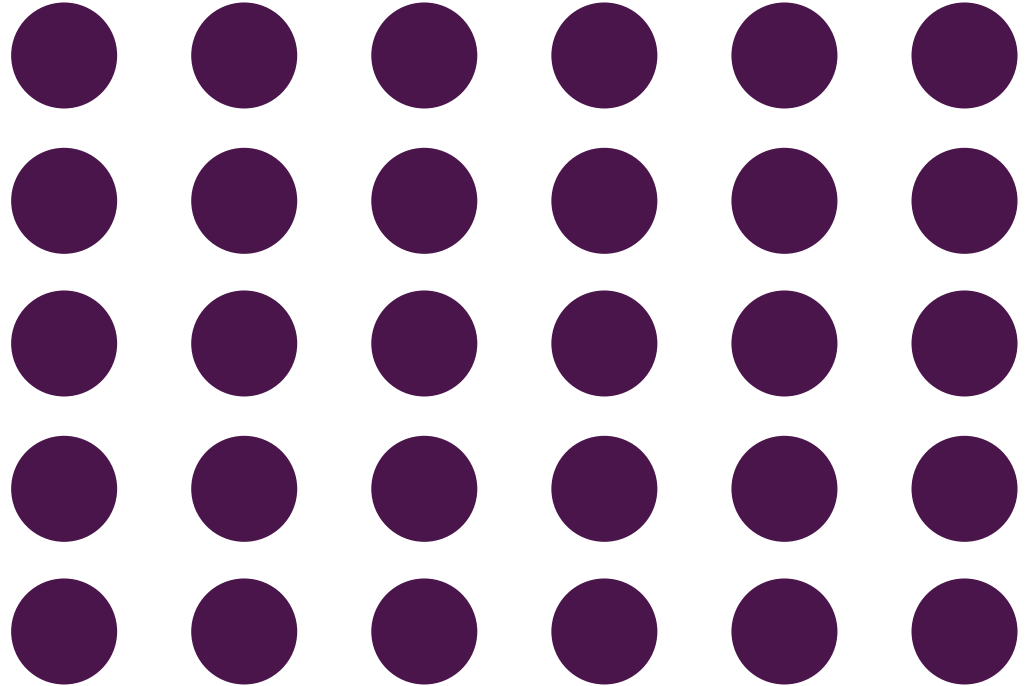
RUNNING DATABASES IN THE CLOUD

Variable infrastructure



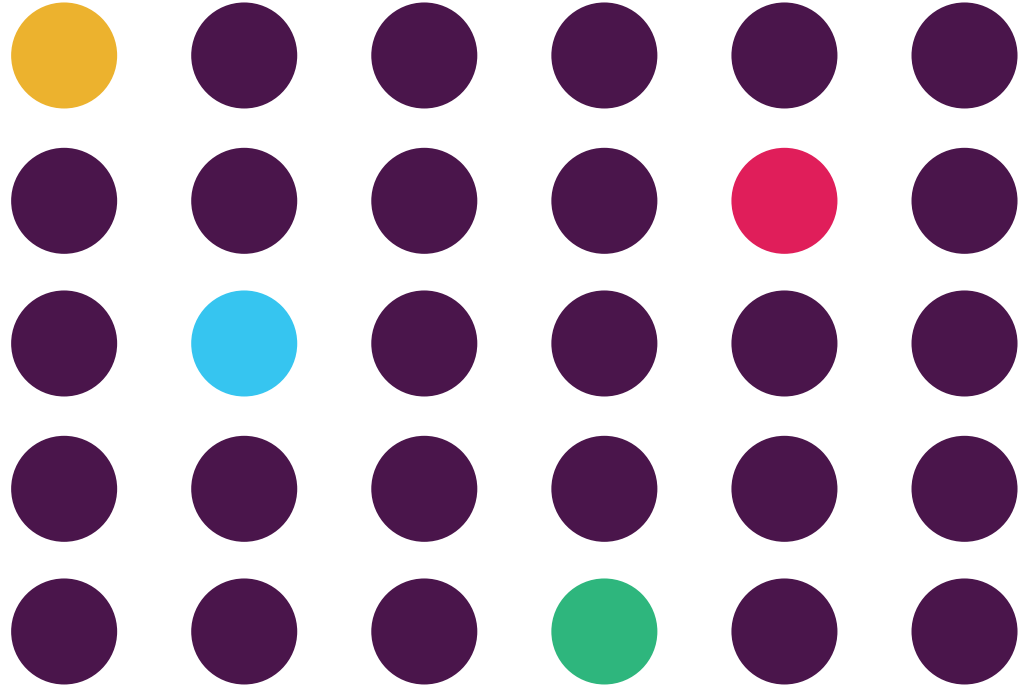
RUNNING DATABASES IN THE CLOUD

Variable infrastructure



RUNNING DATABASES IN THE CLOUD

Variable infrastructure



Immutable infrastructure

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



RUNNING DATABASES IN THE CLOUD

Instance failure

The airplane analogy



Instance failure

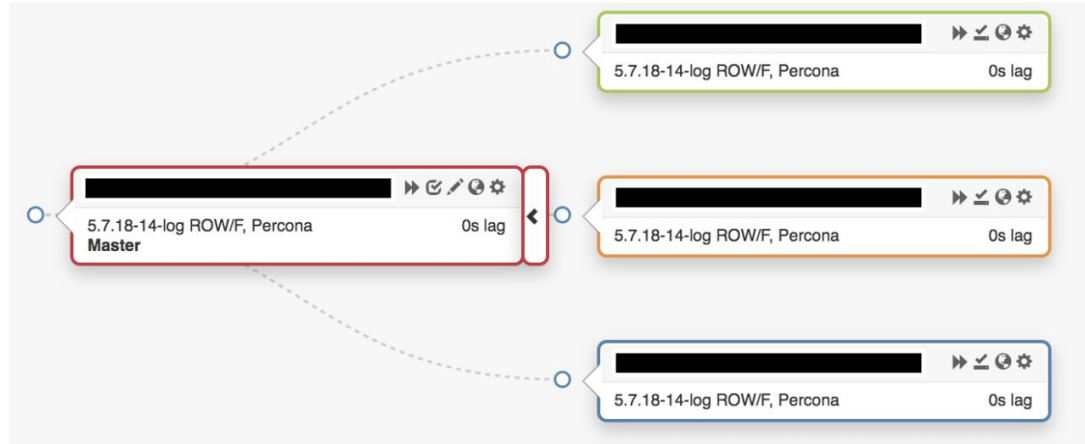
Always reprovision: challenges

- 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 = 9999999999999
```

```
rpl_semi_sync_master_wait_no_slave = 1
```

```
sync_binlog = OFF
```

```
innodb_flush_log_at_trx_commit = 2
```

RUNNING DATABASES IN THE CLOUD

How we run Vitess

- AWS



How we run Vitess

- AWS
- EC2 not k8s



A screenshot of a tweet from Kelsey Hightower (@kelseyhightower) dated February 13, 2018. The tweet discusses Kubernetes' capabilities for stateful workloads. The interface includes a profile picture, a 'Follow' button, and engagement metrics for replies, retweets, and likes.

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

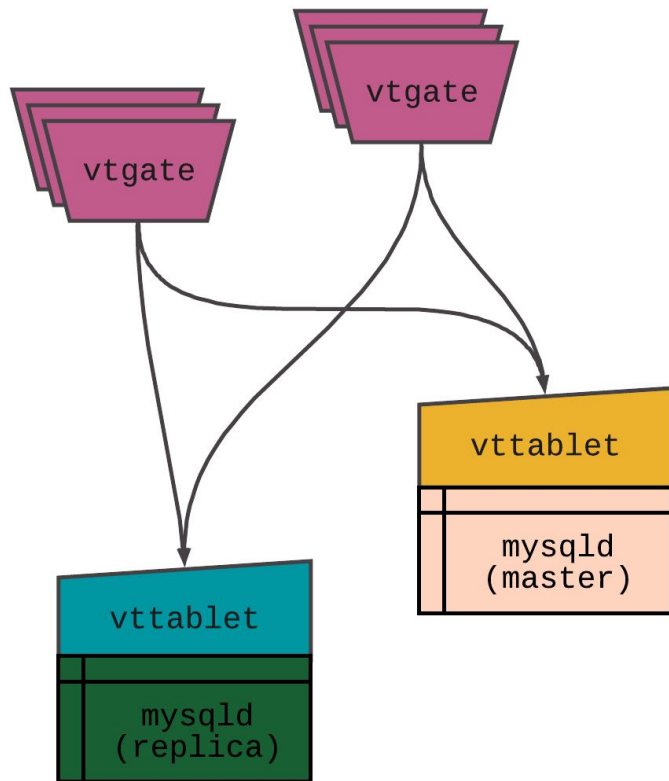
304 Retweets 705 Likes



 24  304  705

How we run Vitess

- AWS
- EC2 not k8s
- ASG for stateless components



How we run Vitess

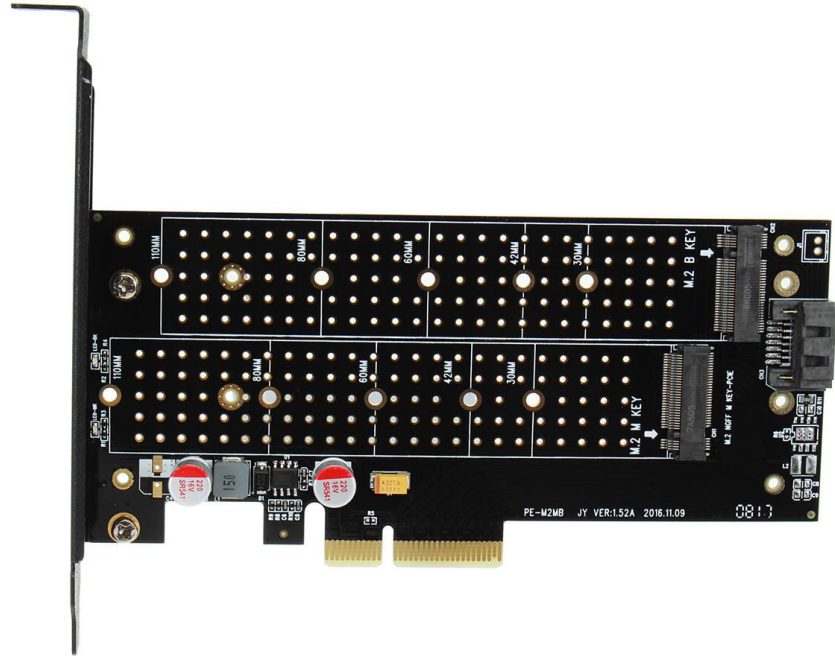
- AWS
- EC2 not k8s
- ASG for stateless components
- MySQL 5.7 (Percona)



PERCONA

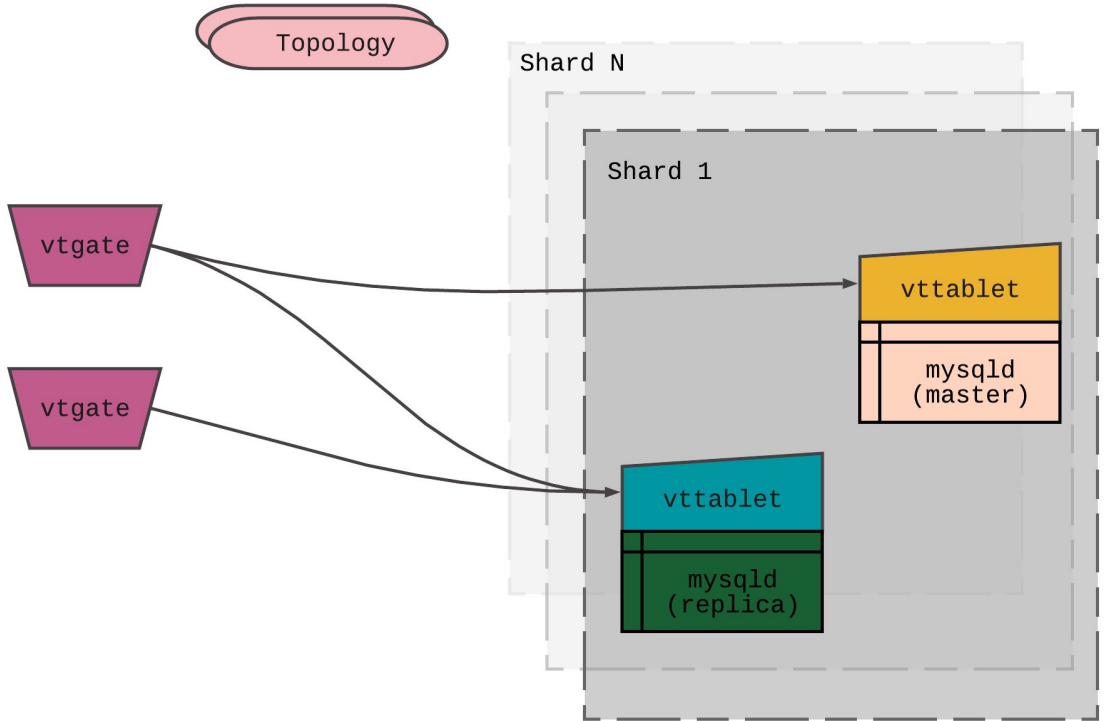
How we run Viteess

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

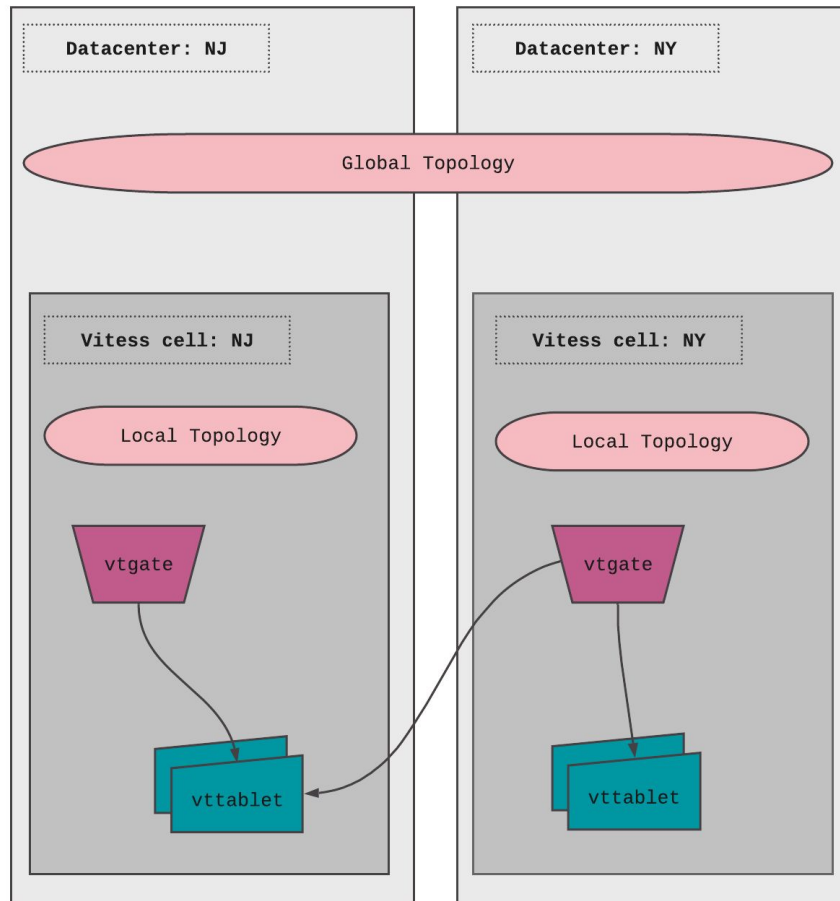


Fault tolerance & isolation

Vitess architecture



Vitess standard deployment



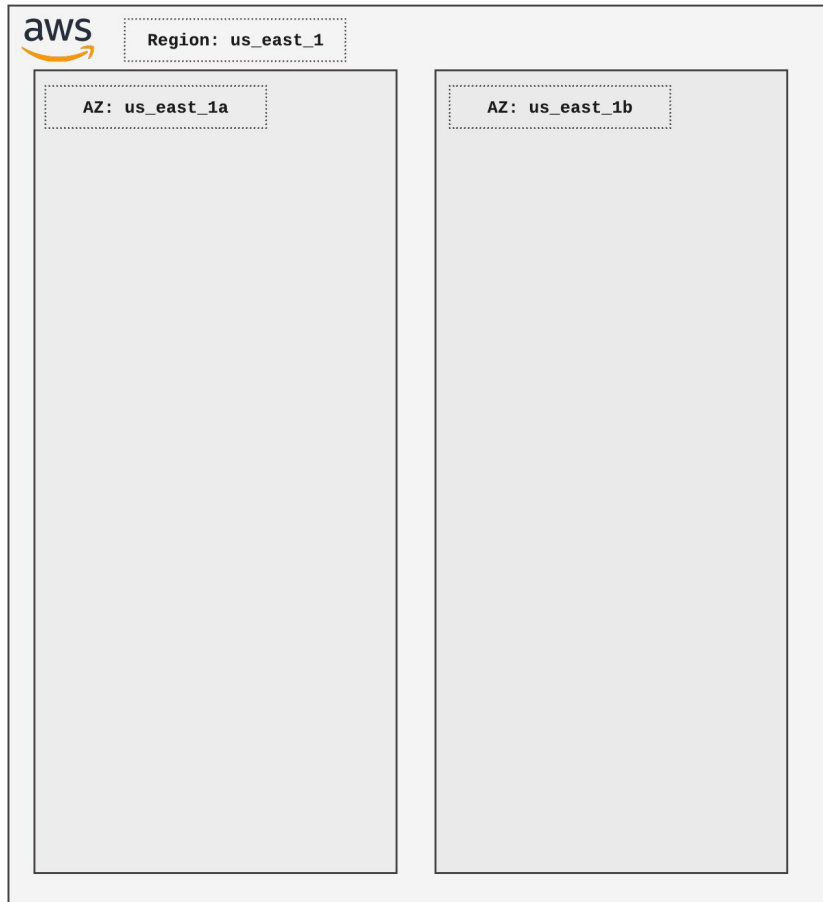
Slack cloud infrastructure

- 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.



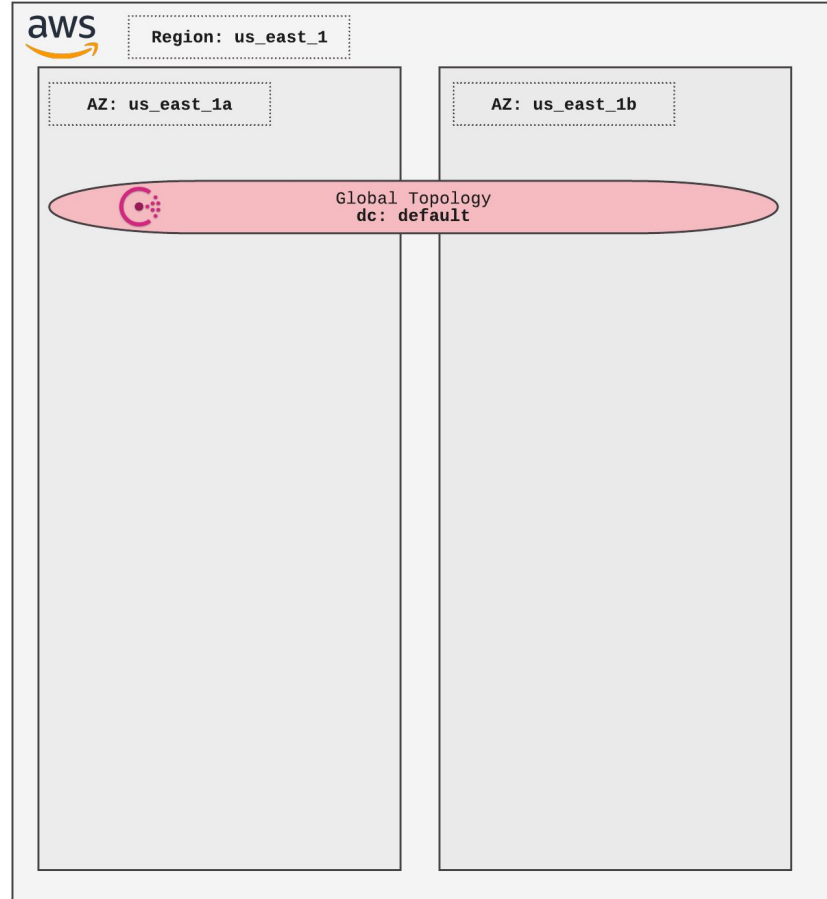
Vitess initial deployment

- We now have multiple clusters in different regions.



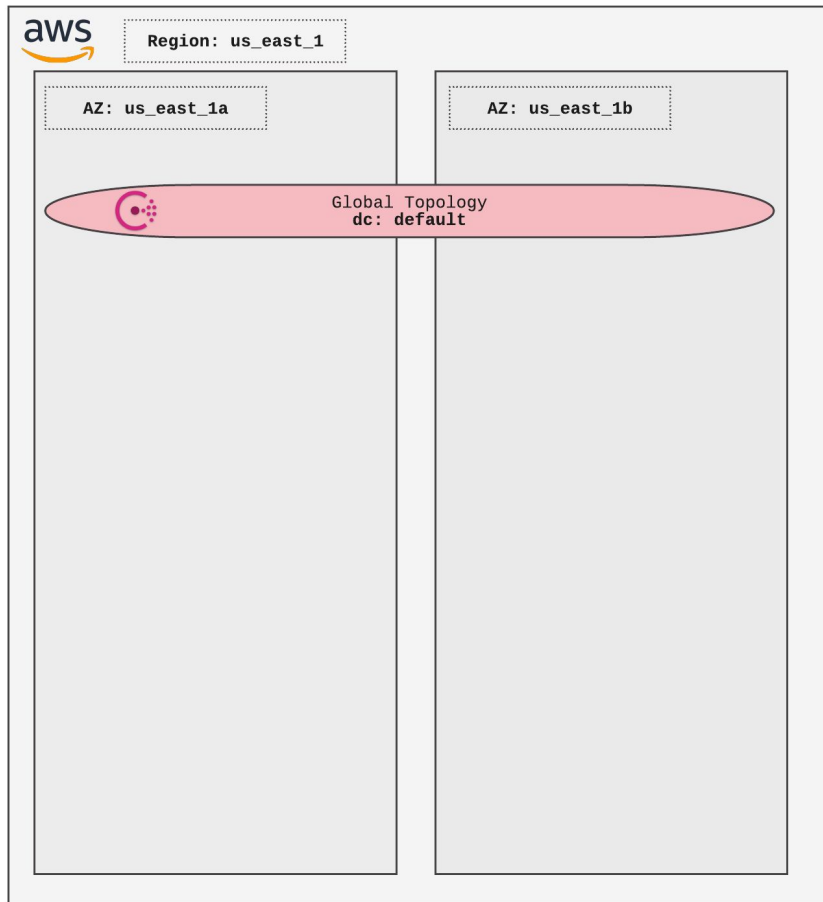
Vitess initial deployment

- We use Consul



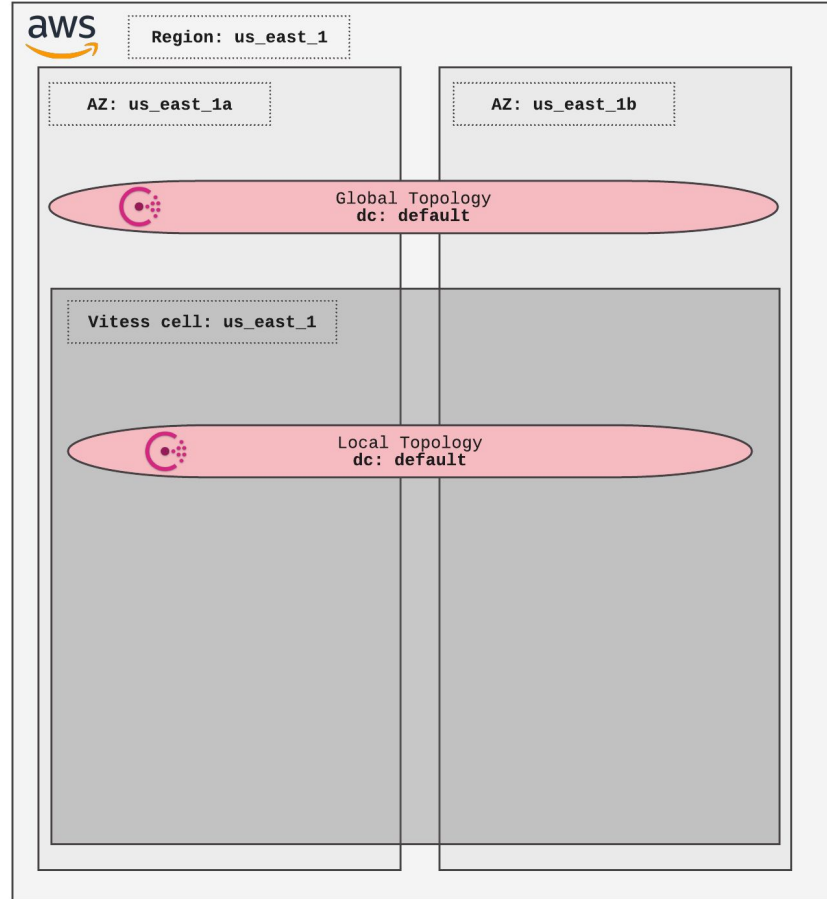
Vitess initial deployment

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



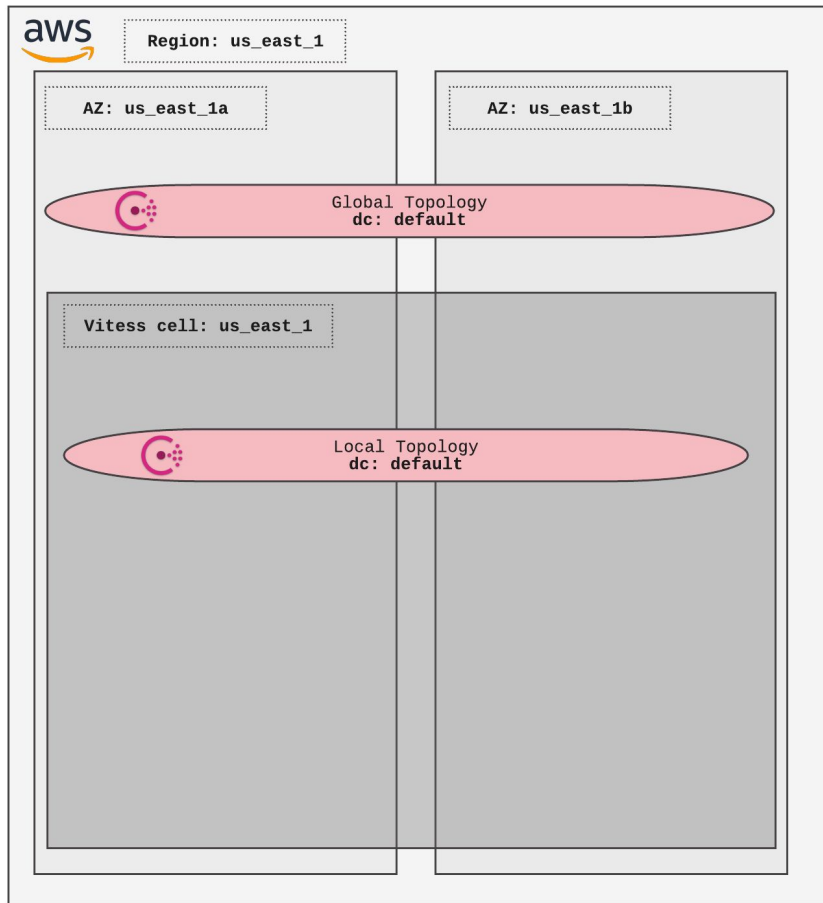
Vitess initial deployment

- Single **cell**.



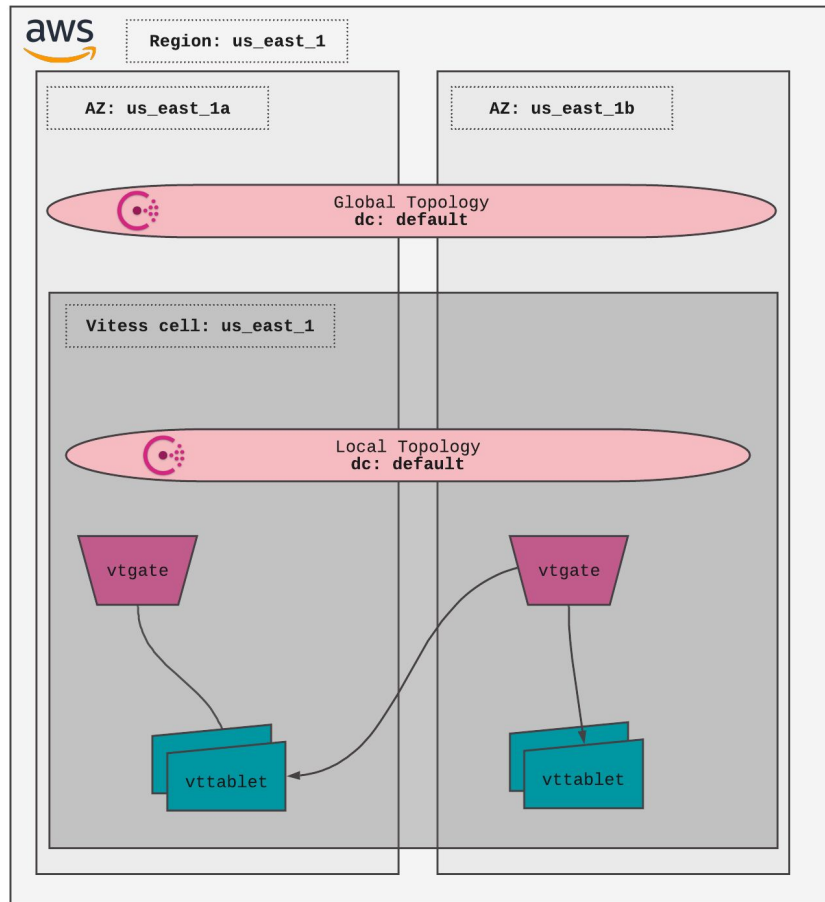
Vitess initial deployment

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



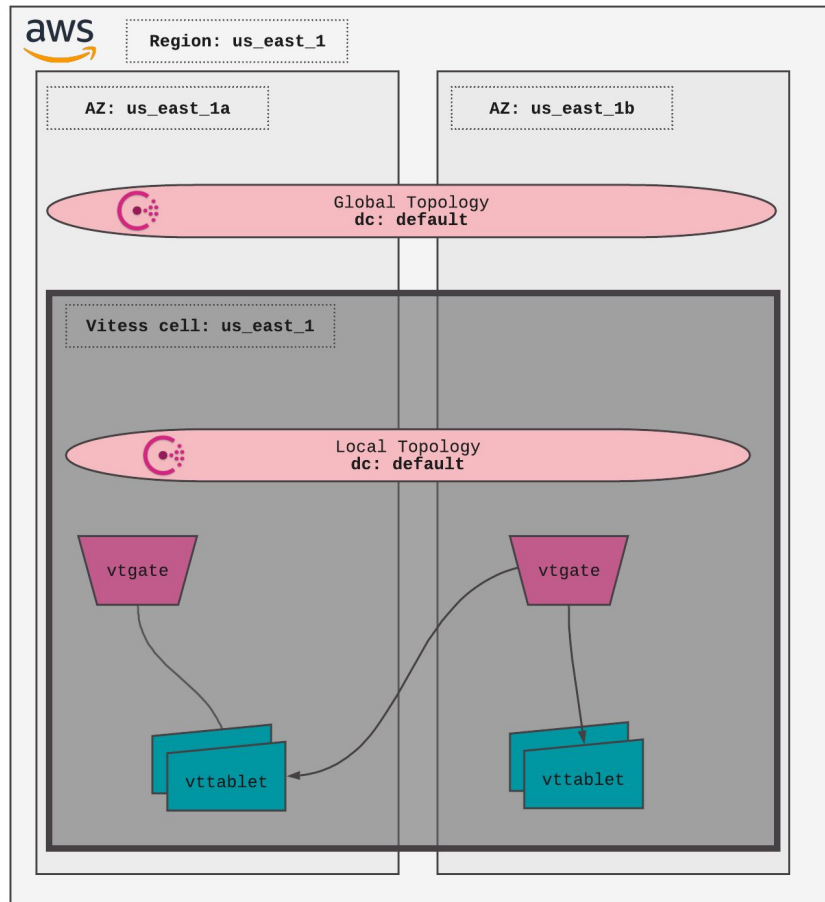
Vitess initial deployment

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

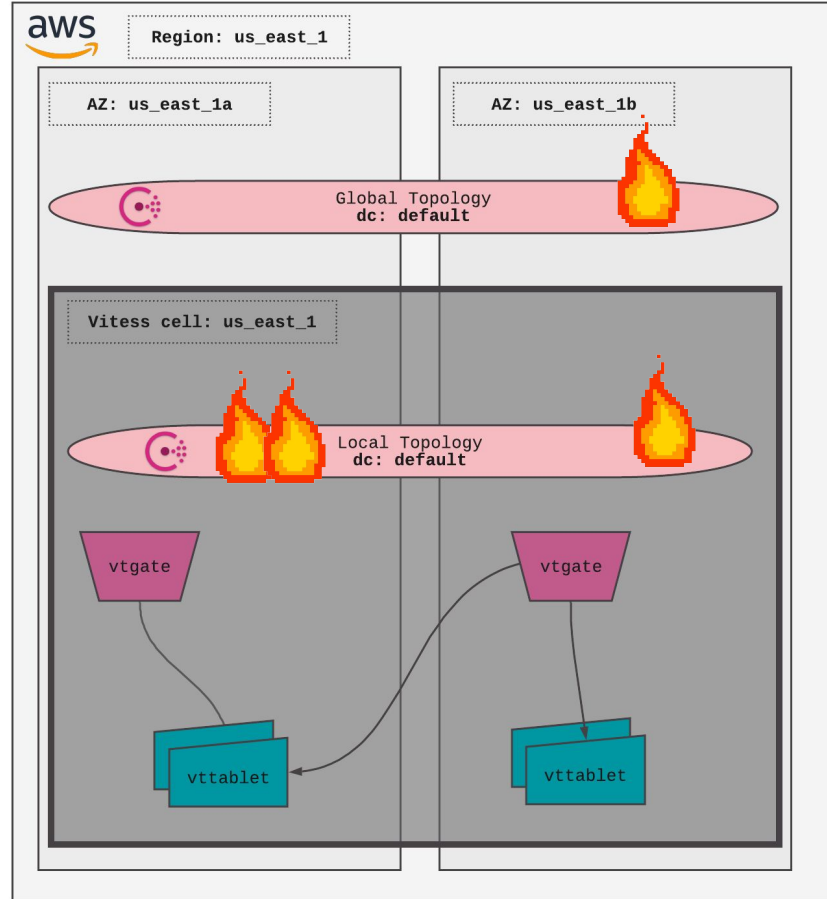


Vitess initial deployment

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



Vitess initial deployment



FAULT TOLERANCE & ISOLATION

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

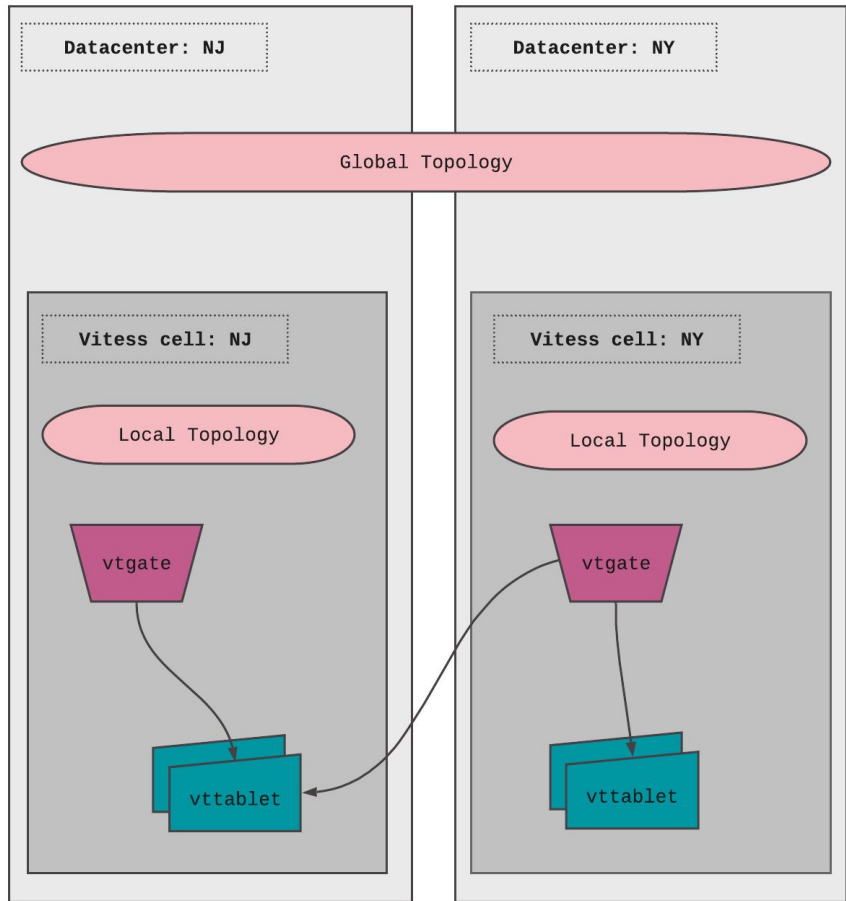
Problem

Surfaced a fundamental issue
in our deployment.



Current deployment

Easy right?



Current deployment

Easy right?

[topology] Global Topology Serving Shards refactor #4496



rafael opened this issue on Jan 2 · 3 comments



rafael commented on Jan 2 • edited ▾

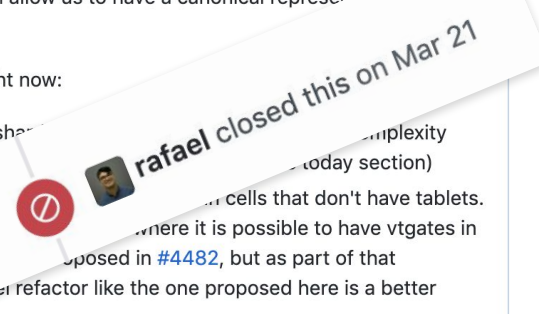
Member +😊 ⋮

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:

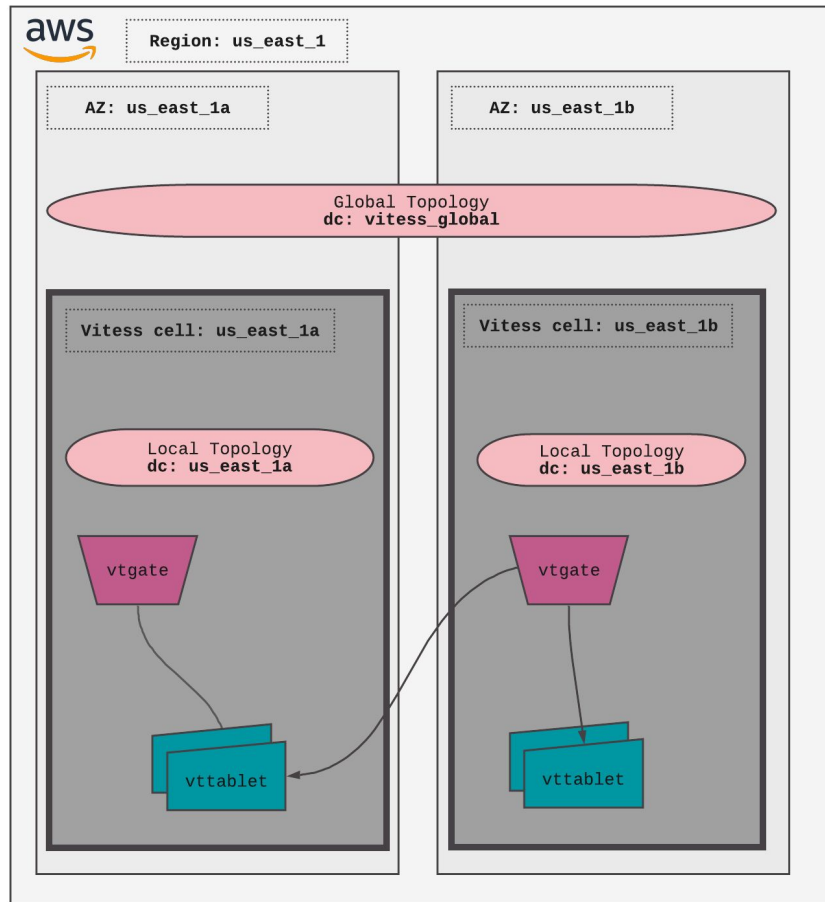
- Not having this canonical way to know if a shard is serving adds complexity that bleeds to different parts of the codebase (e.g. `vtgate` today section)
- Current design has the side effect that it requires `vtgate` to handle cells that don't have tablets. This creates problems with multi cell replication 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.



Assignee
No one assigned
Labels
None yet
Project
None yet
Milestone
No milestone
Notifications
You're not subscribed

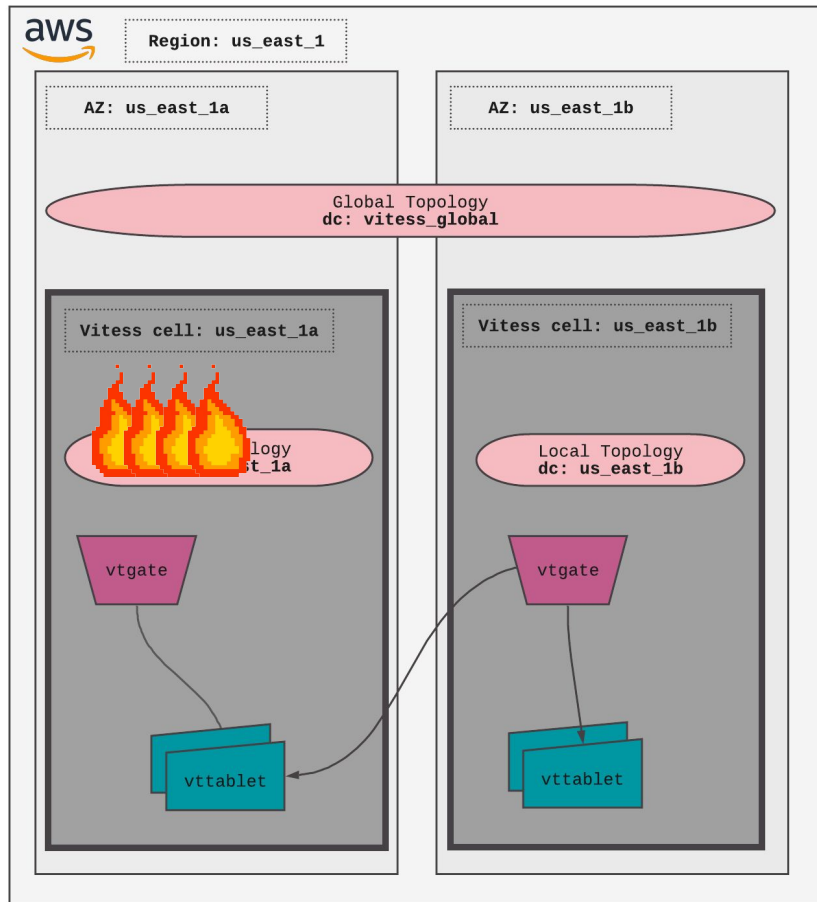
Current deployment

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

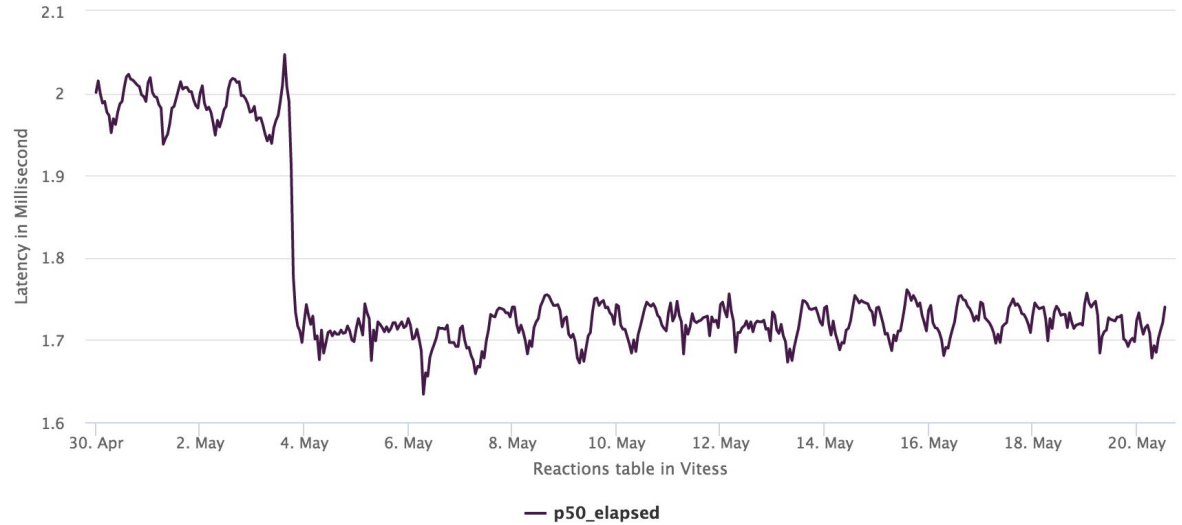


We have benefited already

- 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



misternysql
@misternysql

Follow



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

5:21 PM - 17 Nov 2019



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_vttablets 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.

```
./vtttablet
  -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
```