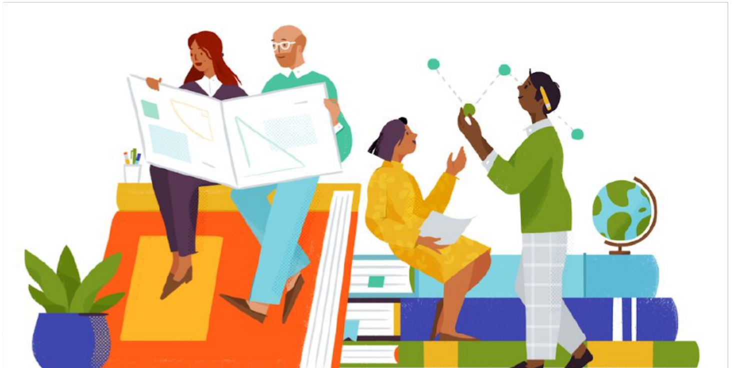
October 2, 2019

# Strength in Numbers: Slack's Database Architecture

Josh Varner, Guido Iaquinti



# $whoami

**Josh Varner**

Staff Database Reliability Engineer

twitter.com/awgy

# $whoami

## Guido Iaquinti

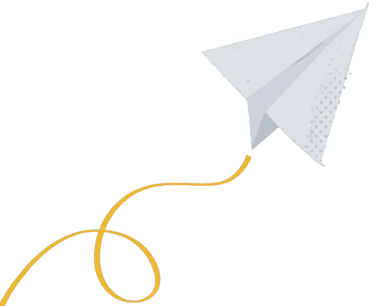
- Freelance Site Reliability Engineer
- previously SRE at Slack for several years

twitter.com/guidoiaquinti

🌍 guido.iaquinti.com

# Agenda

1. Databases at Slack: past, present and future
2. Running databases in a cloud world
3. Isolation & distribution
4. Conclusions
5. Q&A

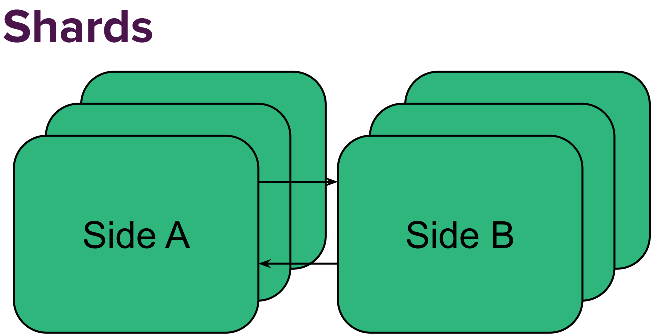


slack

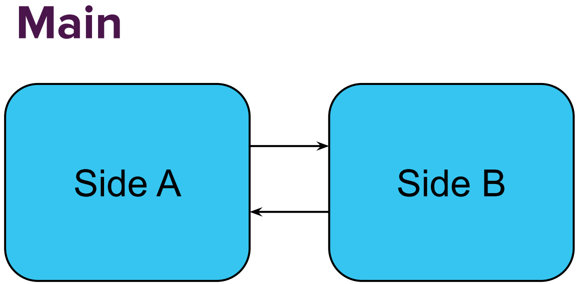
# Databases at Slack: past, present and future

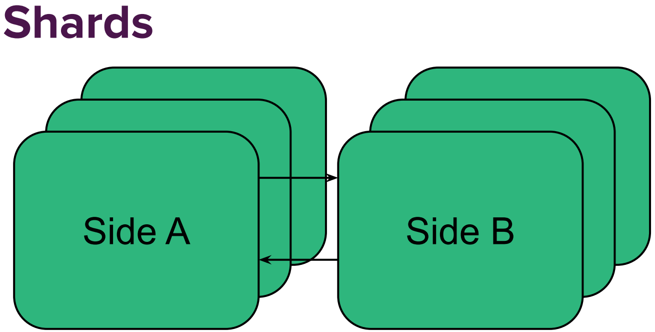
# Database at Slack: past

# Database at Slack: past



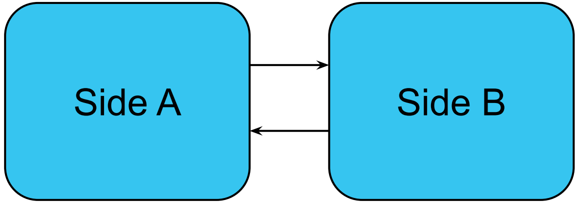
# Database at Slack: past



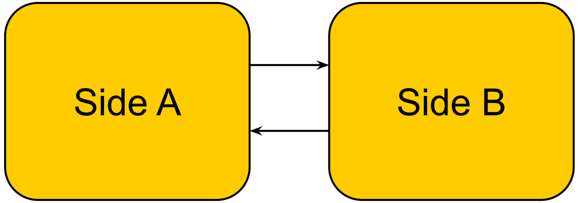


# Database at Slack: past

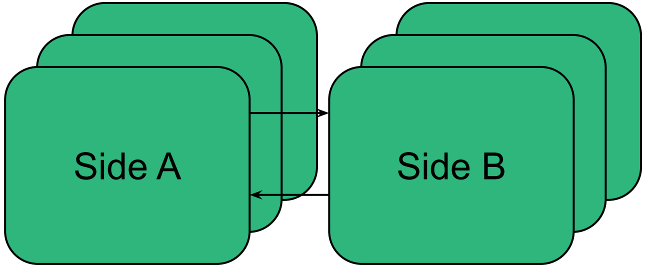
Main



Aux

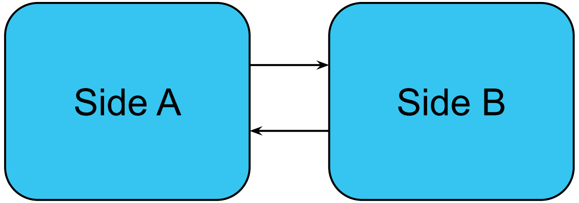


Shards

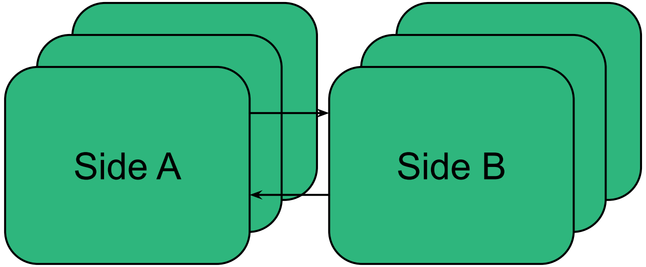


# Database at Slack: past

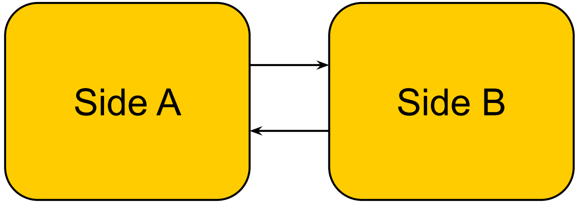
### Main



### Shards



### Aux

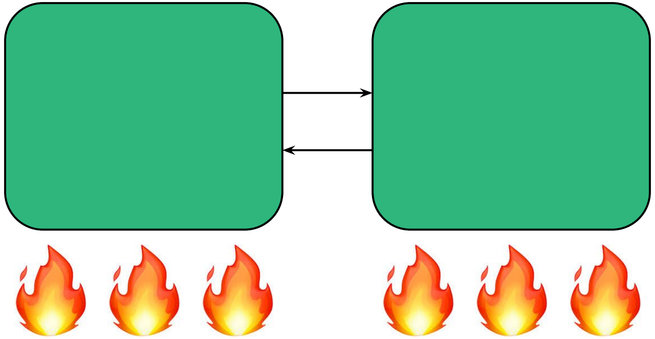


# Database at Slack: past

## What didn’t work/why we changed

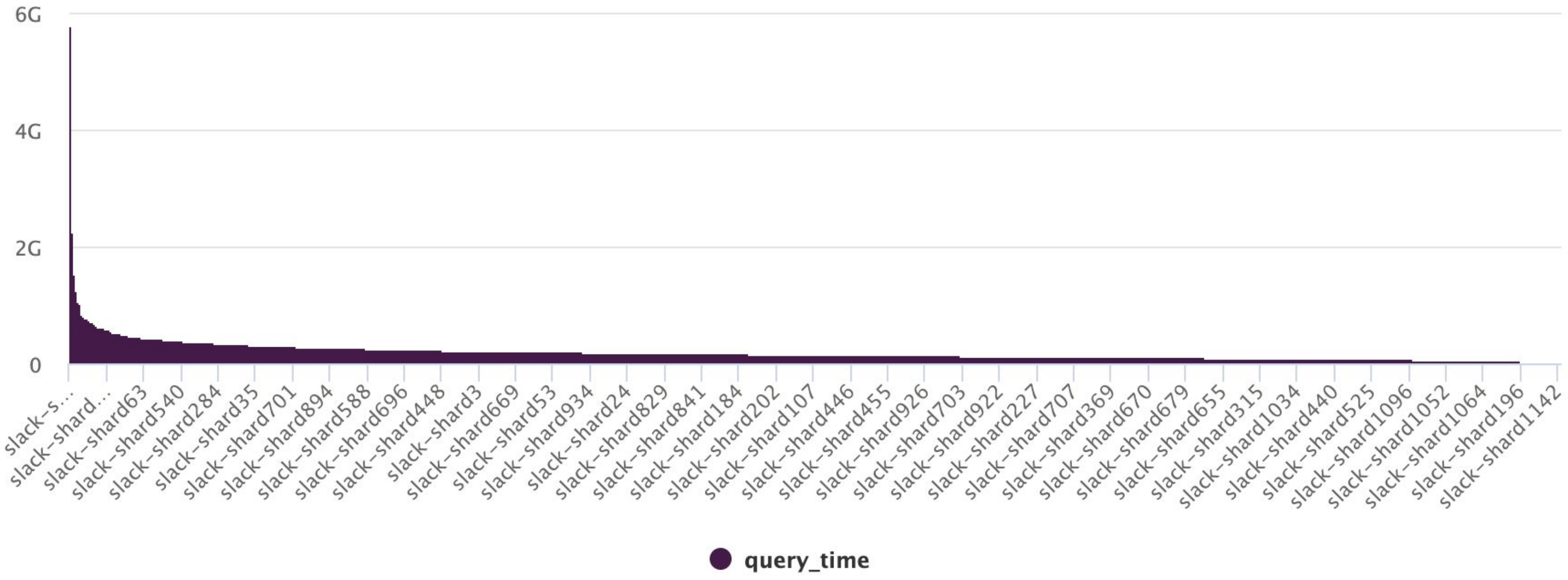
# Database at Slack: past

## What didn’t work/why we changed: shard size



# Database at Slack: past

## What didn’t work/why we changed: inefficient distribution



# Database at Slack: past

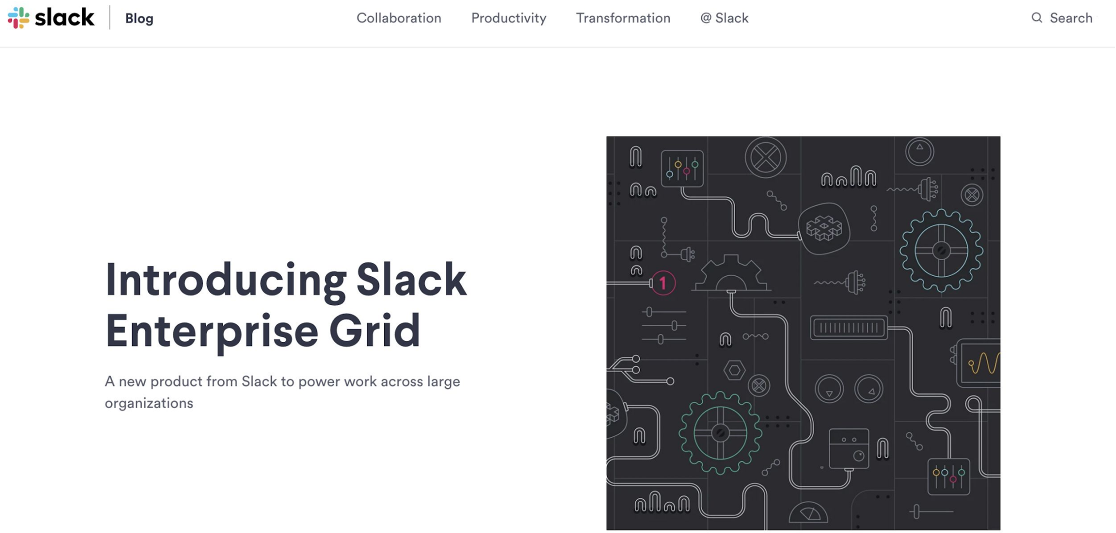
## What didn’t work/why we changed: operational overhead

# Database at Slack: past

## What didn’t work/why we changed: operational overhead

# Database at Slack: past

## What didn't work/why we changed: unique sharding model



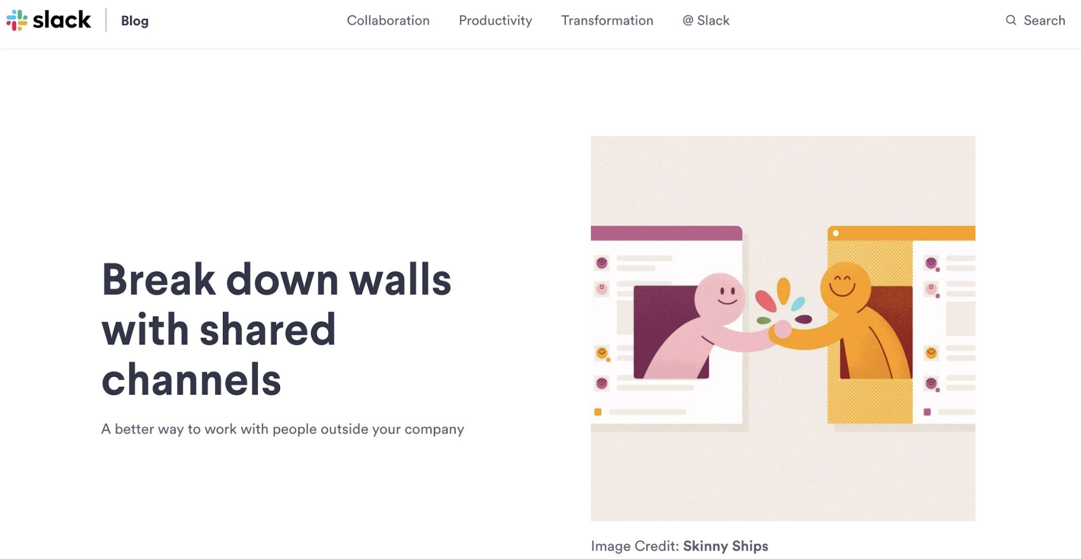
Introducing Slack Enterprise Grid

A new product from Slack to power work across large organizations

https://slackhq.com/introducing-slack-enterprise-grid

# Database at Slack: past

## What didn’t work/why we changed: unique sharding model



Break down walls with shared channels

A better way to work with people outside your company

Image Credit: **Skinny Ships**

https://slackhq.com/slack-shared-channels

# Database at Slack: past

## Percona Live 18

slack

April 25, 2018

Designing and launching the next-generation database system: from whiteboard to production

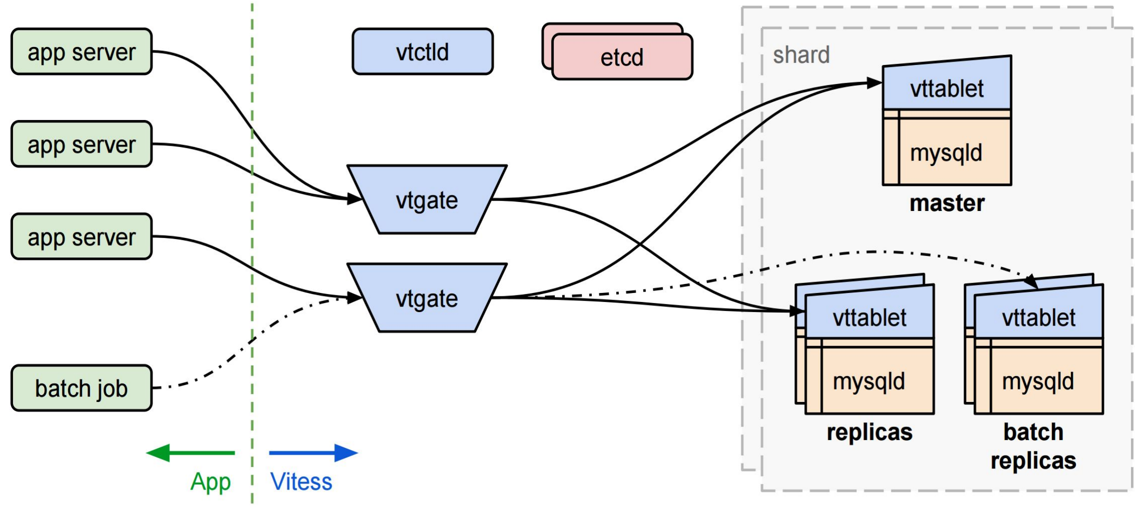
Guido Iaquinti

https://www.percona.com/live/18/sessions/designing-and-launching-the-next-generation-database-system-slack-from-whiteboard-to-production

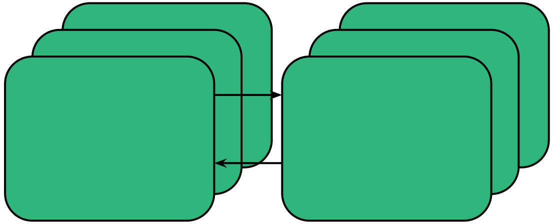
# Database at Slack: present

# Database at Slack: present

## Vitess



## Shards



70% QPS served by Vitess by EOY

# Database at Slack: present

## Stats

- 50+ billion database queries per day
- 7.5 PB of database storage
- Thousand of database servers

# Database at Slack: future

# Database at Slack: future

Consolidate Vitess as our single database solution by mid/end of next year

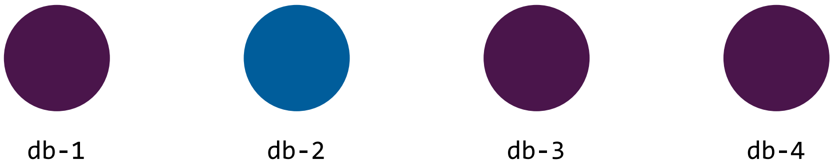
slack

# Running databases in a cloud world

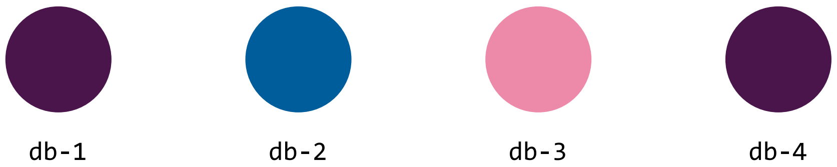
# Variable Infrastructure

db-1
db-2
db-3
db-4

# Variable Infrastructure

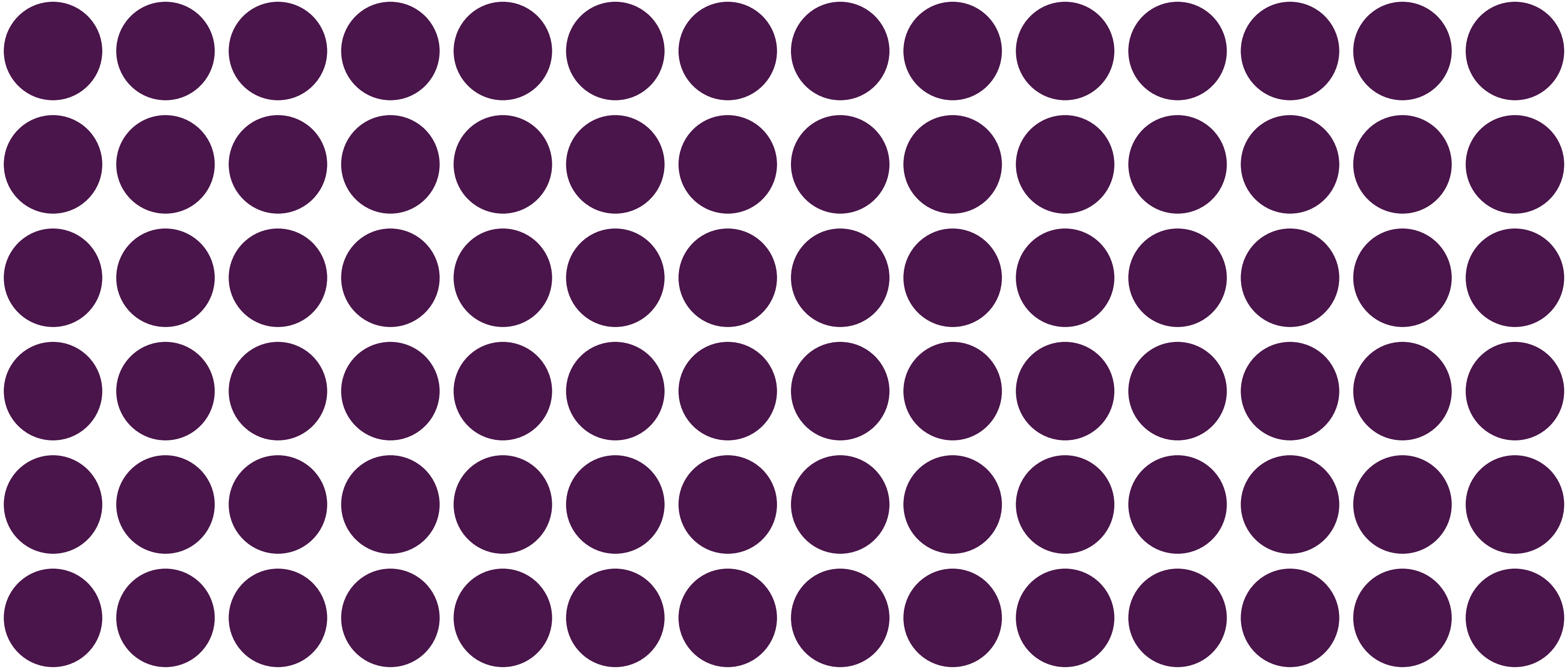


# Variable Infrastructure

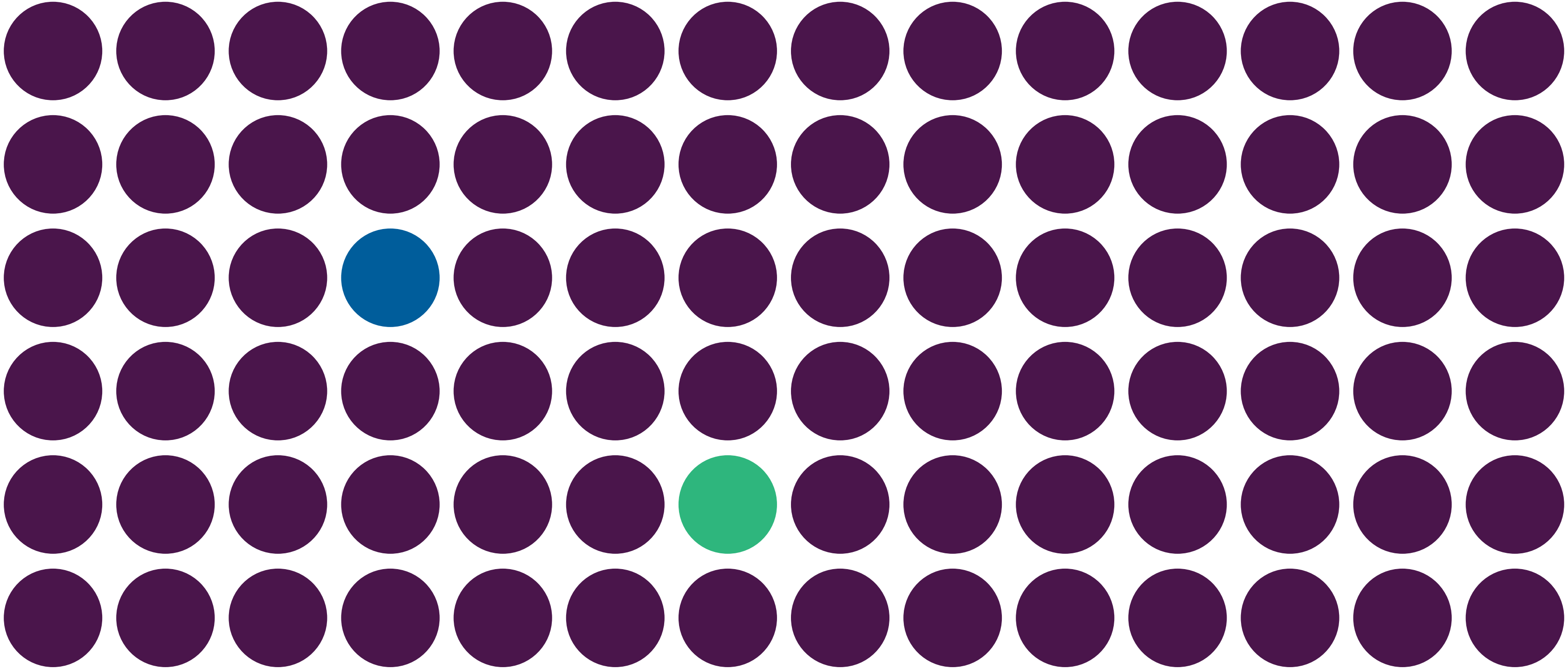


db-1 db-2 db-3 db-4

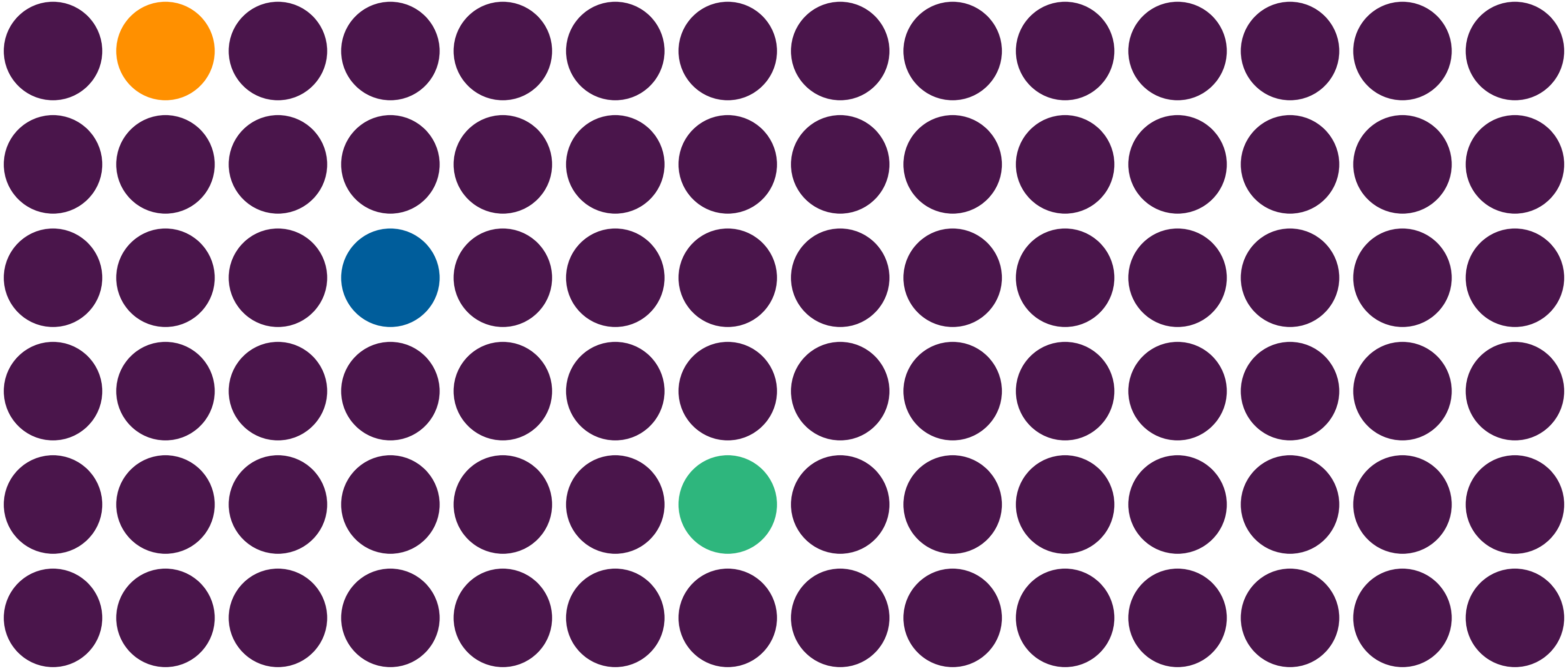
# Variable Infrastructure



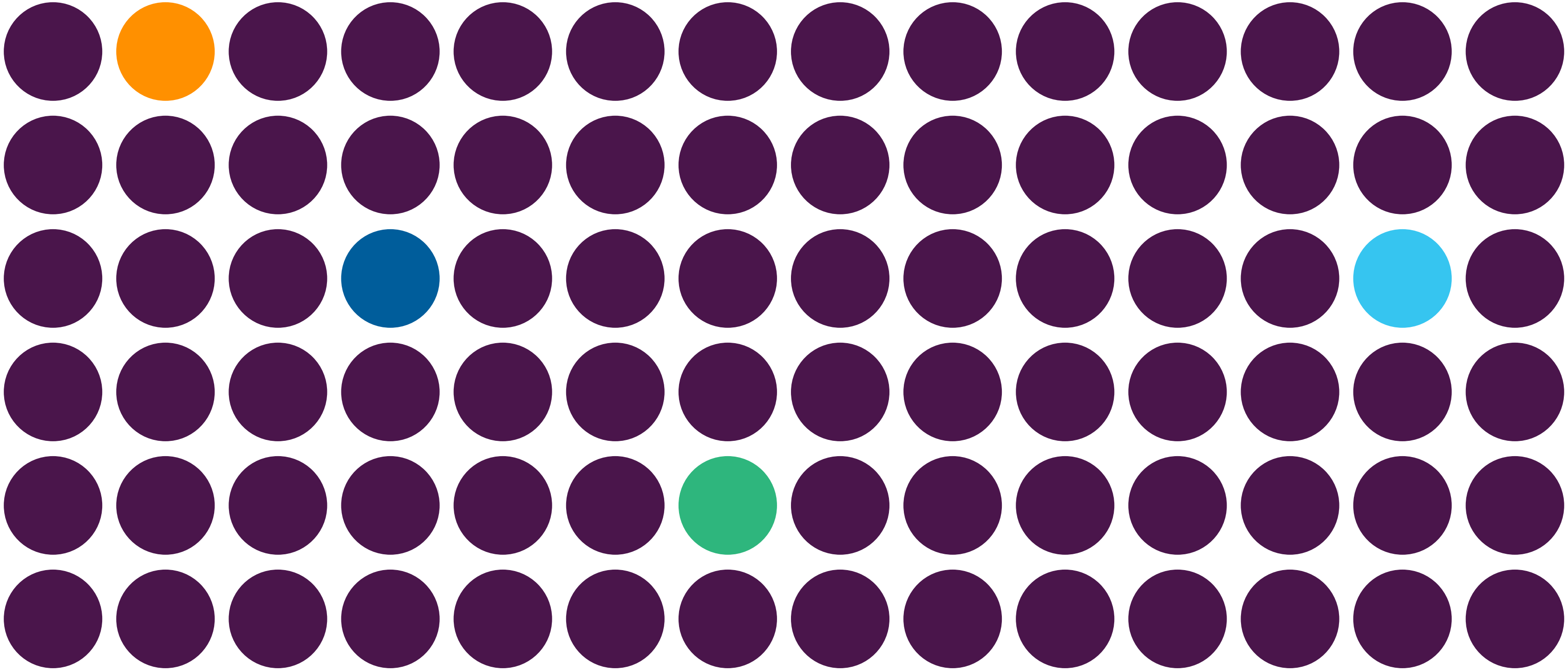
# Variable Infrastructure



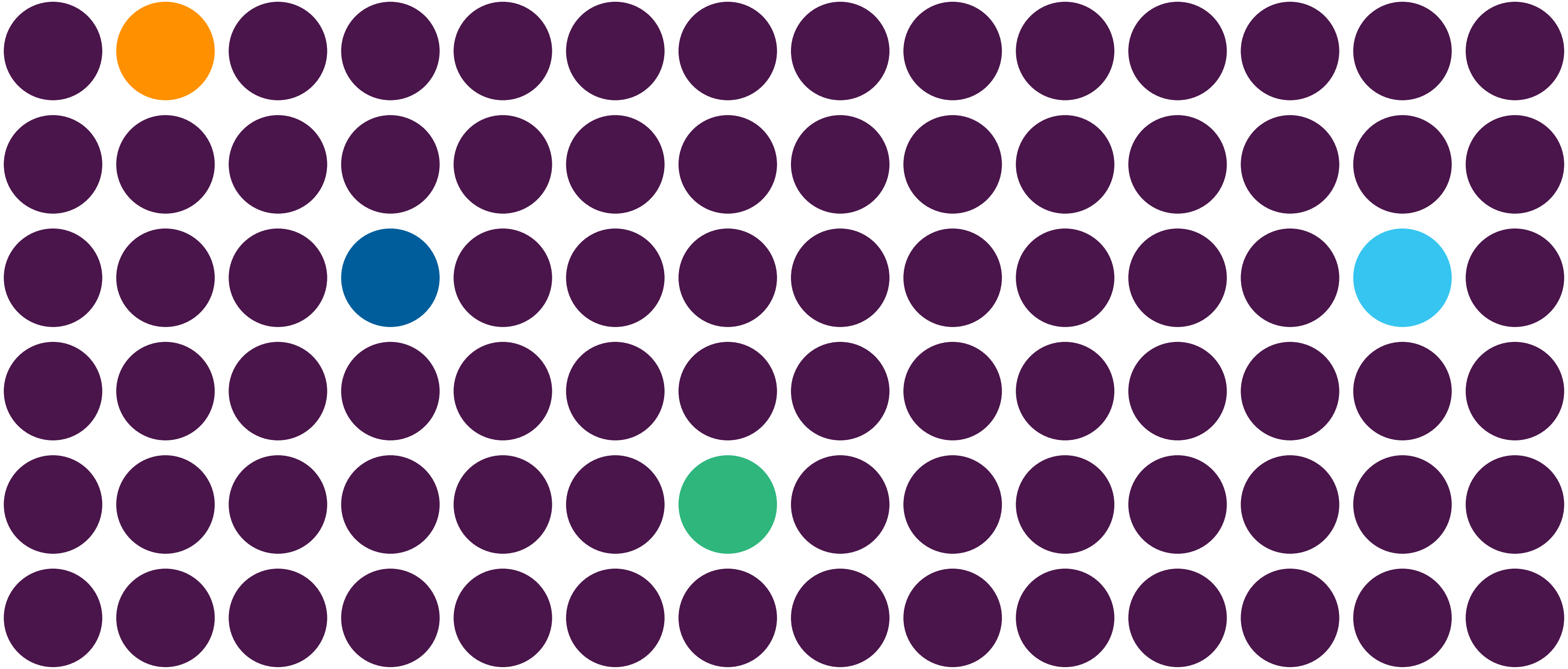
# Variable Infrastructure



# Variable Infrastructure



# Variable Infrastructure



# Immutable Infrastructure

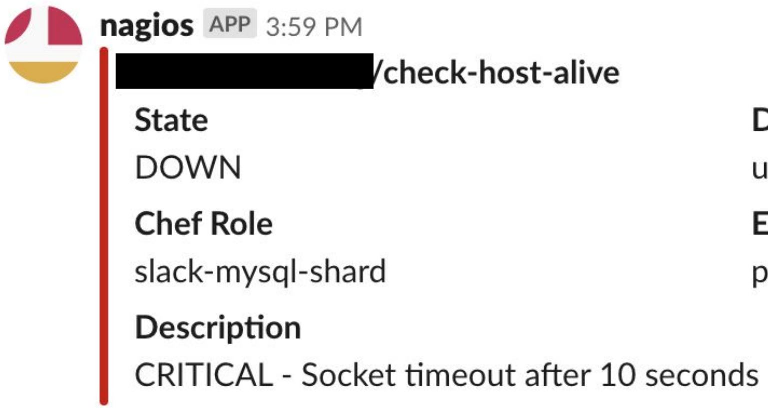
## Manifesto

- Instances are untouched after provisioning
- Configuration changes happen only through reprovisioning
- No in-place patching allowed
- No (real reason to) SSH

# Instance Failures

# Instance Failures

# Instance Failures



nagios APP 3:59 PM

[redacted]/check-host-alive

**State**
DOWN

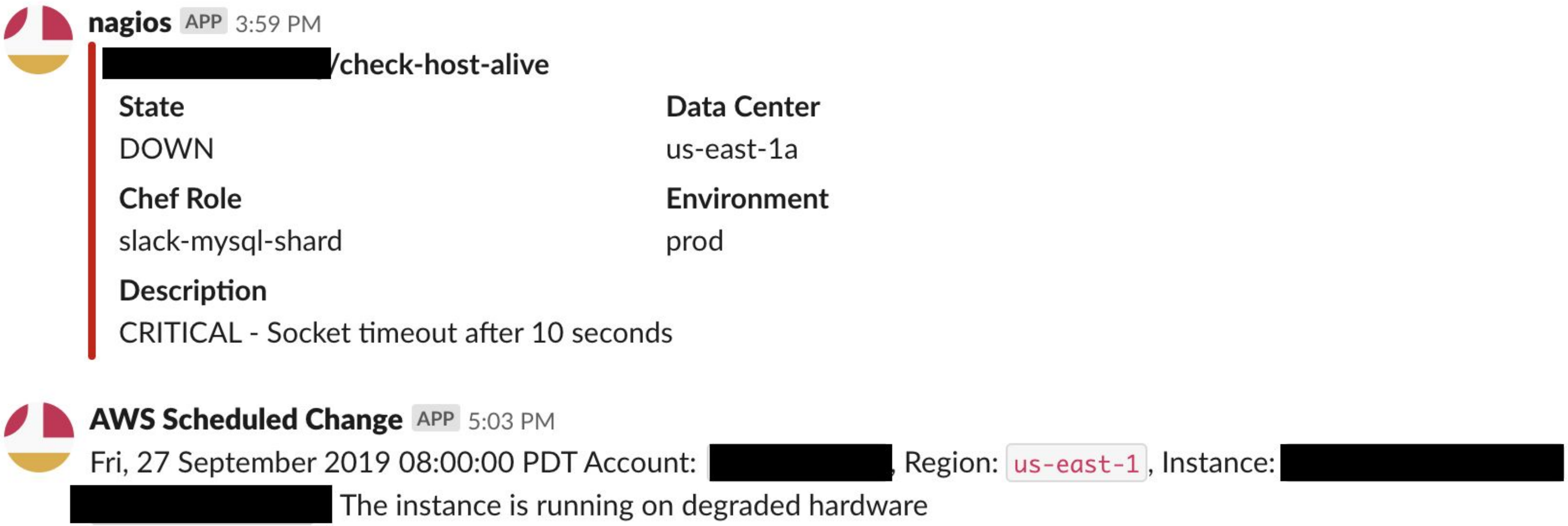
**Data Center**
us-east-1a

**Chef Role**
slack-mysql-shard

**Environment**
prod

**Description**
CRITICAL - Socket timeout after 10 seconds

# Instance Failures



**nagios** APP 3:59 PM

**[redacted]/check-host-alive**

**State**
DOWN

**Data Center**
us-east-1a

**Chef Role**
slack-mysql-shard

**Environment**
prod

**Description**
CRITICAL - Socket timeout after 10 seconds

**AWS Scheduled Change** APP 5:03 PM

Fri, 27 September 2019 08:00:00 PDT Account: [redacted], Region: `us-east-1`, Instance: [redacted] The instance is running on degraded hardware

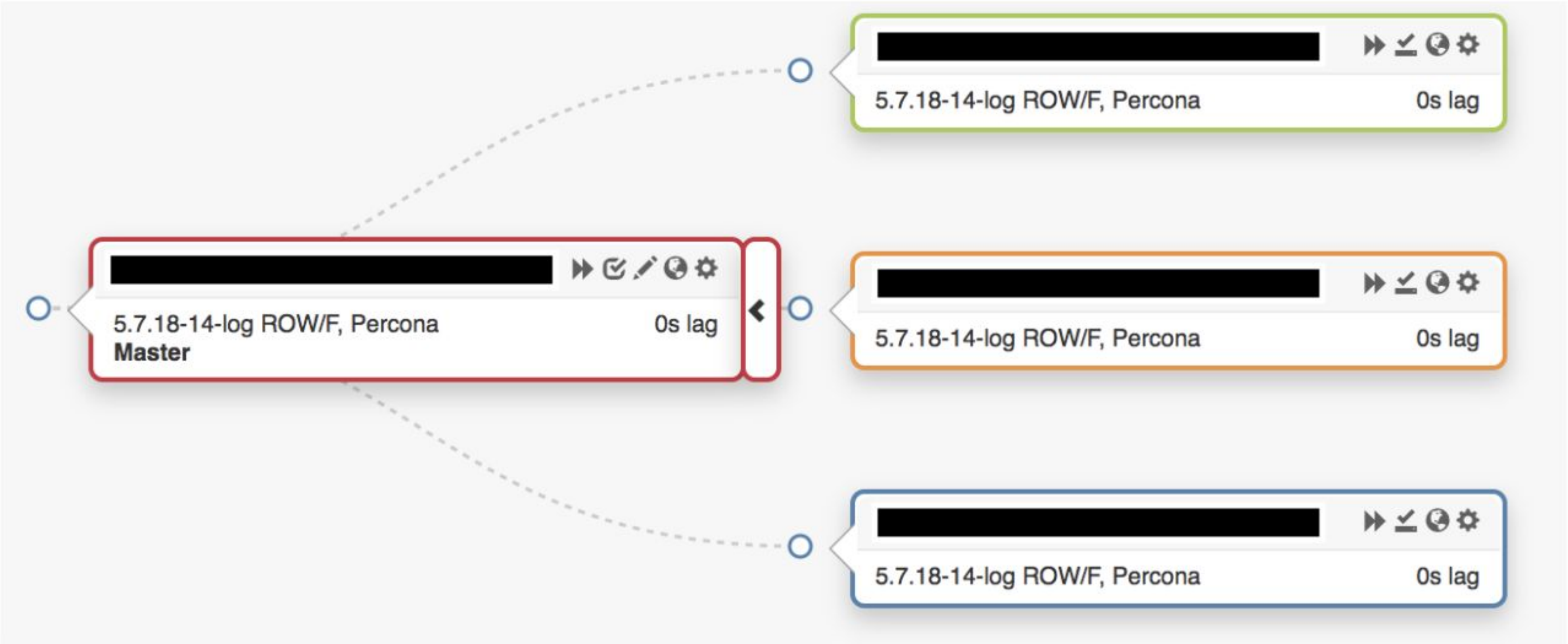
# Instance Failures

- When an instance fails, it can’t be trusted: replace it.
- EBS VS instance store
  - EBS: stop & start instance
  - Instance store: download the latest backup from S3 (NIC is the bottleneck)
- Small shards 👍 VS big shards 👎:
  - Recovery time (if you don’t use EBS)
  - Blast radius
  - Distribute workload VS centralized workload
  - Less contention

# Instance Failures

- When an instance fails, it can’t be trusted: replace it.
- EBS VS instance store
  - EBS: stop & start instance
  - Instance store: download the latest backup from S3 (NIC is the bottleneck)
- Small shards 👍 VS big shards 👎
  - Recovery time (if you don’t use EBS)
  - Blast radius
  - Distribute workload VS centralized workload
  - Less contention

# Durability through replication



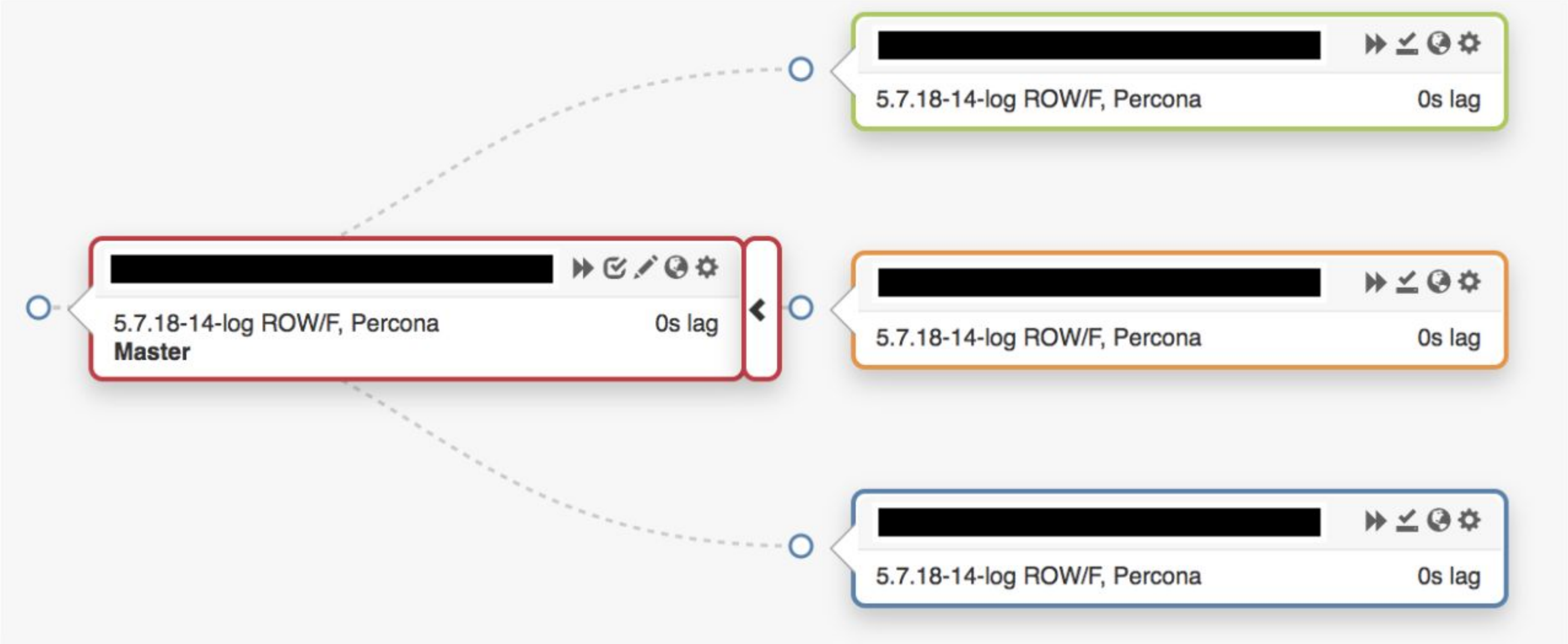
[redacted]
5.7.18-14-log ROW/F, Percona 0s lag
Master

[redacted]
5.7.18-14-log ROW/F, Percona 0s lag

[redacted]
5.7.18-14-log ROW/F, Percona 0s lag

[redacted]
5.7.18-14-log ROW/F, Percona 0s lag

# Durability through replication



[redacted]

5.7.18-14-log ROW/F, Percona 0s lag

**Master**

[redacted]

5.7.18-14-log ROW/F, Percona 0s lag

[redacted]

5.7.18-14-log ROW/F, Percona 0s lag

[redacted]

5.7.18-14-log ROW/F, Percona 0s lag

# Durability through replication

## Semi-sync

- `rpl_semi_sync_master_timeout = 9999999999999`
- `rpl_semi_sync_master_wait_no_slave = 1`

# Durability through replication

### 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`

slack

# Isolation & distribution

# Isolation & distribution

## Design for failures

- External service dependencies
- Minimize blast radius when the unexpected happens
- Noisy neighbours & multi-tenancy
- Let it crash!

# Isolation & distribution

## Design for failures: external service dependencies

- Topology service
- Service discovery
- Failure detection/remediation systems
- Network

# Isolation & distribution

## Design for failures: minimize blast radius

add a GetTopoServer method to srvtopo.Server #3740

update consul api to v1.0.6 #3758

make the resilient topo cache even more resilient and informative #3641

vindex ddl statements #3498

Adds timeout to mysql queries. #3939

Flush binary logs while reparenting #4179

fix a potential blocking issue when StreamHealth fails #3898

Add grpc client knobs #3943

Do not reuse a consul lock #4353

Adds query cache size option to VtGate #3234

Add vtctld health check #3080

Enhanced shard targeting part I #3692

Single round trip commit on BatchExecute #4739

Add grpc window size server config #4037

Refactor region to aliases #4767

Topo to topo compare #4392

4496 topo serving shards refactor #4631

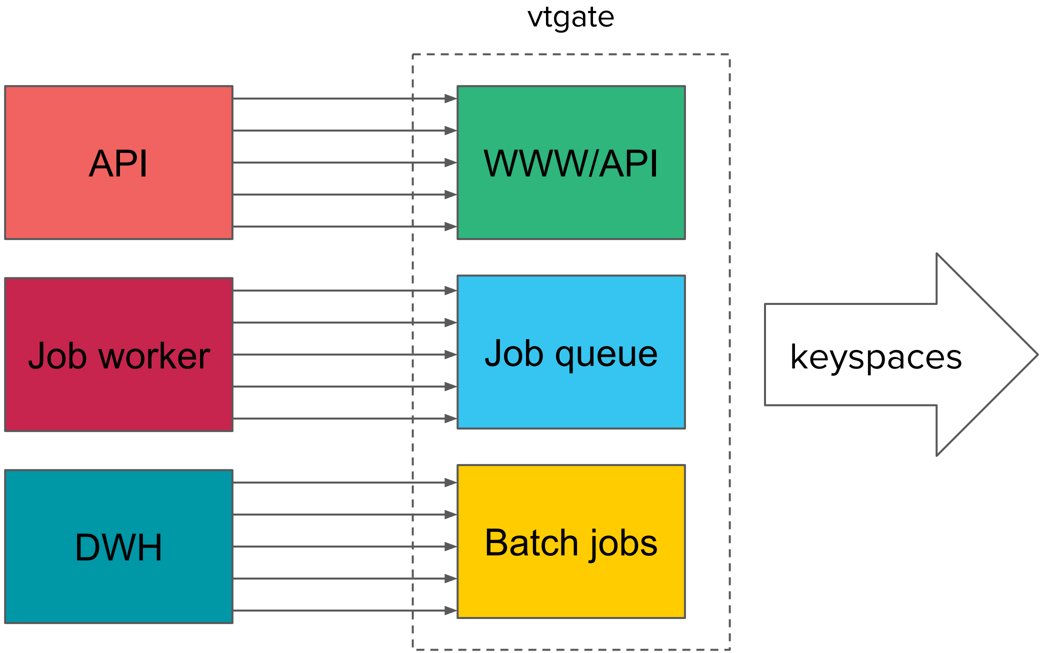
Refactor QueryCache to QueryPlanCache #3345

Do not update replication data if sanity check fails #4102

Adds read and write timeouts to MySQL server #3911

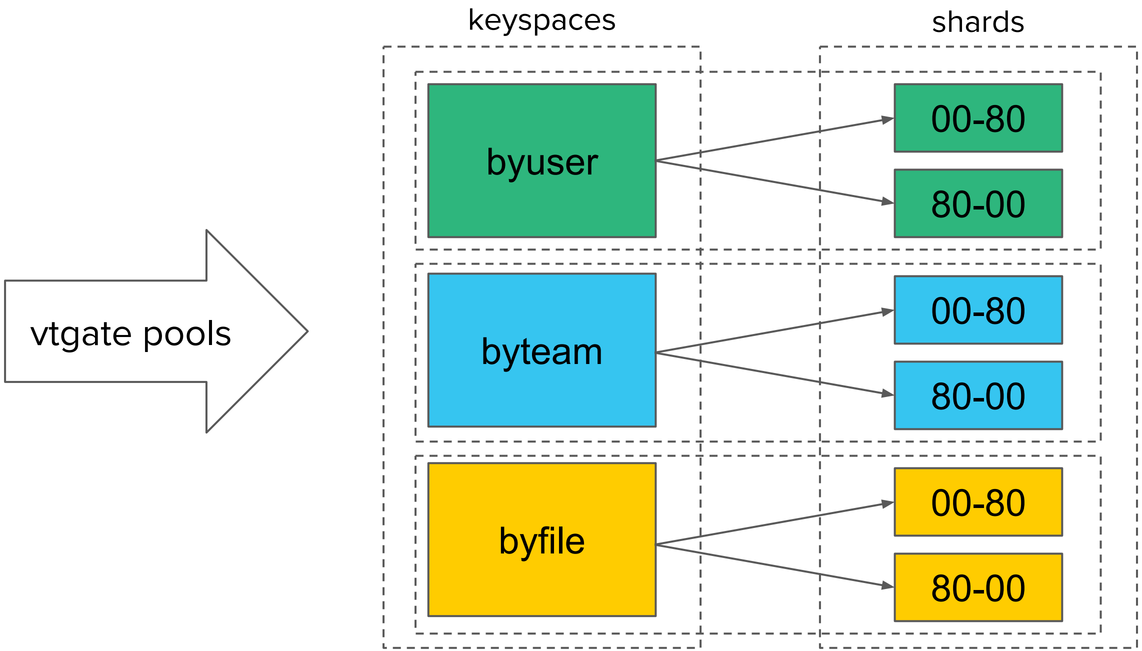
# Isolation & distribution

## Design for failures: noisy neighbours & multi-tenancy



# Isolation & distribution

## Design for failures: noisy neighbours & multi-tenancy



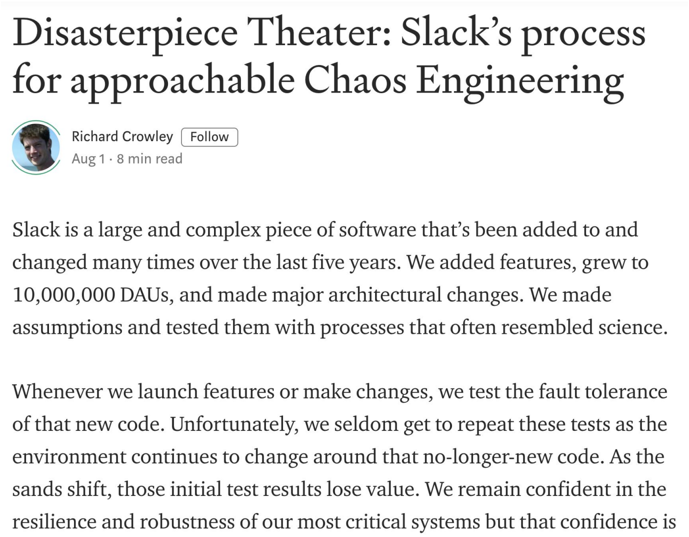
# Isolation & distribution

## Let it crash

- Stateless components in ASG
- Primary/replicas in multiple AZs
- Automated failovers

# Isolation & distribution

## Let it crash



# Disasterpiece Theater: Slack's process for approachable Chaos Engineering

Richard Crowley Follow
Aug 1 · 8 min read

Slack is a large and complex piece of software that's been added to and changed many times over the last five years. We added features, grew to 10,000,000 DAUs, and made major architectural changes. We made assumptions and tested them with processes that often resembled science.

Whenever we launch features or make changes, we test the fault tolerance of that new code. Unfortunately, we seldom get to repeat these tests as the environment continues to change around that no-longer-new code. As the sands shift, those initial test results lose value. We remain confident in the resilience and robustness of our most critical systems but that confidence is

https://slack.engineering/disasterpiece-theater-slacks-process-for-approachable-chaos-engineering-3434422afb54

# Isolation & distribution

## Let it crash

- Topology service failure
- vtctld/vtgate/vttablet/MySQL
  - Complete failure
  - Network partition
- Automated failover system
  - Instance failure
  - Backend failure

slack

# Conclusions

# Conclusions

## Takeaways

- “The cloud is magical but it’s not magic” (Mike Demmer)
- Running databases in a cloud world: immutable infrastructure, instance failures, “durability 2.0”
- Isolation and distribution
- Embrace failures: allow them to happen, just limit their effect

# Conclusions

## Takeaways

- “The cloud is magical but it’s not magic” (Mike Demmer)
- Running databases in a cloud world: immutable infrastructure, instance failures, “durability 2.0”
- Isolation and distribution
- Embrace failures: allow them to happen, just limit their effect

# Conclusions

## Takeaways

- “The cloud is magical but it’s not magic” (Mike Demmer)
- Running databases in a cloud world: immutable infrastructure, instance failures, “durability 2.0”
- Isolation and distribution
- Embrace failures: allow them to happen, just limit their effect

# Conclusions

## Takeaways

- “The cloud is magical but it’s not magic” (Mike Demmer)
- Running databases in a cloud world: immutable infrastructure, instance failures, “durability 2.0”
- Isolation and distribution
- Embrace failures: allow them to happen, just limit their effect

slack

# Thank You!

we are hiring: slack.com/jobs

slack

# Q&A

slack

# Thank You!

we are hiring: slack.com/jobs