Lessons From Building Automation For a Large Distributed Database

Leigh Johnson
Ameet Kotian

October 1, 2019
bit.ly/2ntSVSL
Presenters

Leigh Johnson
(she/her/hers)
Staff DRE, Slack
Google Developer Expert
(Machine Learning)
@grepLeigh

Ameet Kotian
(he/him/his)
Staff DRE, Slack
Past - SRE, Twitter
@ameetkotian
Slack’s mission is to make people’s working lives simpler, more pleasant, and more productive.
Agenda

1. Evolution of Slack Automation
2. Case Study: Self-Healing Databases
3. Lessons
4. Q & A
Evolution of Database Automation

Self-Healing Stateful Systems in 4 Steps

- Monitoring Alerts
- Checklists
- Scripts
- Automated Workflows
Checklists

Runbooks

Shared documents

Lots of hand-over

Limited by team capacity

Just follow these 19 easy steps!
Convert Runbooks to Code
Convert Runbooks to Code
Manual Scripts

Just follow these 5 easy steps!

$ ./provision.sh
$ ./backup_restore.sh
$ ./validate_replication.sh
$ ./service_discovery_stuff.sh
$ ./deprovision_server.sh

Difficult to maintain
Context switching
API contracts
Multi-step process
or...
Write a Do-Everything Script

$ ./fix-it-now.sh

Just follow this 1 easy steps!
Automated Workflows

**Systems (not humans)**

Detects failure mode

Executes appropriate response

Fail Open
You will still need firefighters after installing a fire suppression system. Automation is not a magic bullet.
Automated Workflows are an Investment

Many teams stop automating at the **scripts** stage.

Instrument, monitor, support **n + 1 systems**.
Do Due Diligence

If automation is an **investment** ...

Quantify **value proposition**

Measure **return on investment**
Do Due Diligence

If automation is an **investment**...
Quantify **value proposition**
Measure **return on investment**
Include **qualitative data**
Are you ready to kill your pet databases?
How did we take Slack from...
Presenters

Leigh Johnson
(she/her/hers)
Staff DRE, Slack
Google Developer Expert
(Machine Learning)
@grepLeigh

Ameet Kotian
(he/him/his)
Staff DRE, Slack
Past - SRE, Twitter
@ameetkotian
In January 2019, we reached 10 million DAUs.
Case Study

Building automation for remediating database failures
Goals

Reliably detect MySQL host failures
Goals

**Reliably** detect MySQL host failures

**Automatically** remediate failed MySQL hosts
Goals

**Reliably** detect MySQL host failures

**Automatically** remediate failed MySQL hosts

**Scale** (security fixes, kernel upgrades etc.)
Slack’s database architecture
Slack’s database architecture

Self-manage MySQL on AWS i3 instances

Data is sharded across thousands of hosts

Two main types of clusters - Legacy and Vitess
1. Legacy shard

Application level team-sharded active primary-primary MySQL setup.
1. Legacy shard

Application level team-sharded active primary-primary MySQL setup.

Some shards have read replicas
Strength in Numbers: Slack's Database Architecture.

2nd Oct 2019, 1:30 PM

Guido Iaquinti, Josh Varner
Slack is moving to Vitess

What is Vitess?

- Database solution for MySQL
- Deploy, scale and manage large MySQL cluster
- Built on top of MySQL replication and InnoDB
- MySQL features + scalability of a NoSQL database
- Open source project by YouTube (Google)
- Started in 2010
- Cloud Native Computing Foundation endorsed project
- Ability to run each component in a container
2. Vitess shard

primary-replica MySQL setup

~40% of Slack’s database queries served by Vitess
My First 90 Days with Vitess.

2nd Oct 2019, 11:00 AM

Morgan Tocker

https://vitess.io/
https://vitess.slack.com/
What if there is a failure?

Legacy shard

Vitess shard
Salvage or replace?
Always replace!
Failure detection
Auto remediation requires accurate failure signals.
We use Orchestrator for automatic master failovers for Vitess cluster and it met all the requirements to detect host failures

https://github.com/github/orchestrator
Primary failures

Legacy shard

Vitess shard
Use orchestrator **hooks** triggered on master failovers

```json
"PostMasterFailoverProcesses": [
  "/usr/local/bin/reprovision
  --failed-host {failedHost}
  --failure-type {failureType}
  --failed-shard {failureClusterAlias}
  --count-replicas {countReplicas}"
],
```
Replica failures

Legacy shard

Vitess shard

Optional replicas
Use orchestrator **problems api** to detect replica failures

```python
results = requests.get(self.local_request_api + '/problems/')
for instance in results:
    if ((not instance['Slave_SQL_Running']) or (not instance['Slave_IO_Running'])) and
        ((instance['LastSQLError'] != '') or (instance['LastIOError'] != ''))
        reprovision(instance)
```
Orchestrator can be used to generate accurate failure signals

- Orchestrator is distributed and uses multiple probes to detect failures.
- It uses knowledge of mysql state and replication to detect failures.
- Rich set of APIs
- Has inbuilt concept of a shard
- Downtime/Maintenance mode
Automated Remediation at Scale
Failure Event → Event Handler → Provision Workflow
Automation Components

Task Queue

Scheduler

APIs
Automation Components

- Workflows
- Audit Logs
- Web UI
Failure Event → Event Handler → Provision Workflow

orchestrator

Celery
from boto import ec2

def provision(ec2_options):
    return ec2.create_instances(**ec2_options)

def main():
    ec2_options = {'InstanceType': 'i3.xlarge'}
    provision(ec2_options)
    ...

Manual Script
from boto import ec2
import celery

app = celery.Celery()
@app.task()
def provision(ec2_options):
    return ec2.create_instances(**ec2_options)

ec2_options = {'InstanceType': 'i3.xlarge'}

# distributed execution
request = provision.delay(ec2_options)

# blocks until result is ready
result = request.get()
from celery import chain

@app.task()
def add(x, y):
    return x + y

workflow = chain(
    add.si(2, 2),
    add.s(10)
)
request = workflow.delay()
request.get()
# 14
Celery

Task API

Workflows

```python
from celery import chain

workflow = chain(
    provision_host.si(new_host),
    wait_for_restore.si(new_host),
    service_discovery_remove.si(old_host),
    service_discovery_add.si(new_host),
)

request = workflow.delay()

result = request.get()
```
Celery

Task API
Workflows
Queue Isolation
Rate Limits
Retry Behavior

```python
from boto import ec2
import celery

app = celery.Celery()
app.conf.task_routes = {'provision': {'queue': 'ec2'}}

@app.task(
    autoretry_for=(boto.ClientError),
    retry_backoff=True,
    max_retries=24,
    rate_limit='100/m'
)

def provision(ec2_options):
    return ec2.create_instances(**ec2_options)
```
from boto import ec2
import celery

app = celery.Celery()
app.conf.task_routes = {'provision': {'queue': 'ec2'}}
app.conf.beat_schedule = {
    'prune-every-30-seconds': {
        'task': 'prune',
        'schedule': 30.0
    },
}

@app.task()
def prune():
    ...
Celery

- Task API
- Workflows
- Queue Isolation
- Rate Limits
- Retry Behavior
- Scheduler
- Web UI

Distributed Task Queue Framework

```
pip install celery-flower
```
Distributed Task Queue Framework

Celery

Task API
Workflows
Queue Isolation
Rate Limits
Retry Behavior
Scheduler
Web UI
Slack Notifications

pip install celery-slack-webhooks

github.com/leigh-johnson/celery-slack-webhooks
Distributed Lock

HashiCorp
Consul

Any Strongly Consistent DB

redis
Lessons learned
Three important things - Safety, safety, safety...
Automation software is just like regular software...use the same scalability/reliability principles
Automation software is just regular software... *release early and often*
You need a **rollout** strategy...
And a **rollback** strategy
Show Value Proposition
Commitment to automation
Questions?

We are hiring in DREs, SREs in San Francisco and Dublin locations