Building Blocks of MySQL Automation
Production Engineering

MySQL Infrastructure
Simon Martin
Database Administration at Facebook

Operational scale

- A large number of MySQL hosts, with multiple services per host
- Arranged into a vast number of replicaset of varying composition

Administration scale

- A tiny number of DBAs
- An astonishing number of live promotions every day
- An impressive number dead master promotions
- A depressing number of host replacements
Automate Everything

10X servers,
not 10X DBAs

Humans are slow,
computers are fast
Rome was not built in a day

Many relatively simple blocks

Open source and public knowledge

Independent development

Continuous improvement
Blueprints

Configuration per service

Which roles in which regions

Hardware and versions
Scaffolding
Automated provisioning

- OS Bootstrap
- Chef/Puppet
Scaffolding
Automated provisioning

- OS Bootstrap
- Chef/Puppet

Service Directory

- Fast, scalable reads, atomic writes
- Enabled/disabled for read/write
Foundations
Promotion

- Live promotions - stop writes, re-point slaves, enable writes
- Dead MySQL promotions - recover, promote, replace
- Dead host promotions - most up to date slave?
Foundations
Promotion

- Live promotions - stop writes, re-point slaves, enable writes
- Dead MySQL promotions - recover, promote, replace
- Dead host promotions - most up to date slave?

Hot copy

- Dump and load
- Xtrabackup/MySQL Enterprise
Build It

Disable Service

Promote new master

Enable Service

Replace dead instance

Update Replicaset

Re-provision dead instance
Tidy up the edges
Failed promotions

- Be brittle to do no harm, iterate to make robust

Failed copy operations

- Long running - easier to split
  - Backup
  - Copy
  - Restore
  - Replication
80% done, 80% to go

Track of all the copies

Track all the services

Automate allocation

Relax
What next?

Automated alarm response

- Robots are faster than humans
- Escalate if the remediation fails
- Alarms for humans are dead robots, not dead services
What next?

Automated alarm response

- Robots are faster than humans
- Escalate if the remediation fails
- Alarms for humans are dead robots, not dead services

Proactive

- Predictive errors - demote, replace, send to repair
- Monitor configuration compliance
Scalable visibility

Dedicated health daemon

Routine maintenance
Scalable visibility

Dedicated health daemon

Routine maintenance

Replicaset aware

Voting
What next?

Recovery is too slow

Can not be sure any slave got the last transactions

We have to recover the master before we can promote

MySQL recovery is slow - Rebooting is slower still
Faster Failover
Semisync replication

- Wait for at least one slave to acknowledge binlogs before commit
- Use WebScaleSQL (or 5.7 with rpl_semi_sync_master_wait_point)
- Use local slaves for low latency
- Need 2 or more acknowledgers
Faster Failover
Semisync replication

- Wait for at least one slave to acknowledge binlogs before commit
- Use WebScaleSQL (or 5.7 with rpl_semi_sync_master_wait_point)
- Use local slaves for low latency
- Need 2 or more acknowledgers

Use mysqlbinlog (WebScaleSQL!)

```
--raw --read-from-remote-server --stop-never --to-last-log --use-semisync
```
Binlogs as a Service

- Python Thrift
  - mysqlbinlog

Logtailer Service

Logtailer Service

Logtailer Service

Logtailer Service

Logtailer Service

Logtailer Service

Logtailer Service

Logtailer Service
Binlogs as a Service

Python Thrift
mysqlbinlog

Logtailer Service

Region A
Master
DBStatus

Region B
Slave
DBStatus

Region C
Slave
DBStatus
Look to reuse blocks
Detection is not always easy

▪ Is a host dead if you get a network timeout?
▪ Can not allow commit on old master after we promote
▪ If the host is unreachable we can't even power it off remotely
Look to reuse blocks
Detection is not always easy

- Is a host dead if you get a network timeout?
- Can not allow commit on old master after we promote
- If the host is unreachable we can't even power it off remotely

Node fencing

- With no semisync acknowledgment nothing will commit
- If we can contact all semisync slaves and stop them we can safely failover
Our building blocks

Configurator - Service configuration

SMC - Service Directory

Cyborg/Chef - Automated host provisioning

Dedicated promotion script

MPS - Automated MySQL allocation

DBStatus - Maintenance and fault detection service
From simple building blocks

Fully automated life cycle

New replicaset on demand - grows from pool of spares

Failed service or host recovery <10s

A whole weekend without having to login