MySQL Infrastructure Testing Automation @ GitHub

Tom Krouper, Shlomi Noach
GitHub

Percona Live Europe 2017
Agenda

• About
• MySQL @ GitHub
• Automation
• Backup/restores
• Failovers
• Schema migrations
About me

• Database Infrastructure Engineer
  • aka Sr. DIE
• Working on MySQL since 2003 (MySQL 4.0 release era)
• Worked at Twitter, Booking, and Box previous to GitHub
• Enjoy medium walks on the beach, as long as there aren't too many shells. (This is not to degrade shell scripts, I like those.)

github.com/tomkrouper
@CaptainEyesight
About me

• Infrastructure engineer at GitHub
• Member of the database-infrastructure team
• MySQL community member
• Author of orchestrator, gh-ost, common_schema, freno, ccql and other open source tools.
• Blog at openark.org

github.com/shlomi-noach
@ShlomiNoach
• The world’s largest Octocat T-shirt and stickers store
• And water bottles
• And hoodies
• We also do stuff related to things
• Word is new swag is coming up
GitHub

- 66M repositories
- 24M developers
- 117K businesses
- More than a million teams
- World’s largest open source hosting
- Alexa top 100
- Critical path in build flows
MySQL at GitHub

- GitHub stores repositories in git, and uses MySQL as the backend database for all related metadata:
  - Repository metadata, users, issues, pull requests, comments etc.
  - Website/API/Auth/more all use MySQL.
- We run a few (growing number of) clusters, totaling over 100 MySQL servers.
- The setup isn’t very large but very busy.
MySQL at GitHub

- Our MySQL servers must be available, responsive and in good state
- GitHub has 99.95% SLA
- Availability issues must be handled quickly, as automatically as possible.
github/database-infrastructure

- @ggunson, @jessbreckenridge, @jonahberquist, @shlomi-noach, @tomkrouper
- (We’re growing!)
- We’re concerned with:
  - Data availability
  - Data integrity
Testing
Backups/restores

that ^
Your data

It’s important
Restores

- Dedicated restore servers.
- One per cluster.
- Continuously restores, catches up with replication, restores, catches up with replication, restores, …
- Sending a “success” event at the end of each cycle.
- We monitor for number of “success” events in past 24-ish hours, per cluster.
auto-restore replicas

production replicas

master

backup replica

auto-restore replica
Restores

- New host provisioning uses same flow as restore.
- A human may kick a restore/reclone manually.
- Chatops:
  .mysql backup-restore -H restore.this.host -r
Restore failure

- A specific backup/restore may fail because of computers.
- No reason for panic.
  - Previous backup/restore proven to be working
  - At most we lose time
- Two sequential failures, or failures across clusters are incidents to be investigated
Restore: delayed replica

- One delayed replica per cluster
  - Lagging at 4 hours
- Chatops: `.mysql panic`
Failovers

^ that, too
MySQL setup @ GitHub

- Plain-old single writer master-replicas asynchronous replication.
- Not yet semi-sync
- Cross DC, multiple data centers
- 5.7, RBR
- Servers with special roles: production replica, backup, auto-restore, migration-test, analytics, ...
- 2-3 tiers of replication
- Occasional cluster split (functional sharding)
- Very dynamic, always changing
Points of failure

- Master failure, sev1
- Intermediate masters failure
orchestrator

- Topology discovery
- Refactoring
- Failovers for masters and intermediate masters
- Open source, Apache 2 license
- [github.com/github/orchestrator](https://github.com/github/orchestrator)
orchestrator failovers @ GitHub

• Automated master & intermediate master failovers for all clusters.
• On failover, runs GitHub-specific hooks
  • Grabbing VIP/DNS
  • Updating server role
  • Kicking services (e.g. pt-heartbeat)
  • Notifying chat
  • Running puppet
Testing cluster

- Dedicated testing cluster in production
- Does not take production traffic
  - “load-test” traffic
- Resembles a production topology:
  - OS, MySQL Versions
  - Data centers
  - Server roles
  - DNS
  - Proxy
- Used for many of our deployment tests
Failover testing

- Multiple times per day:
  - Setup the cluster in desired topology layout
  - Inject failure (kill/block/reject)
  - Wait, expect recovery
  - Check topology:
    - Expect new master, correct DNS changes, replica capacity, …
  - Restore old master from backup
    - (an implicit backup/restore test)
  - “success/failure” event
Failover in production

- We expect < 30s failover
- Intermediate master failover has low impact on subset of users, depending on cluster/DC/server
- Master failover implies outage
- Planned master switchover takes a few seconds
A moment of reflection

GitHub
What builds trust in failovers?

A testing environment?
Chaos testing in production

- First steps into regular testing
- Manual
- Supported by our peers
- Learning, understanding impact
Tests that go wrong

- Many things can go wrong
  - Corrupt replication
  - Invalidated servers
  - Unassigned DNS
- Cleanups
Schema migrations
Is your data correct?

The data you see is merely a ghost of your original data.
gh-ost

• Young. 1yr old.
• In production at GitHub since born.
• Software
• Bugs
• Development
• Bugs
gh-ost testing

• gh-ost works perfectly well on our data
• Tested, re-tested, and tested again
• Full coverage of production tables
gh-ost testing servers

- Dedicated servers that run continuous tests
gh-ost testing replicas

production replicas

master

testing replica

production replicas

master

testing replica
ghost testing

- Trivial \texttt{ENGINE=INNODB} migration
- Stop replication
- Cut-over, cut-back
- Checksum both tables, compare
- Checksum failure: \texttt{stop the world, alert}
- Success/failure: event
- Drop ghost table
- Catch up
- Next table
gh-ost development cycle

• Work on branch
  .deploy gh-ost/mybranch to prod/mysql_role=ghost_testing
• Let continuous tests run
• Depending on nature of change, observe hours/days/more.
• Merge
• Tests run regardless of deployed branch
Conclusion

• Backup & restore
• Failovers
• Schema migrations
Thank you!

Questions?

github.com/tomkrouper
@CaptainEyesight

github.com/shlomi-noach,
@ShlomiNoach