The MySQL Ecosystem
@ GitHub
Sam Lambert
Director of Technology

github.com/samlambert
twitter.com/isamlambert
samlambert.com
THE HUMAN BRAIN IS A WONDERFUL THING. IT STARTS WORKING THE MOMENT YOU ARE BORN, AND NEVER STOPS UNTIL YOU STAND UP TO SPEAK IN PUBLIC.

- GEORGE JESSEL
What is GitHub?

- 130+ TB Git data
- 239 GitHubbers
- 100 Engineers

the best way to build and ship software
the best way to build and ship software

Slave
the best way to build and ship software
What is MySQL's role at GitHub?
What is MySQL’s role at GitHub?

- Janky
- MySQL
- Redis
- Elasticsearch
- Jenkins
- Memcached
What is the Ecosystem?
What is the Ecosystem?

mysqld

everything else
Why is the Ecosystem so important?
MySQL is fine on its own
We can (mostly) trust MySQL
The GitHub Stack
The GitHub Stack

WEB

Ruby on Rails
Powerful application framework. Rapid development.

Javascript
UI goodness

BACKEND

C
Powers a number of backend applications

Puppet
Provisioning

Git
Obviously

DATABASES

MySQL
Main source of truth

Elasticsearch
Indexes all the code, as well as issues and pull requests.

Redis
Resque, cache data.
SELECT DATE_SUB(NOW(), INTERVAL 2 YEAR);
Small number of MySQL hosts
Majority of queries served from one host
Replicas only used for backups and failover
Old hardware
Unscalable
Contention everywhere
Traffic spikes caused query response times to go up.
Time for a change!
Need to move data centers
A chance to update hardware
New Datacenter

HARDWARE
- More hosts
- Faster CPUs
- SSDs

NETWORKING
- 10gb
A chance to tune config
Time to functionally shard
Functional Sharding

- Partition by function
- Split features out
- Isolate tables as a whole
- App specific
- Milage may vary
Large volume single table
Constantly growing
No joins - app refactor
Regression testing is essential
Replaying queries from live
the best way to build and ship software

Go Live
Maintenance window
13 minutes

the best way to build and ship software
Results!

MySQL app mean time
Tooling
Lots to mention
the best way
to build and ship software

Haystack
the best way to build and ship software

def call
    return doc unless prefix

    doc.css('*[name], *[id]').each do |element|
        %w(name id).each do |attribute|
            if element[attribute]
                element[attribute] = element[attribute].sub(/\A(#{prefix})\d*/, prefix)
            end
        end
    end
end
Slow queries
the best way to build and ship software
the best way to build and ship software
Haystack is awesome
Failbotd
Failbotd

- C daemon
- Takes exceptions from the app
- Queues them
- Posts to Haystack
Staff toolbar
the best way to build and ship software
<table>
<thead>
<tr>
<th>START</th>
<th>DURATION</th>
<th>RESULTS</th>
<th>QUERY</th>
</tr>
</thead>
<tbody>
<tr>
<td>45.472ms</td>
<td>0.327ms</td>
<td></td>
<td>lib/github/sad.rb:192 +6 github_ro_264.app.post_name_github_production (˓→github_production)</td>
</tr>
<tr>
<td>79.021ms</td>
<td>0.389ms</td>
<td></td>
<td>lib/github/sad.rb:192 +6 github_ro_264.app.post_name_github_production (˓→github_production)</td>
</tr>
<tr>
<td>80.941ms</td>
<td>0.884ms</td>
<td>1</td>
<td>lib/github/sad.rb:264 +6 github_ro_264.app.post_name_github_production (˓→github_production)</td>
</tr>
<tr>
<td>84.958ms</td>
<td>1.069ms</td>
<td>1</td>
<td>lib/github/sad.rb:264 +6 github_ro_28.app.post_name_github_production (˓→github_production)</td>
</tr>
<tr>
<td>91.287ms</td>
<td>0.844ms</td>
<td>1</td>
<td>lib/github/sad.rb:264 +6 github_ro_28.app.post_name_github_production (˓→github_production)</td>
</tr>
</tbody>
</table>
Chat
~137 remote employees
100+ chat rooms
Hubot
Who is Hubot?
Who is Hubot?

"GitHub, Inc., wrote the first version of Hubot to automate our company chat room. Hubot knew how to deploy the site, automate a lot of tasks, and be a source of fun in the company. Eventually he grew to become a formidable force in GitHub. But he led a private, messy life. So we rewrote him."

the best way to build and ship software
the best way to build and ship software
Open Source
CoffeeScript
the best way to build and ship software
the best way to build and ship software

github.com/hubot-scripts
Super easy to set up
Hubot, on Heroku, with the Slack adaptor

This is a version of GitHub's Campfire bot, Hubot, which is designed to be deployed on Heroku with the Slack adaptor. To deploy your own Hubot, just click the button below:

Deploy to Heroku

License

MIT, just like the upstream Hubot license, so go 🤘.
the best way to build and ship software

github.com/cobyism/hubot-heroku-slack
What can hubot do?
850+ commands
Nearly everything
Part of shipping is adding ChatOps
the best way to build and ship software
ChatOps is core to our culture
You are on your own

```
➜  ~ mysql
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.6.23 Homebrew

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```
Invite your friends

TwP
oh we’re definitely exercising those disks
/graph me -h @collectd.disk_utilization(codesearch2-storage)

grantr
well we can’t hold everything in memory even with 215gb per node. that’s still less than 10% of the index

Hubot
mute

@collectd.disk_utilization(codesearch2-storage*github*)

grantr
we’ll have the same amount of file cache as in the current i2.4xl cluster

grantr
the point of trying 8xls is to see if there’s some virtualization issue causing 2 i2.4xls to perform worse than 1 i2.8xl

sam lambert
Aiming for in memory is expensive and probably not scalable for anyone large
It’s all about context
Context is implicit
Incident response
DDoS
/mitigation enable
the best way to build and ship software
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/mysql analyze</td>
<td>analyze a table</td>
</tr>
<tr>
<td>/mysql archive</td>
<td>start or stop archiving on a MySQL cluster</td>
</tr>
<tr>
<td>/mysql backup</td>
<td>backup a table</td>
</tr>
<tr>
<td>/mysql backup-list</td>
<td>list backups for a table</td>
</tr>
<tr>
<td>/mysql clone</td>
<td>clone a table to the MySQL staging environment</td>
</tr>
<tr>
<td>/mysql cluster</td>
<td>show cluster topology</td>
</tr>
<tr>
<td>/mysql drop</td>
<td>safely drops a mysql table</td>
</tr>
<tr>
<td>/mysql drop-all</td>
<td>drops all eligible tables</td>
</tr>
<tr>
<td>/mysql drop-check</td>
<td>checks if a table is safe to drop</td>
</tr>
<tr>
<td>/mysql drop-list</td>
<td>lists the tables that are ready to be dropped</td>
</tr>
<tr>
<td>/mysql drop-pending</td>
<td>lists the tables that are pending dropping</td>
</tr>
<tr>
<td>/mysql dummy-drop</td>
<td>drops a table from production by renaming it</td>
</tr>
<tr>
<td>/mysql engine</td>
<td>shows output of show engine innodb status on a given host</td>
</tr>
<tr>
<td>/mysql explain</td>
<td>explain a query</td>
</tr>
<tr>
<td>/mysql gather</td>
<td>gathers MySQL diagnostic information</td>
</tr>
<tr>
<td>/mysql log-migration-version</td>
<td>insert the version of a migration into production</td>
</tr>
<tr>
<td>/mysql inno</td>
<td>list queries</td>
</tr>
<tr>
<td>/mysql index-stats</td>
<td>Prints out index stats for the specified table</td>
</tr>
<tr>
<td>/mysql kill</td>
<td>kills queries matching supplied criteria</td>
</tr>
<tr>
<td>/mysql kills</td>
<td>shows victims of pt-kill on given date</td>
</tr>
<tr>
<td>/mysql maintenance</td>
<td>begin and end the maintenance of a mysql node</td>
</tr>
<tr>
<td>/mysql master-swap</td>
<td>returns the binlog position to start replication from a new master</td>
</tr>
<tr>
<td>/mysql mutexes</td>
<td>prints a list of mutex events for a MySQL host</td>
</tr>
<tr>
<td>/mysql nibble</td>
<td>deletes all rows in a table</td>
</tr>
<tr>
<td>/mysql nibble-all</td>
<td>nibble all non nibbled migration tables</td>
</tr>
<tr>
<td>/mysql panic</td>
<td>stops the delayed replica to prevent propagation of data loss</td>
</tr>
<tr>
<td>/mysql pk-pct</td>
<td>shows the % of the primary key range that is used</td>
</tr>
<tr>
<td>/mysql pool</td>
<td>saves or restores the innodb buffer pool of a host</td>
</tr>
</tbody>
</table>
Learn together
github.com/samlambert/hubot-mysql-chatops
Collaborative workspace
A backup job has been created for the yubicats table. You will be notified in +The Database Infrastructure Room when the backup is complete.

[2015-04-13T14:43:0700] --sam@sam.lambert: yubicats has been backed up for you and sent away to the cloud fairies with the file name github-iyorpiisubel.pickle-gate.net-github_production-yubicats-1428959921.sql.gz
mysql clone

samlambert
/mysql clone yubicats

Hubot
Killed queries
Killed queries

| Hubot |
The following queries have been killed in the last minute:

| Info |

| SELECT |
User access
User access

Hubot
probablycorey connected to github.com with console_rw ("analytics") as console_rw via gh-dbus on 02/12/2023
the best way to build and ship software
the best way to build and ship software
the best way to build and ship software
Amen
• Monitoring app
• Set up backends
• Easily gather metrics
• Pushed to Graphme
We have these tools. Let's use them
It’s about the app
Let's talk strategy
All the hardware
Doesn't scale
The best way to build and ship software.

GitHub.com

nginx
unicorn

Read Proxy
haproxy

primary
replica
replica
replica
Use existing patterns
Tuning Haproxy
App failover logic
the best way to build and ship software
/mysqlproxy status

```plaintext

<table>
<thead>
<tr>
<th>Host</th>
<th>sess</th>
<th>peak</th>
<th>status</th>
<th>weight</th>
<th>chkfails</th>
</tr>
</thead>
<tbody>
<tr>
<td>github</td>
<td>54</td>
<td>75</td>
<td>UP</td>
<td>10</td>
<td>18</td>
</tr>
<tr>
<td>github</td>
<td>55</td>
<td>76</td>
<td>UP</td>
<td>10</td>
<td>51</td>
</tr>
<tr>
<td>github</td>
<td>58</td>
<td>75</td>
<td>UP</td>
<td>10</td>
<td>17</td>
</tr>
<tr>
<td>github</td>
<td>55</td>
<td>72</td>
<td>UP</td>
<td>10</td>
<td>26</td>
</tr>
<tr>
<td>github</td>
<td>0</td>
<td>0</td>
<td>DRAIN</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>github</td>
<td>0</td>
<td>0</td>
<td>DRAIN</td>
<td>0</td>
<td>336</td>
</tr>
<tr>
<td>github</td>
<td>54</td>
<td>75</td>
<td>UP</td>
<td>10</td>
<td>18</td>
</tr>
<tr>
<td>github</td>
<td>0</td>
<td>0</td>
<td>DRAIN</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>github</td>
<td>0</td>
<td>0</td>
<td>DRAIN</td>
<td>0</td>
<td>27</td>
</tr>
<tr>
<td>github</td>
<td>56</td>
<td>75</td>
<td>UP</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>github</td>
<td>0</td>
<td>0</td>
<td>DRAIN</td>
<td>0</td>
<td>5</td>
</tr>
</tbody>
</table>
```
/mysqlproxy disable
/mysqlproxy enable
App refactor
Send all GET requests to a replica
Stick to master for a second after POST
How do you refactor these changes?
Some GETs make writes
Set up write alert connection
Write alerts go to Haystack
Get the stack trace
Gradual iteration over the codebase
Gitauth
Single host

the best way to build and ship software
the best way to build and ship software
Now we are sensitive to delay
Delay!!

The best way to build and ship software.
What is happening?
Delay!!
Ok, so deletes are happening
Attack on two fronts
Performance schema
```
@client.query "SELECT object_name AS table_name, \
  LOWER(index_name) as index_name,\n  count_fetch AS select_count,\n  count_insert AS insert_count,\n  count_update AS update_count,\n  count_delete AS delete_count\nFROM performance_schema.table_io_waits_summary_by_index_usage\nWHERE object_schema = 'github_production'\nAND object_name NOT REGEXP '^_.'", (err, results) =>
  if err
    log 'error', err
  else
    prefix = "#{metric}.table_stats.waits"
    for row in results
      @gauge "#{prefix}.#{row.table_name}.#{row.index_name}.select.count", parseInt(row.select_count)
      @gauge "#{prefix}.#{row.table_name}.#{row.index_name}.insert.count", parseInt(row.insert_count)
      @gauge "#{prefix}.#{row.table_name}.#{row.index_name}.update.count", parseInt(row.update_count)
      @gauge "#{prefix}.#{row.table_name}.#{row.index_name}.delete.count", parseInt(row.delete_count)
```
Haystack
Affected rows bucket
Changed 9665 rows
DELETE FROM `commit_contributions` WHERE ((repository_id = ))

<table>
<thead>
<tr>
<th>class_name</th>
<th>QueryAffectedRowsLogger::QueryAffectedRows</th>
</tr>
</thead>
<tbody>
<tr>
<td>current_ref</td>
<td>master</td>
</tr>
<tr>
<td>network_id</td>
<td></td>
</tr>
<tr>
<td>query</td>
<td>DELETE FROM <code>commit_contributions</code> WHERE ((repository_id = ))</td>
</tr>
<tr>
<td>queue_time</td>
<td>0.09425826991015625</td>
</tr>
<tr>
<td>queued_from</td>
<td>Api::Repos#DELETE</td>
</tr>
<tr>
<td>rails</td>
<td>3.0.20.github12</td>
</tr>
<tr>
<td>request_id</td>
<td></td>
</tr>
<tr>
<td>rescue_args</td>
<td>23807443, nil, <code>request_id</code></td>
</tr>
<tr>
<td>rescue_job</td>
<td>GitHub::Jobs::RepositoryDelete</td>
</tr>
<tr>
<td>ruby</td>
<td>ruby 2.1.2p85-github-tcmalloc (71c08a81c0) [x86_64-linux]</td>
</tr>
<tr>
<td>server</td>
<td></td>
</tr>
</tbody>
</table>

GitHub-query-affected_rows / QueryAffectedRowsLogger::QueryAffectedRows
October 16, 2014 08:35:22
samlambert mentioned in Repository deletes create replication lag
October 16, 2014 08:47:39
Haystack

samlambert mentioned in Repository deletes create replication lag
October 16, 2014 08:47:39
Delete wait volume per table (Top 10) by most deviant

commit_contributions
languages
issue_events
user_sessions
reflog_entries
issues
issue_comments
abilities
statuses
labels

the best way to build and ship software
Too many affected rows
Contributions

Summary of Pull Requests, issues opened, and commits. Learn more.

Contributions in the last year

4,026 total
Apr 14, 2014 – Apr 14, 2015

Longest streak

223 days
May 11 – December 19

Current streak

0 days
Last contributed a day ago

the best way to build and ship software
Dependent destroy
Problem found
Throttler
Github::Throttler

```ruby
# Public: throttle based on replication delay
#
# Waits until replication delay is at or below ALLOWABLE_DELAY, then yields.
def throttle
  GitHub.stats.increment "replication-delay.called"
  slept = 0
  total_slept = 0

  loop do
    begin
      hostname, delay = maximum_replication_delay

      if delay <= ALLOWABLE_DELAY
        return yield
      end

      log "#{hostname} delay #{delay} > #{ALLOWABLE_DELAY}, sleeping"

      sleep ALLOWABLE_DELAY
      slept += ALLOWABLE_DELAY
      total_slept += ALLOWABLE_DELAY
      GitHub.stats.increment "replication-delay.slept"

      if slept >= REPLICA_RESET_TIME
        GitHub.stats.increment "replication-delay.reload-indices"
        reset_replicas
        slept = 0
      end
    end
  end
end
```
def self.clear_contributions(repo)
repo.commit_contribution_ids.each_slice(DELETE_BATCH_SIZE) do |slice|
  throttle do
    GitHub::SQL.run <<SQL, :ids => slice
      DELETE FROM commit_contributions WHERE id IN :ids
    end
  end
end
Problem solved
We are hiring!!
the best way to build and ship software

git.io/vv8yi
Roadmap

- GTIDs
- MySQL 5.7
- Cluster aware monitoring
- Failover improvements
- So much more
Questions?
Sam Lambert
Director of Technology

github.com/samlambert
twitter.com/isamlambert
samlambert.com