



YesSQL, Process and Tooling at Scale

Rocio Delgado
Universe 2016

whoami?

 **@rokkzy**

 **@rocio**



A Relational Model of Data for Large Shared Data Banks

E. F. Codd

IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network models of the data. In Section 1, inadequacies of these models are discussed. A model based on n -ary relations, a normal form for data base relations, and the concept of a universal data sublanguage are introduced. In Section 2, certain operations on relations (other than logical inference) are discussed and applied to the problems of redundancy and consistency in the user's model.

KEY WORDS AND PHRASES: data bank, data base, data structure, data organization, hierarchies of data, networks of data, relations, derivability, redundancy, consistency, composition, join, retrieval language, predicate calculus, security, data integrity

CR CATEGORIES: 3.70, 3.73, 3.75, 4.20, 4.22, 4.29

The relational view (or model) of data described in Section 1 appears to be superior in several respects to the graph or network model [3, 4] presently in vogue for non-inferential systems. It provides a means of describing data with its natural structure only—that is, without superimposing any additional structure for machine representation purposes. Accordingly, it provides a basis for a high level data language which will yield maximal independence between programs on the one hand and machine representation and organization of data on the other.

A further advantage of the relational view is that it forms a sound basis for treating derivability, redundancy, and consistency of relations—these are discussed in Section 2. The network model, on the other hand, has spawned a number of confusions, not the least of which is mistaking the derivation of connections for the derivation of relations (see remarks in Section 2 on the “connection trap”).

Finally, the relational view permits a clearer evaluation of the scope and logical limitations of present formatted data systems, and also the relative merits (from a logical standpoint) of competing representations of data within a single system. Examples of this clearer perspective are cited in various parts of this paper. Implementations of systems to support the relational model are not discussed.

1.2. DATA DEPENDENCIES IN PRESENT SYSTEMS

The provision of data description tables in recently developed information systems represents a major advance toward the goal of data independence [5, 6, 7]. Such tables facilitate changing certain characteristics of the data representation stored in a data bank. However, the variety of data representation characteristics which can be changed *without logically impairing some application programs* is still quite limited. Further, the model of data with which users interact is still cluttered with representational properties, particularly in regard to the representation of collections of data (as opposed to individual items). Three of



YES SQL



One system fits all?



~~One system fits all~~



16+ M Users



125+ M Issues



38+ M Repos



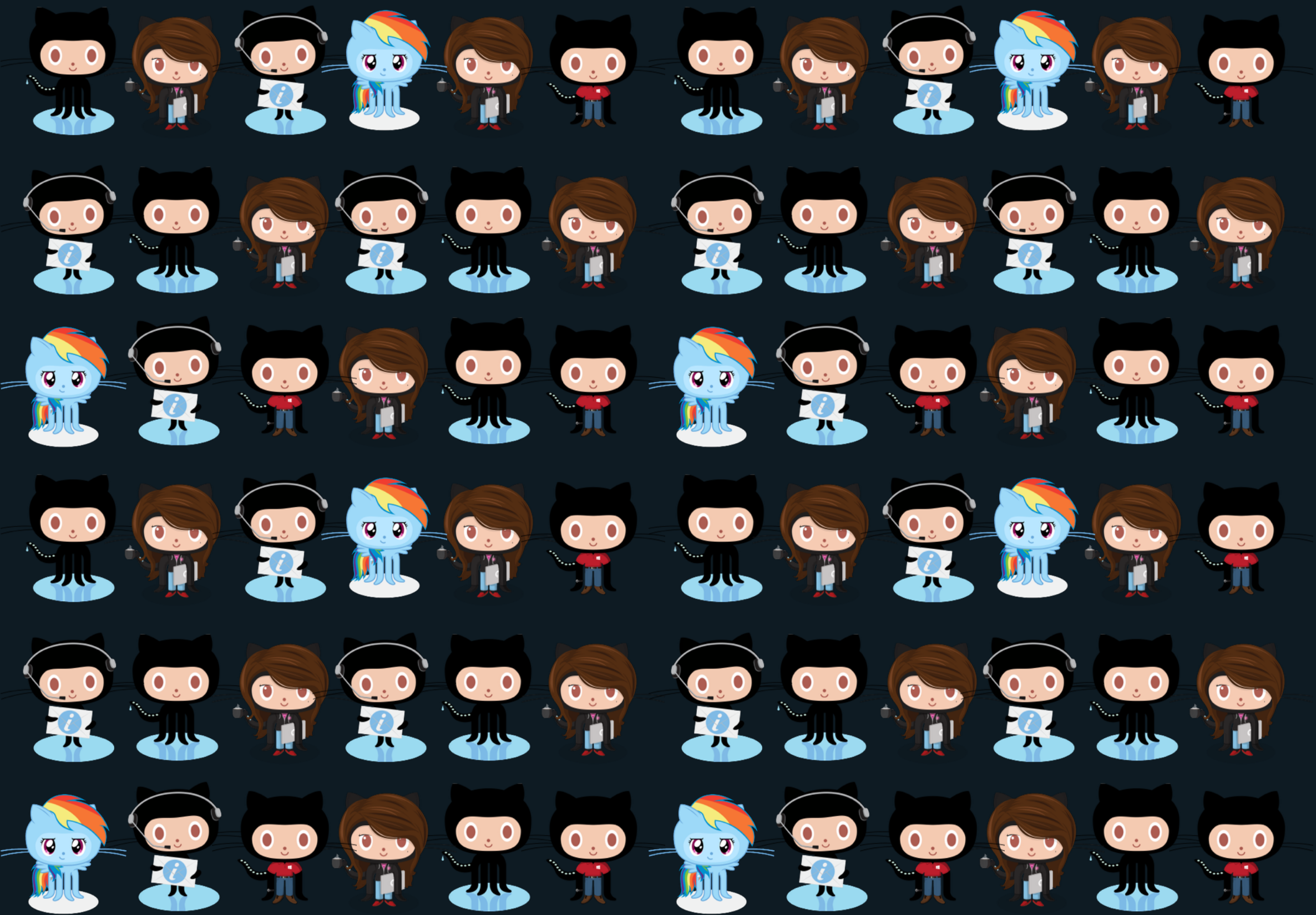
78+ M Pull Requests*

* Since 2010

Team Growth



Team Growth





Performance as a Feature

Why care about performance?



Performance Culture



**Everybody's
responsibility**



Process &
Tooling



Metrics



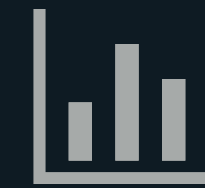
Performance Culture



Everybody's
responsibility

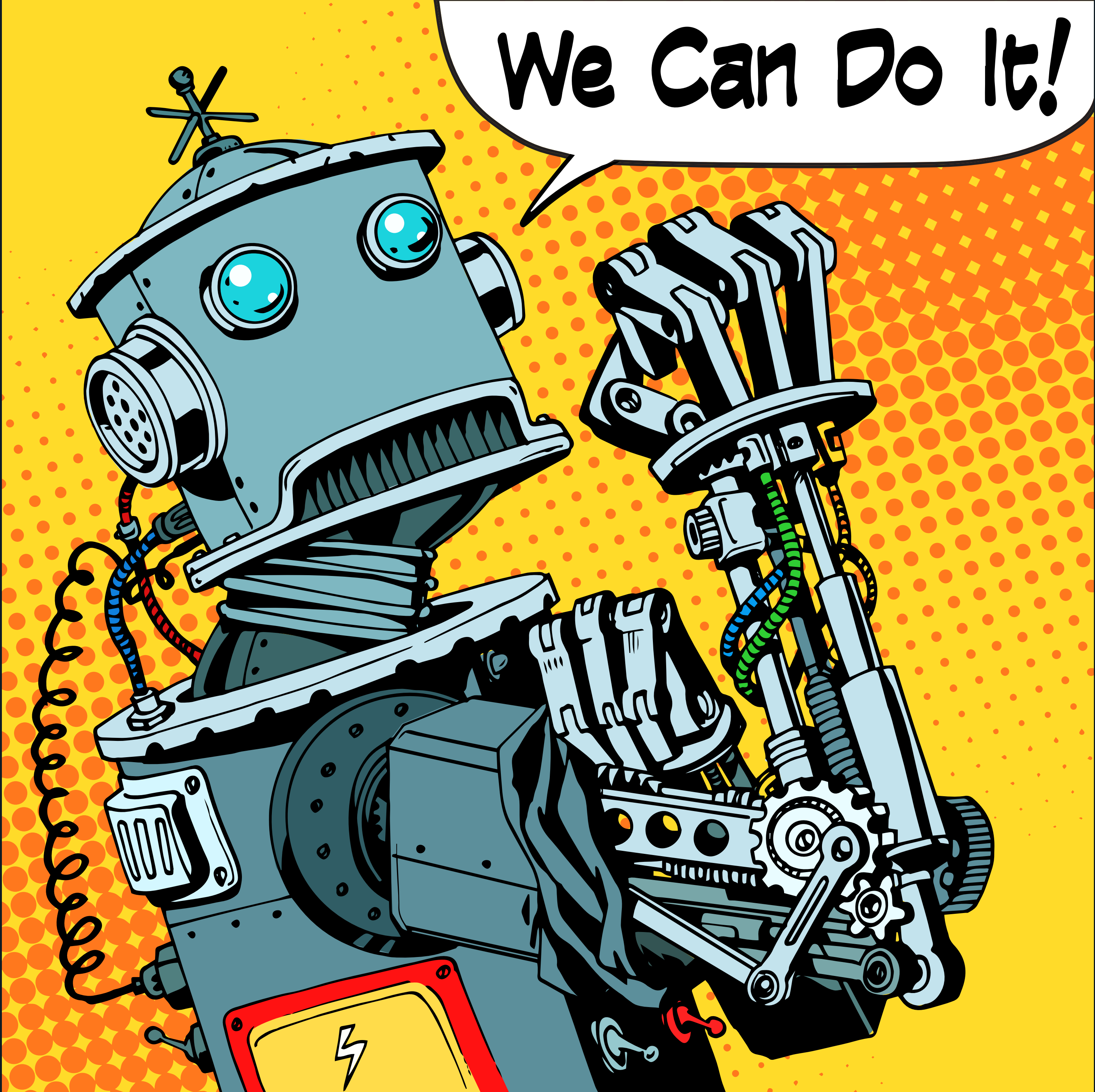


**Process &
Tooling**



Metrics

We Can Do It!

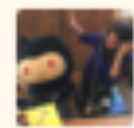




I HAVE
FOUND
THE
THINGS

HU-BOT

Process & Tooling



rocio

.graph me -1d..now substr(highestAverage(github.browser.{chrome,firefox,ie,safari}.performance.crossBrowserLoad.median,10000),2,3) + title=per-browser+page+load+time+(median)



hubot BOT

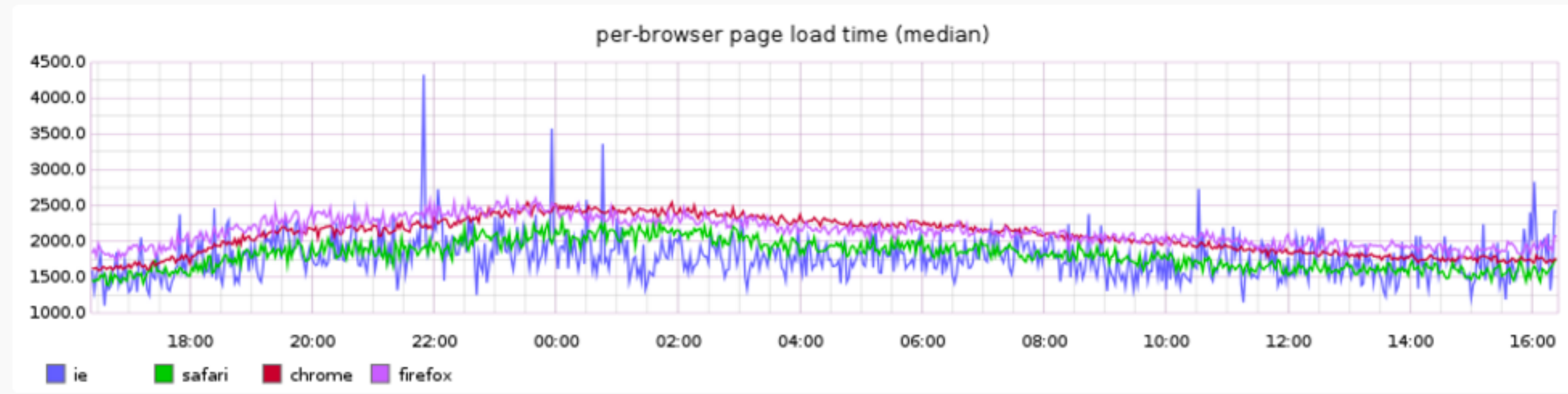
Graph Me

-1h

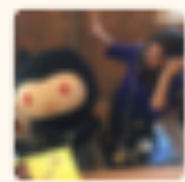
-8h

-24h

-1w



Process & Tooling



rocio

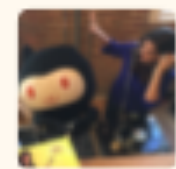
`.mysql table-sizes`



hubot BOT

usage: `.mysql table-sizes <cluster|hostname> [schema] [table-name-pattern]`
prints out the sizes of all the tables

Deploying code



rocio

2:09 PM

.deploy github/index_labels to prod



@rocio: Looks like queue-prep-index_labels-c39235d-00ea35a-1224d27 is green and equivalent to your branch; deploying. 2:09 PM

[github:index_labels]: rocio pushed 1 commit

777859c: Auto-merged master into index_labels on deployment – rocio

@rocio is deploying github/index_labels (f7fdff3...9586425) to production. Check out the deployment confidence dashboard or haystack firehose so you're the first to know if you've broken anything.



+



YOU GET A DATABASE AND YOU GET A DATABASE

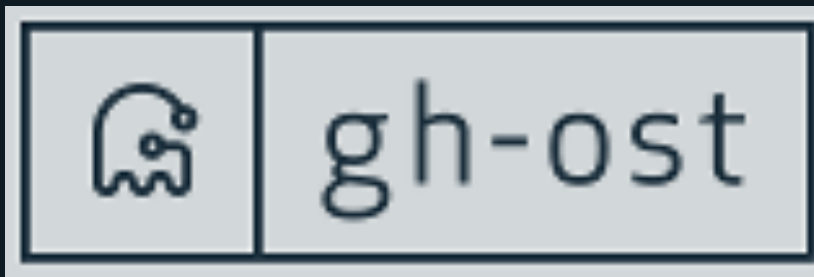


EVERYONE GETS A DATABASE!



GitHub's **O**nline **S**chema

Transmogri~~f~~ier
Transfiguration
Transformer
Thingy



- * Triggerless
- * Lightweight
- * Pauseable
- * Dynamically controllable
- * Auditable
- * Testable
- * Trustable

Process & Tooling



rocio

`.migration-queue`

7:32



hubot BOT

7:32

Usage: `.migration-queue <command> [<args>...]`

Workflow chatops for database migrations

- | | |
|--|-----------------------------------|
| <code>add <pr-number></code> | - Add a migration to the queue. |
| <code>schedule <pr-number> <schedule></code> | - Schedule a migration to be run. |
| <code>show [needs_review reviewed scheduled]</code> | - Show all migrations in queue. |
| <code>completed <pr-number> <migration-version></code> | - Mark a migration as completed. |

Process & Tooling



mikemcquaid

.migration-queue add <https://github.com/github/github/pull/60574>



hubot BOT

[github/github#60574](#) needs scheduling for migration.

Process & Tooling



rocio

.migration-queue show

8:28 AM



hubot BOT

Needs review:

- [github/github#60834](#): New index on labels (`repository_id`, `name`)
- [github/github#60754](#): migrations for outdated column on pull request review comments
- [github/github#60574](#): Add user_hidden and privacy indices in migration to Issues.
- [github/github#60573](#): Add user_hidden indices in migration to Issues.
- [github/github#60521](#): Migration: Create organization_external_identities table
- [github/github#60327](#): [WIP] Add migration for privacy indexes in issues table
- [github/github#60170](#): Transition Review Comment State from Null to Submitted
- [github/github#60162](#): Migration for PullRequestReviewComment default state submitted

Process & Tooling



hubot commented 7 days ago



This migration was scheduled by @github/database-infrastructure to be run at right now.

@github/database-infrastructure: Running migrations in read-only mode to generate DDLs...

To run the migration:

```
gh-migrate-ghost -c mysql1 -t repositories -d 'ADD INDEX index_repositories_organization_active_public (organization_id, active, public)'
```


Once the migration is complete:

```
.migration-queue completed 60753 20160829175757
```




hubot added **DB migration - scheduled** and removed **DB migration - ready** labels 7 days ago

Process & Tooling

 **rocio**

.migration sup

 **hubot** BOT

7:45 PM

Copy: 76390800/158038281 48.3%; Applied: 281395; Backlog: 4/100; Time: 6h46m45s(total), 6h46m44s(copy); streamer: mysql-bin.011093:87270693; State: throttled, lag=0.565436s; ETA: 7h14m44s





A process alone won't solve all the problems.

Communication



<https://www.flickr.com/photos/wocintechchat/>

Collaboration



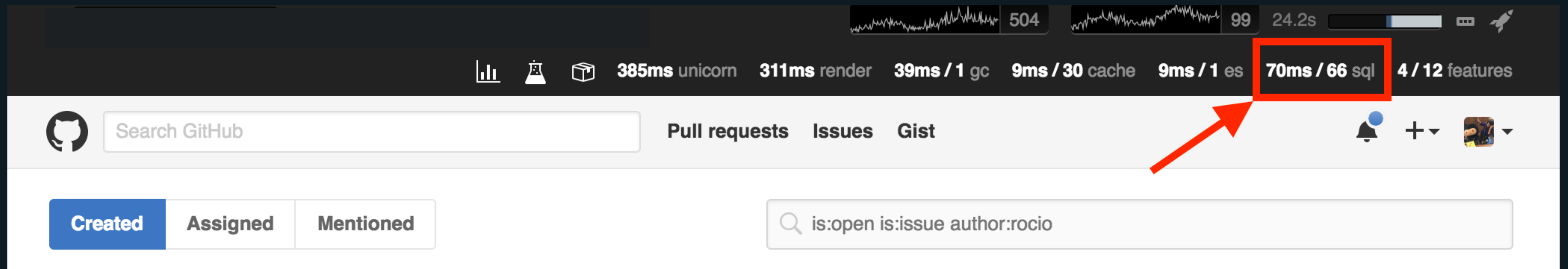
Tooling








The background is a dark navy blue space filled with various celestial elements. In the top-left corner, there is a large orange sun with rays and a small red planet. The top-right corner features a cluster of stars, including a blue four-pointed star and several orange and grey dots. The bottom-left corner shows a blue planet with horizontal stripes, a small purple planet, and a blue four-pointed star. The bottom-right corner contains a blue planet, a pink comet with a yellow tail, and a purple star. Numerous small white dots of varying sizes are scattered throughout the background, representing distant stars.

Speaking of tools...

Process & Tooling - Peek




Process & Tooling - Peek

ne	9ms / 1 es	<u>70ms / 66</u> sql	4 / 12 features
77		AR objects	
			    
40		Label	
26		Issue	
4		User	
2		Organization	
2		Repository	
1		TwoFactorCredential	
1		Survey	
1		UserSession	

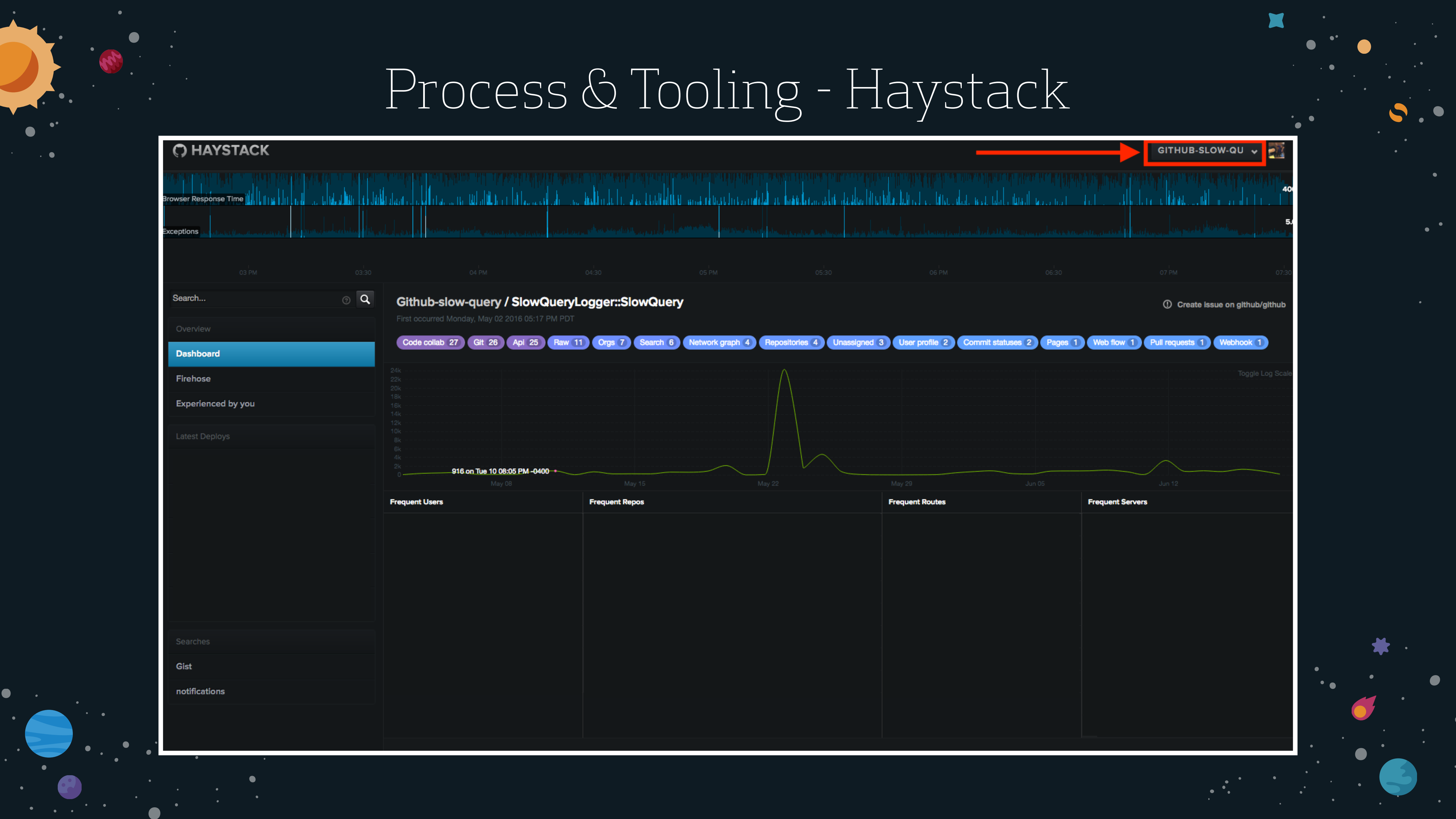
Finding N+1's

125.185ms	0.671ms	0	SELECT DISTINCT `labels`.* FROM `labels` INNER JOIN `issues_labels` ON `labels`.`id` = `issues_labels`.`label_id` WHERE `issues_labels`.`issue_id` = 149018956 ORDER BY name app/views/issues/_issue.html.erb:33
127.569ms	0.532ms	1	SELECT `users`.* FROM `users` WHERE `users`.`id` = 7544775 LIMIT 1 app/models/repository.rb:655
129.688ms	0.575ms	0	SELECT DISTINCT `labels`.* FROM `labels` INNER JOIN `issues_labels` ON `labels`.`id` = `issues_labels`.`label_id` WHERE `issues_labels`.`issue_id` = 149002571 ORDER BY name app/views/issues/_issue.html.erb:33
132.970ms	0.555ms	1	SELECT DISTINCT `labels`.* FROM `labels` INNER JOIN `issues_labels` ON `labels`.`id` = `issues_labels`.`label_id` WHERE `issues_labels`.`issue_id` = 148207422 ORDER BY name app/views/issues/_issue.html.erb:33
137.602ms	1.209ms	1	SELECT DISTINCT `labels`.* FROM `labels` INNER JOIN `issues_labels` ON `labels`.`id` = `issues_labels`.`label_id` WHERE `issues_labels`.`issue_id` = 144654005 ORDER BY name app/views/issues/_issue.html.erb:33
141.084ms	0.533ms	2	SELECT DISTINCT `labels`.* FROM `labels` INNER JOIN `issues_labels` ON `labels`.`id` = `issues_labels`.`label_id` WHERE `issues_labels`.`issue_id` = 143395300 ORDER BY name app/views/issues/_issue.html.erb:33

Process & Tooling - Haystack



The screenshot displays the Haystack monitoring interface. At the top, a red arrow points to a dropdown menu labeled 'GITHUB-SLOW-QU'. Below this, a timeline view shows a significant spike in activity around May 22, 2016, at 10:08:05 PM. The main panel displays a line graph titled 'Github-slow-query / SlowQueryLogger::SlowQuery' with a peak value of 916. The left sidebar contains navigation links for Overview, Dashboard, Firehose, Experienced by you, Latest Deploys, Searches, Gist, and notifications. The bottom section shows a table with columns for Frequent Users, Frequent Repos, Frequent Routes, and Frequent Servers.



Performance Culture



Everybody's
responsibility

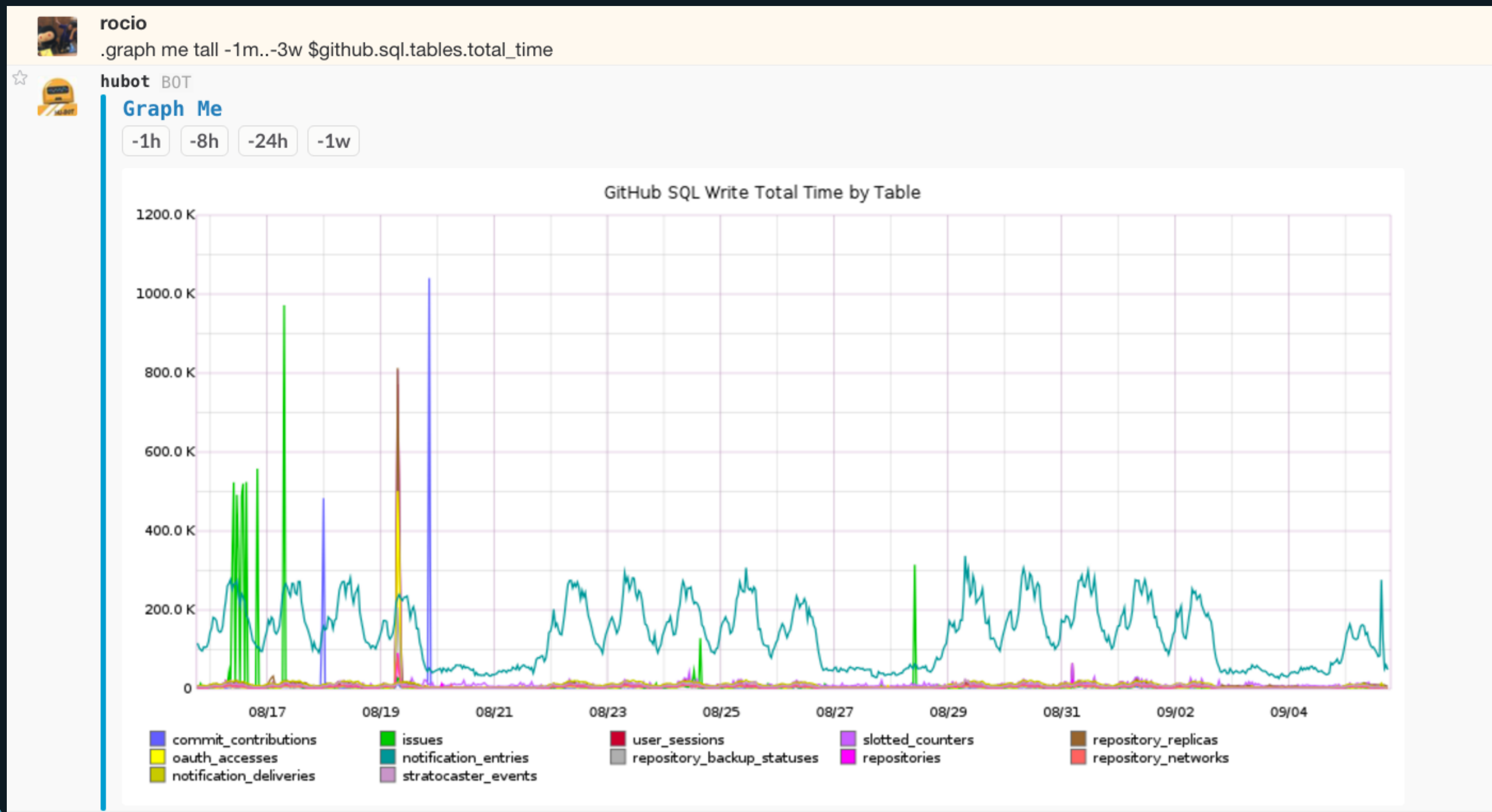


Process &
Tooling

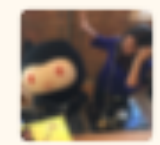


Metrics

Measure all the things



Measure all the things



rocio

.graph me -1m..-3w @github.common_problem_rates



hubot BOT

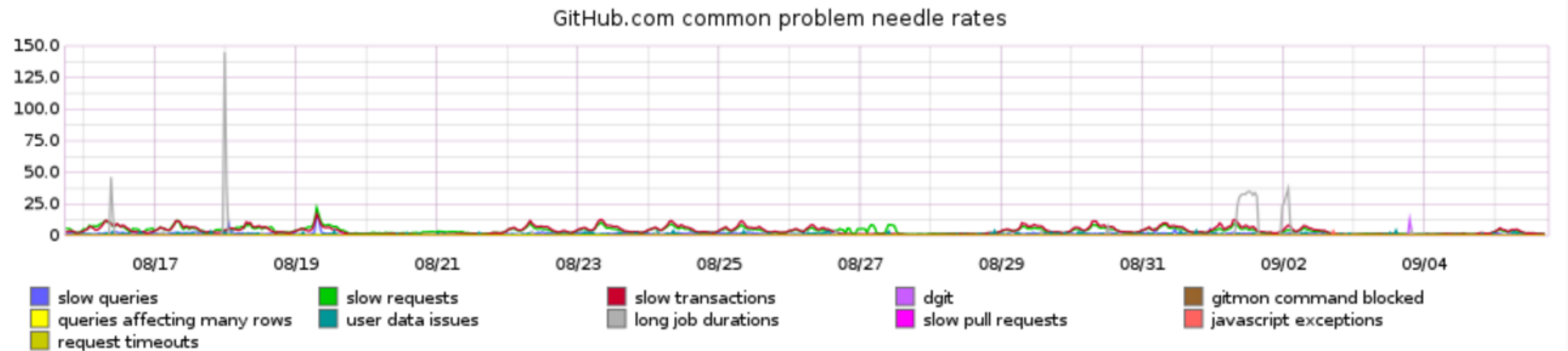
Graph Me

-1h

-8h

-24h

-1w



Measure all the things

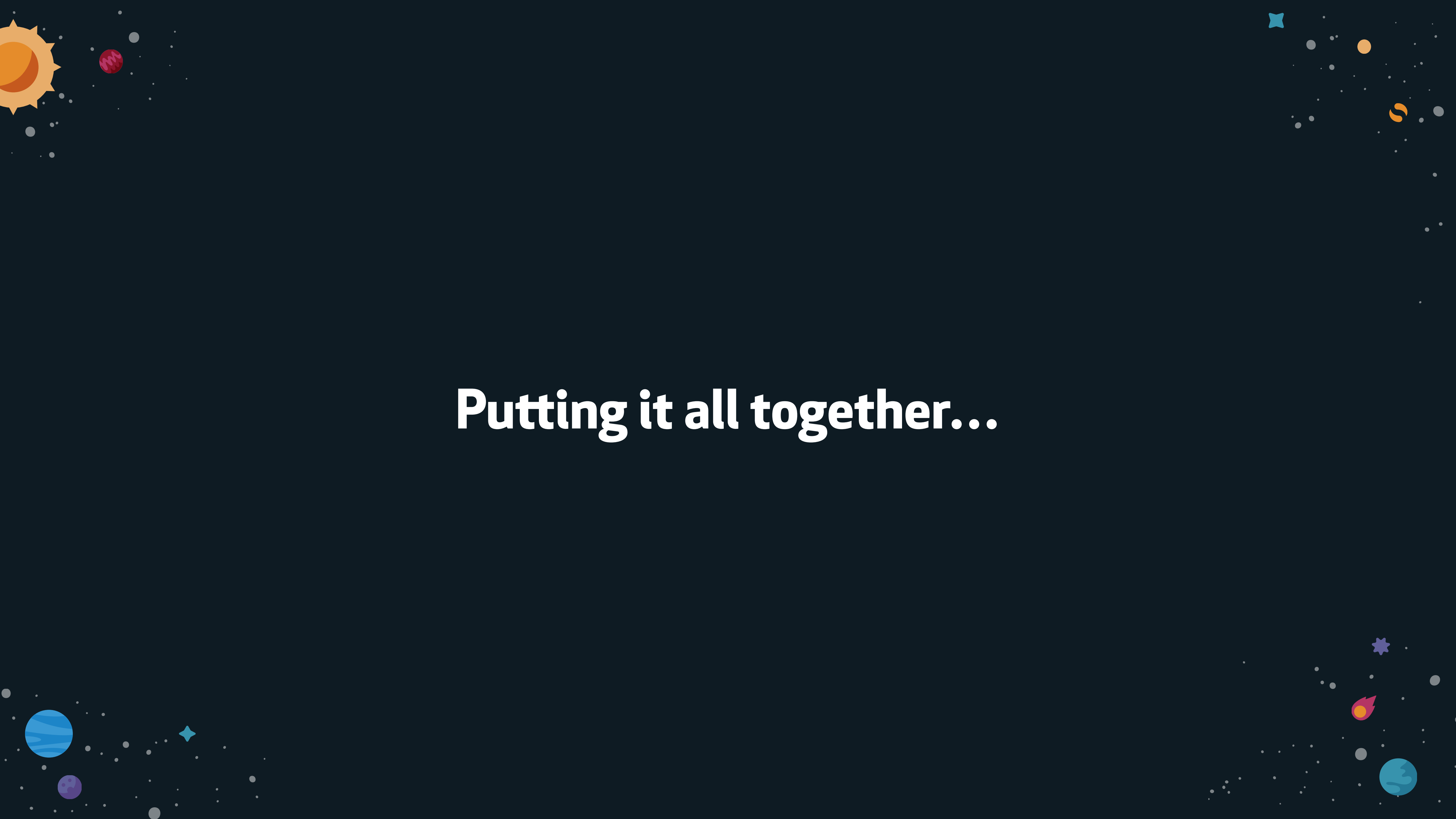
Saturday, June 21, 2014

#LatencyTipOfTheDay : Measure what you need to monitor. Don't just monitor what you happen to be able to easily measure.

Posted by [Gil Tene](#) at 8:40 PM




Recommend this on Google

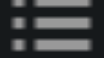


Putting it all together...

Github-slow-query / SlowQueryLogger::SlowQuery

August 20, 2016 1:37:09 PM

 [Create issue on github/github](#)

 [Related needles](#)



rocio mentioned in [Add new index commit_contributions](#)

August 22, 2016 8:18:47 PM

[github/github#60403](#)

```
[1.18 sec] SELECT DISTINCT `commit_contributions`.`user_id` FROM `commit_contributions` WHERE
`commit_contributions`.`repository_id` = 16362580 ORDER BY committed_date DESC LIMIT 894
/*application:github,category:ajax,route:suggestions#show,request_id:0410C166:2CF CF:5EDBA8F:57B89544*/
```


VividCortex

0b2ea5192f35458f

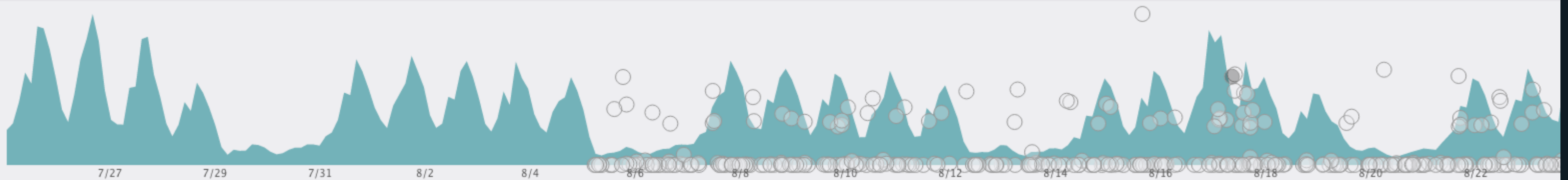
First Seen: **3 months ago** | Last Seen: **16 minutes ago**

```
select distinct `commit_contributions`.`user_id` from `commit_contributions` where `commit_contributions`.`repository_id`=? order by committed_date desc limit ?
```

Metric: Total Time

Filter Samples by Query Text

Show Samples



SAMPLE INFO

Query Text

```
SELECT DISTINCT `commit_contributions`.`user_id` FROM `commit_contributions` WHERE  
`commit_contributions`.`repository_id` = 28023 ORDER BY committed_date DESC LIMIT 855  
/*application:github,category:ajax,route:suggestions#show,request_id:DCF49FB6:19574:2AABB1B:57B4F890*/
```

Cloning the table to staging



rocio

.mysql clone commit_contributions



A clone job has been created for the `github_production.commit_contributions` table from host `github-production-mysql-1`. You will be notified in [#database-ops](#) when the clone is complete.

Testing a new index and open PR

Add new index commit_contributions #60403

Merged

rocio merged 9 commits into master from index_commit_comments 15 days ago

Conversation 22

Commits 9

Files changed 3

+20 -2

rocio commented 17 days ago • edited

Context

Commit contribution queries are very slow as shown by [vividcortex](#) and [this Haystack needle](#).

Solution

After testing a new index over `(repository_id, user_id, committed_date)` with the following query:

```
[1.18 sec] SELECT DISTINCT `commit_contributions`.`user_id` FROM `commit_contributions` WHERE `co`
```

the query execution goes from
93 rows in set (3.10 sec)
to
93 rows in set (0.01 sec) in the replica.

Projects

None yet

Labels

DB migration

DB migration - completed

DB Performance Improvement

Milestone

No milestone

Assignees

No one—assign yourself

8 participants

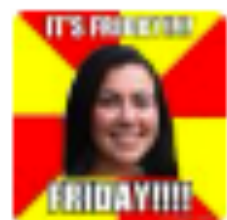
Migration is approved



✓ MikeMcQuaid approved these changes 15 days ago

[View changes](#)

Makes sense to me 👍



✓ bryanaknight approved these changes 14 days ago

[View changes](#)



hubot commented 14 days ago

+ 😊 ✎ ✕

This migration was reviewed by @github/platform-data and is ready to be scheduled and run by @github/database-infrastructure.

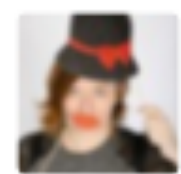
Schedule migration



ggunson

.migration-queue schedule 60403 today

Run migration



ggunson

.migration unpostpone



hubot BOT

Unpostponed

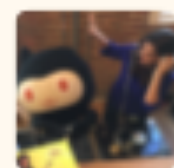


@ggunson: github_production.commit_contributions migration is complete! 🐙 🎉

Ship it



.explain



rocio

2:51 PM

```
.explain SELECT DISTINCT commit_contributions.user_id FROM commit_contributions WHERE  
commit_contributions.repository_id = 16362580 ORDER BY committed_date DESC LIMIT 100
```



hubot BOT

2:51 PM

rocio:

```
***** 1. row *****
```

Select Type: SIMPLE

Table: commit_contributions

Type: ref

Possible Keys: commit_contrib_user_repo_date, commit_contrib_repo_date, index_commit_contributions

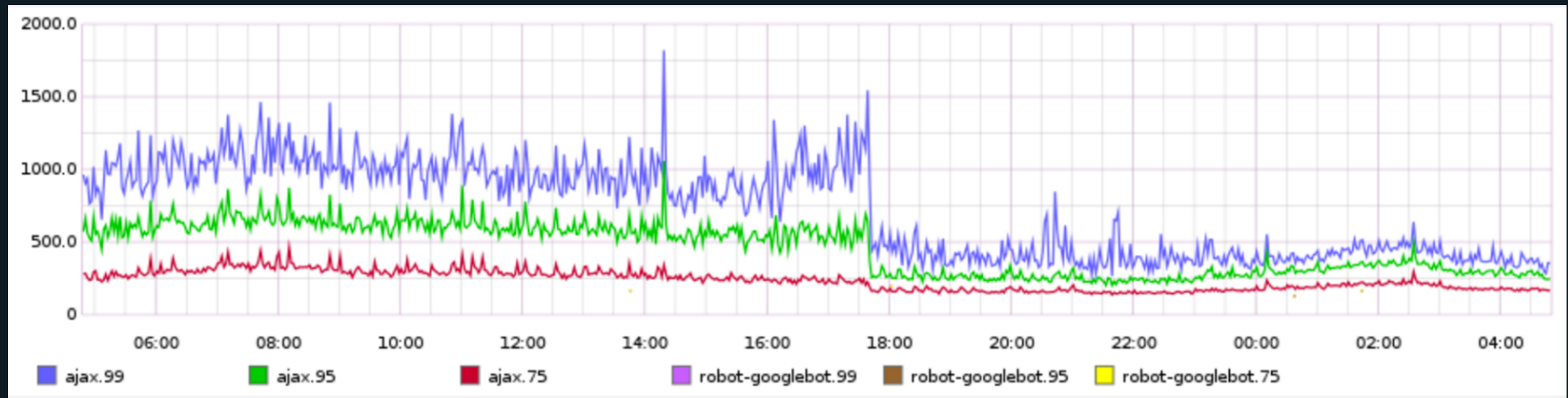
Key: index_commit_contributions_repository_user_date

Key Len: 5

Ref: const

Rows: 25880

Profit



Verify

```
select distinct `commit_contributions`.`user_id` from `commit_contributions` where `commit_contributions`.`repository_id`=? order by  
committed_date desc limit ?
```

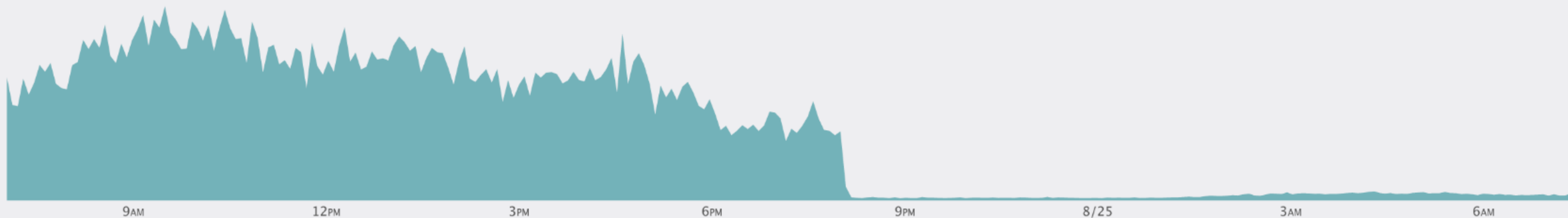
Metric: Total Time

Filter Samples by Query Text

Enter Query text you wish to filter by

Show Samples

OFF



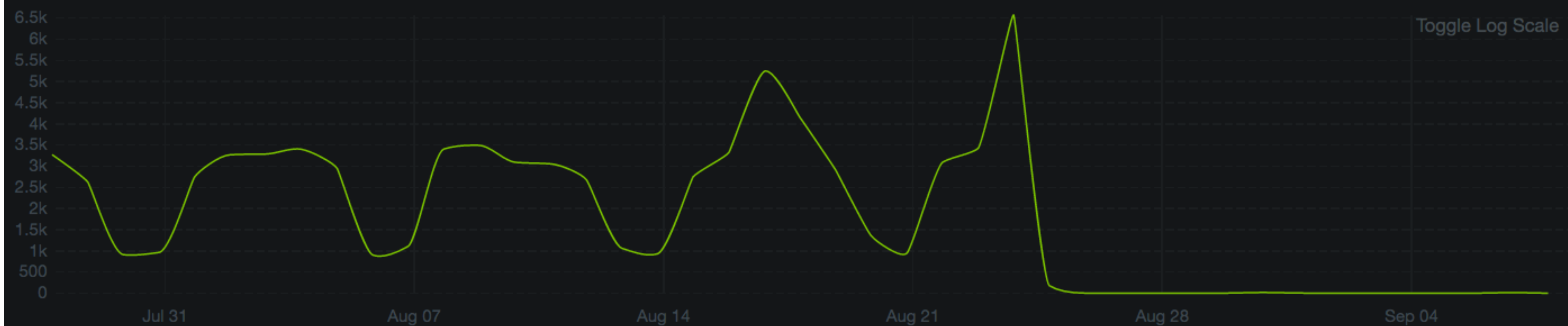
No more needles

Github-slow-query / SlowQueryLogger::SlowQuery

[Create issue on github/github](#)

First occurred Wednesday, July 27 2016 05:17 PM PDT

Search 92 Orgs 8 Api 8





Using Science

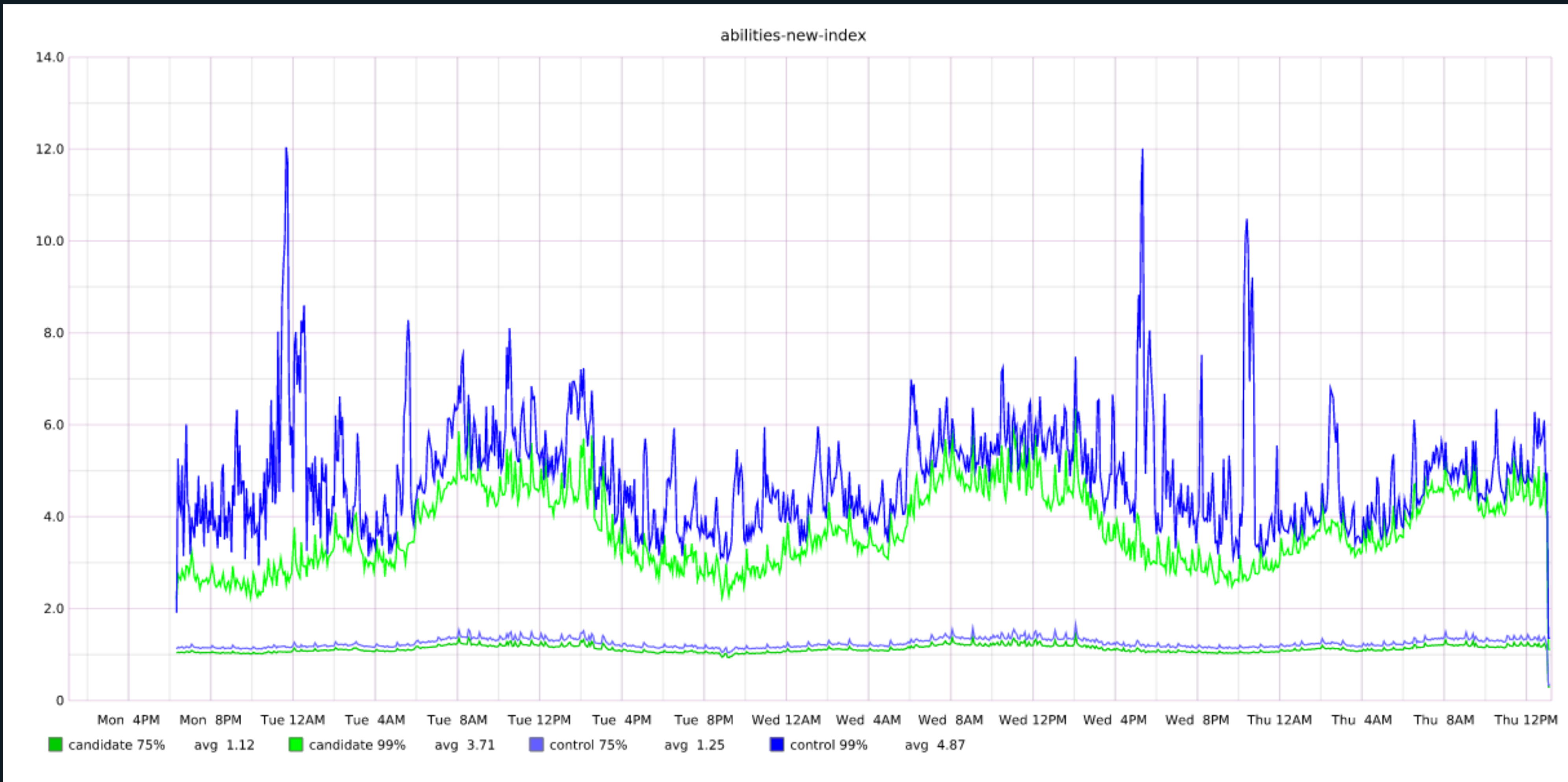
Scientist

```
require "scientist"

def method
  science("test-index-optimizer") do |experiment|
    experiment.use {
      Issue.from("issues USE INDEX( existing_index_name )")
    }
    experiment.try {
      Issue.from("issues USE INDEX( new_index_name )")
    }
  end
end
```

<https://github.com/github/scientist>

Performance test between 2 indexes



Performance Culture



**Everyone's
responsibility**



**Process &
Tooling**



Metrics



MySQL as primary data
store



MySQL

- * Predictability
- * Scalability
- * High performance
- * High availability
- * Operational experience
- * Optimized for fast reads

K,V table in MySQL



rocio

9:59 AM

.mysql table key_values



hubot BOT

9:59 AM

***** 1. row *****

Table: key_values

Create Table: CREATE TABLE **key_values** (

id int(11) NOT NULL AUTO_INCREMENT,

key varchar(255) NOT NULL,

value blob NOT NULL,

created_at datetime NOT NULL,

updated_at datetime NOT NULL,

expires_at datetime DEFAULT NULL,

PRIMARY KEY (**id**),

UNIQUE KEY **index_key_values_on_key** (**key**),

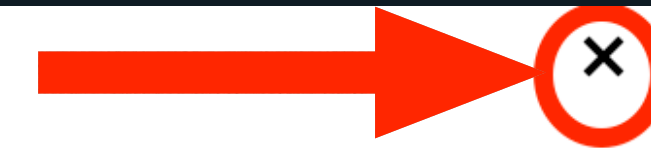
KEY **index_key_values_on_expires_at** (**expires_at**)

) ENGINE=InnoDB

DEFAULT CHARSET=utf8

***** 1. row *****

Storing users dismissal notices in K,V



Seamless communication with teams

Teams are a great way for groups of people to communicate and work on code together. Take a look at why they're great.



Flexible repository access

You can add repositories to your teams with more flexible levels of access (Admin, Write, Read).



Request to join teams

Members can quickly request to join any team. An owner or team maintainer can approve the request.



Team mentions

Use team @mentions (ex. **@github/design** for the entire team) in any comment, issue, or pull request.

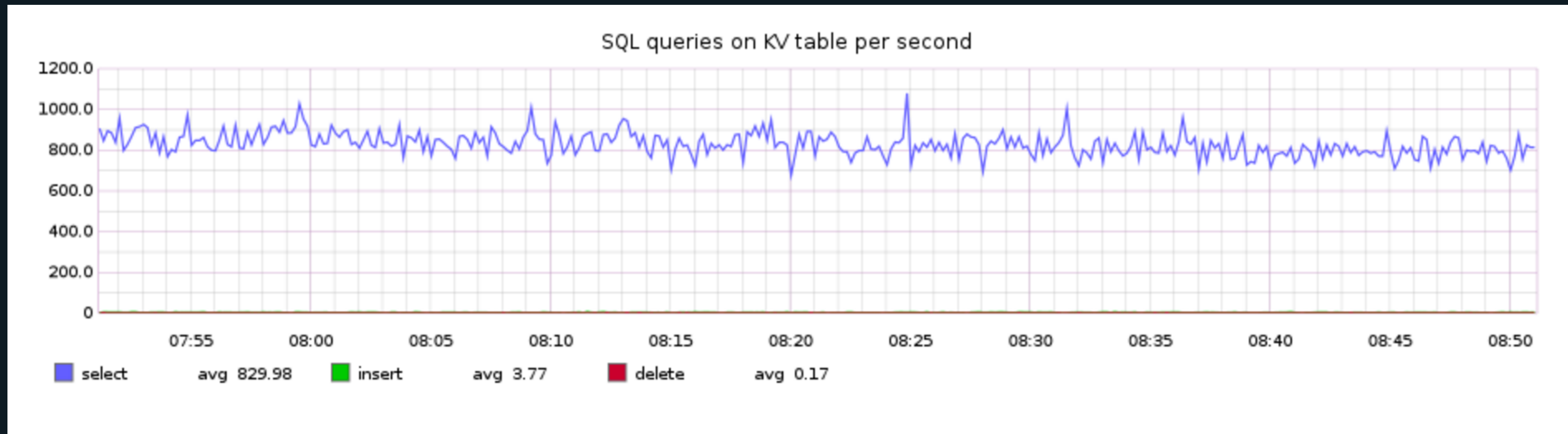
Storing users dismissal notices in K,V

```
# Set User Key Value for Notice
#
# notice – key to be added
def dismiss_notice(notice)
  |  GitHub.kv.set("user.dismissed_notice.#{notice}.#{id}", notice)
end
```


Checking for presence

```
# Get User Key Value for Notice
#
# notice – key to check for in key values
#
# Returns true or false
def dismissed_notice?(notice)
  |  GitHub.kv.get("user.dismissed_notice.#{notice}.#{id}").value! == notice
end
```

SQL queries on KV table per second





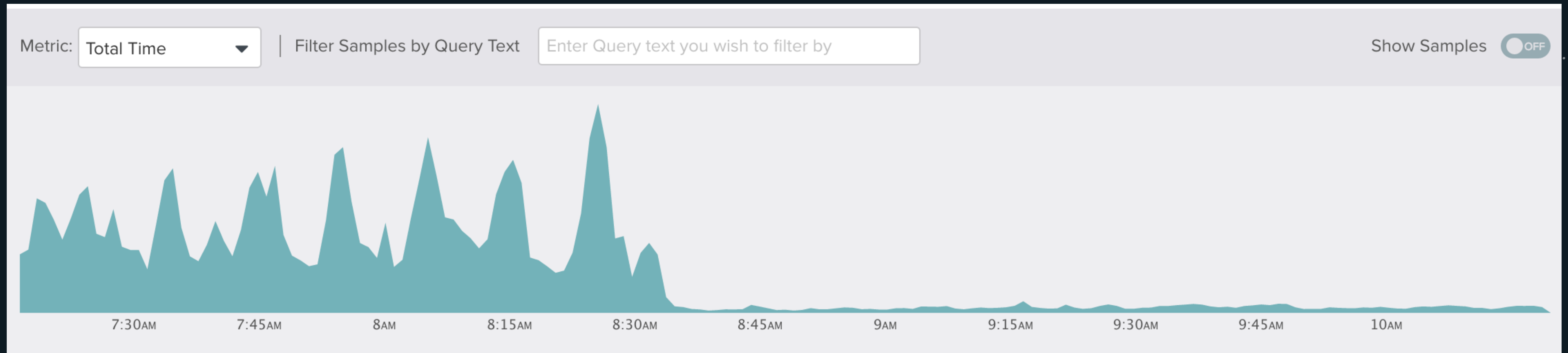
What IS a good index strategy?

- * Build indexes for performance critical queries
- * Build index order that benefits more queries
 - * `SELECT * FROM issues WHERE user_id = 2 AND repository_id = 2;`
 - * `SELECT * FROM issues WHERE user_id > 2 AND repository_id = 2;`
 - * **INDEX ON (repository_id, user_id) is best**
- * Prefer to extend an existing index rather than create a new one
- * Favor multi column indexes



What's NOT a good index strategy?

Lack of indexes

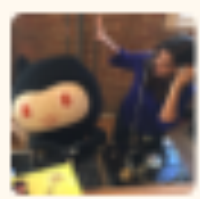




Unnecessary indexes

- * Indexes require space, the more you have the bigger the table.
- * Write operations will be slower.

Finding unused indexes



rocio

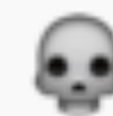
.mysql dead-indexes issues



hubot BOT



Dead indexes for github_production.issues



No redundant indexes for issues.

No non-unique unused indexes for issues found



What's coming next in
MySQL?

You had me at emoji



markcallaghan
@markcallaghan



Following

MySQL 8 had me at better support for emojis

Vadim Tkachenko @VadimTk

Common Table Expression and Invisible Index are coming in MySQL 8.0 [Inkd.in/eY-Q6a](https://www.inkd.in/eY-Q6a)

RETWEETS

5

LIKES

7



8:38 PM - 1 Sep 2016



5



7





It's work in progress



Hubot <https://hubot.github.com>

gh-ost <https://github.com/github/gh-ost>

Haystack <http://githubengineering.com/exception-monitoring-and-response>

Peek <https://github.com/peek/peek>

Scientist <https://github.com/github/scientist>

Graphite <http://graphite.wikidot.com>



FIN

Gracias!

 **@rokkzy**
 **@rocio**