Django the next steps

Django is great*!

Tips, tricks and tuning

-beyond-

simple Django projects.

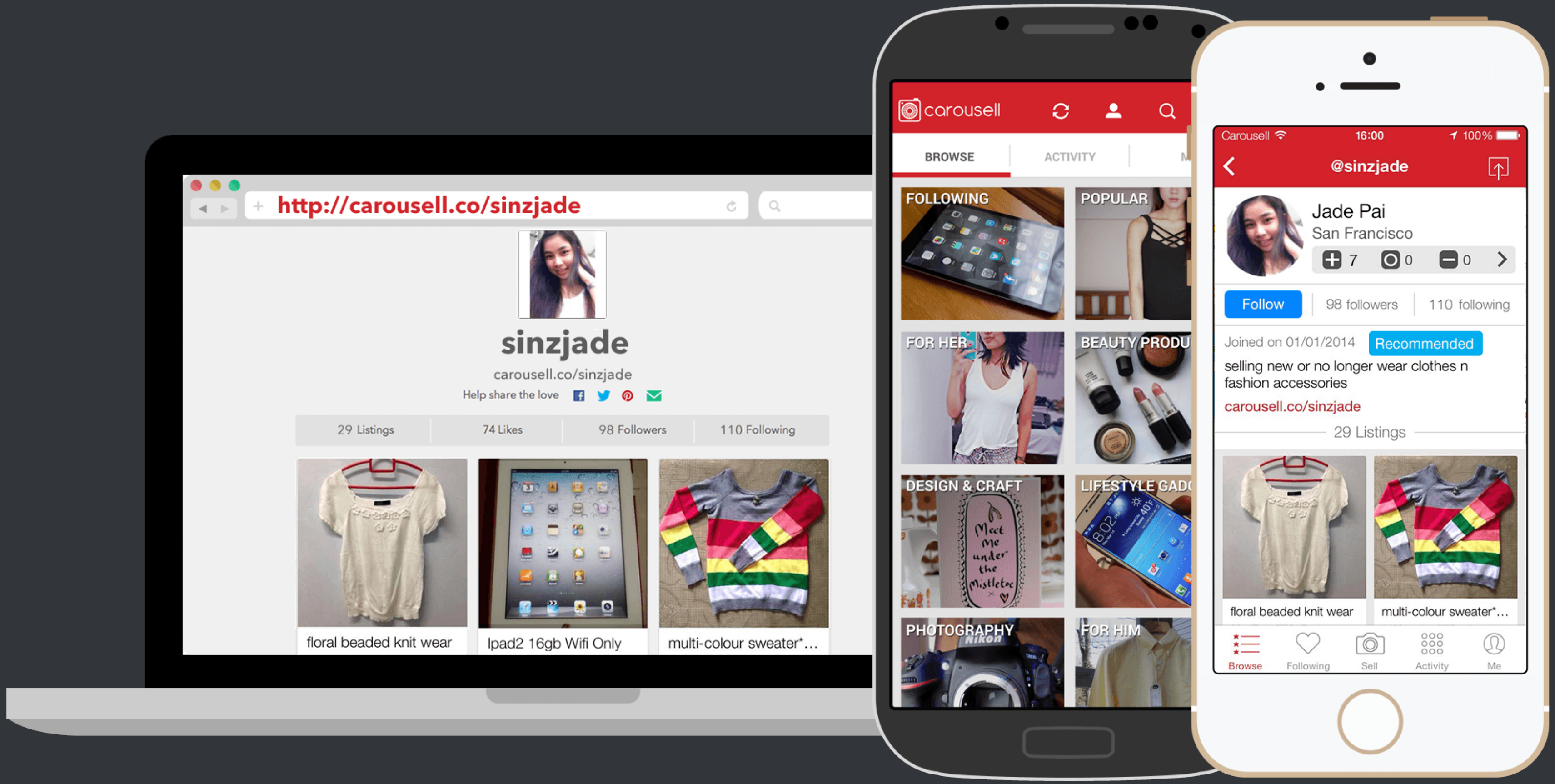
premature optimization is the root of all evil

yet we should not pass up our opportunities in that critical 3%

hi there, I am

victor neo

Lead Engineer @ Carousell

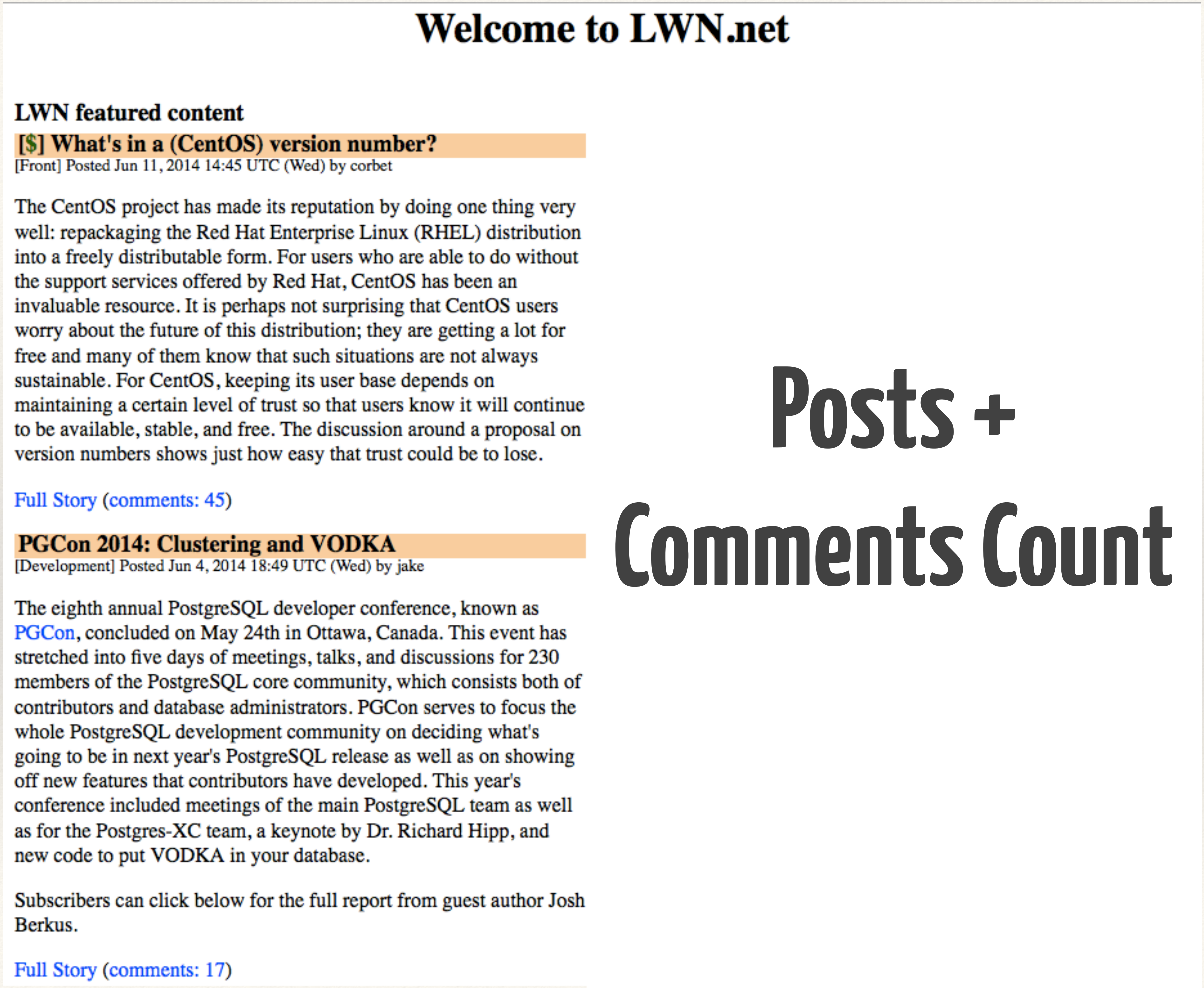

http://carousell.co/sinzjade
sinzjade
carousell.co/sinzjade
Help share the love
29 Listings
74 Likes
98 Followers
110 Following
floral beaded knit wear
Ipad2 16gb Wifi Only
multi-colour sweater*...
carousell
BROWSE
ACTIVITY
FOLLOWING
POPULAR
FOR HER
BEAUTY PRODU
DESIGN & CRAFT
LIFESTYLE GAD
PHOTOGRAPHY
FOR HIM
Carousell
16:00
100%
@sinzjade
Jade Pai
San Francisco
7
0
0
Follow
98 followers
110 following
Joined on 01/01/2014
Recommended
selling new or no longer wear clothes n fashion accessories
carousell.co/sinzjade
29 Listings
floral beaded knit wear
multi-colour sweater*...
Browse
Following
Sell
Activity
Me

➡ Caching*

ORM / Database*

Asynchronous Tasks

Logging / Monitoring


Welcome to LWN.net

LWN featured content

[$] What's in a (CentOS) version number?
[Front] Posted Jun 11, 2014 14:45 UTC (Wed) by corbet

The CentOS project has made its reputation by doing one thing very well: repackaging the Red Hat Enterprise Linux (RHEL) distribution into a freely distributable form. For users who are able to do without the support services offered by Red Hat, CentOS has been an invaluable resource. It is perhaps not surprising that CentOS users worry about the future of this distribution; they are getting a lot for free and many of them know that such situations are not always sustainable. For CentOS, keeping its user base depends on maintaining a certain level of trust so that users know it will continue to be available, stable, and free. The discussion around a proposal on version numbers shows just how easy that trust could be to lose.

Full Story (comments: 45)

PGCon 2014: Clustering and VODKA
[Development] Posted Jun 4, 2014 18:49 UTC (Wed) by jake

The eighth annual PostgreSQL developer conference, known as PGCon, concluded on May 24th in Ottawa, Canada. This event has stretched into five days of meetings, talks, and discussions for 230 members of the PostgreSQL core community, which consists both of contributors and database administrators. PGCon serves to focus the whole PostgreSQL development community on deciding what's going to be in next year's PostgreSQL release as well as on showing off new features that contributors have developed. This year's conference included meetings of the main PostgreSQL team as well as for the Postgres-XC team, a keynote by Dr. Richard Hipp, and new code to put VODKA in your database.

Subscribers can click below for the full report from guest author Josh Berkus.

Full Story (comments: 17)

Posts + Comments Count

Homepage of a Blog

1. GET /index
2. Load posts from DB
3. Render template

Takes 200~500ms :(

Caching is your

first layer of defence

Speeding it up

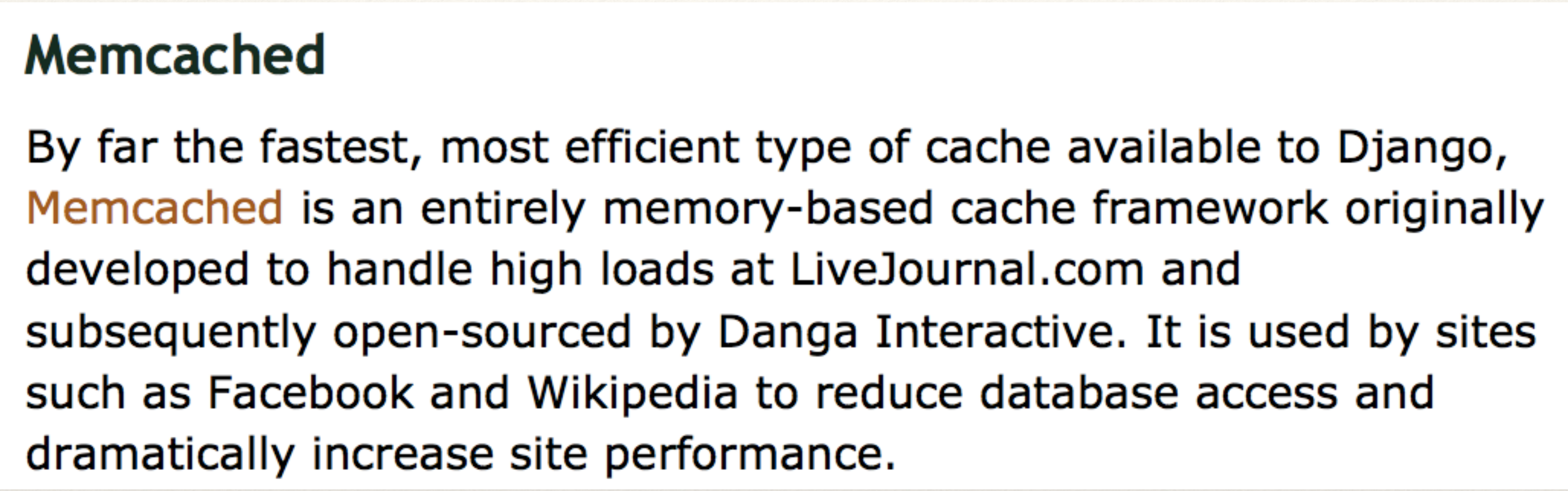
1. GET /index
2. Return cached output

(completes in < 50ms!)

Django’s per view cache

```
@cache_page(60 * 15)
def homepage(request):
```

Caches homepage for ~15 minutes



Memcached

By far the fastest, most efficient type of cache available to Django, Memcached is an entirely memory-based cache framework originally developed to handle high loads at LiveJournal.com and subsequently open-sourced by Danga Interactive. It is used by sites such as Facebook and Wikipedia to reduce database access and dramatically increase site performance.

But I need more CONTROL!

Django’s low level cache

```
posts = Post.objects.….all()

# some complicated logic here

cache.set(‘posts’, posts, 60)
```

Django Cache Machine

Automatic caching and invalidation of Models

Provides a Cache Manager

Cache Machine

```
# models.py
class Post(models.Model):
    objects = CachingManager()

Post.objects.….all()
```

Subsequent calls are cached!

Can I use REDIS?

Redis for Caching

Django-Redis: Use redis instead of Memcached

Django-cacheops: Similar to Cache Machine, for Redis

Decisions, decisions

	Memcached	Redis
Feature-packed	Cache Machine	Django-Cacheops
Custom	Roll your own with Django cache API	Django-Redis

Rule of thumb

Cache if data freshness is not an issue

Critical if computation is expensive

Caching

→ ORM / Database

Asynchronous Tasks

Logging / Monitoring

Foreign Keys

```
e = Post.objects.get(id=5)

e.blog.name        # Additional
                   # DB query
                   # for blog
```

select_related

```
e = Entry.objects.\
    select_related(‘blog’).\
    get(id=5)

e.blog.name    # No DB query!
```

select vs prefetch related

One-to-one / Foreign Key:
select_related

Many-to-many / Many-to-one / Generic relations:
prefetch_related

```
Post.objects.filter(
  is_published=True,
  is_edited=True,
  ...).all()
```

```
SELECT … FROM blog_posts
WHERE is_published = true
AND is_edited = true
AND …
```

Know thy Database!

PostgreSQL

```
EXPLAIN ANALYZE <query>
```

Non-Active Users

```
User.objects.\
filter(is_active=False).\
.all()
```

Non-Active Users (SQL)

```
SELECT * FROM auth_users
WHERE is_active = false;
```

Analyzing Queries

Prepend

```
EXPLAIN ANALYZE
SELECT * FROM AUTH_USERS
WHERE is_active = false;
```

```
                  QUERY PLAN
------------------------------------
 Seq Scan on auth_user

(cost=0.00..15761.01 rows=2413 width=140)
(actual time=0.161..279.318 rows=2384 loops=1)
 Filter: (NOT is_active)

 Total runtime: 280.890 ms

(3 rows)
```

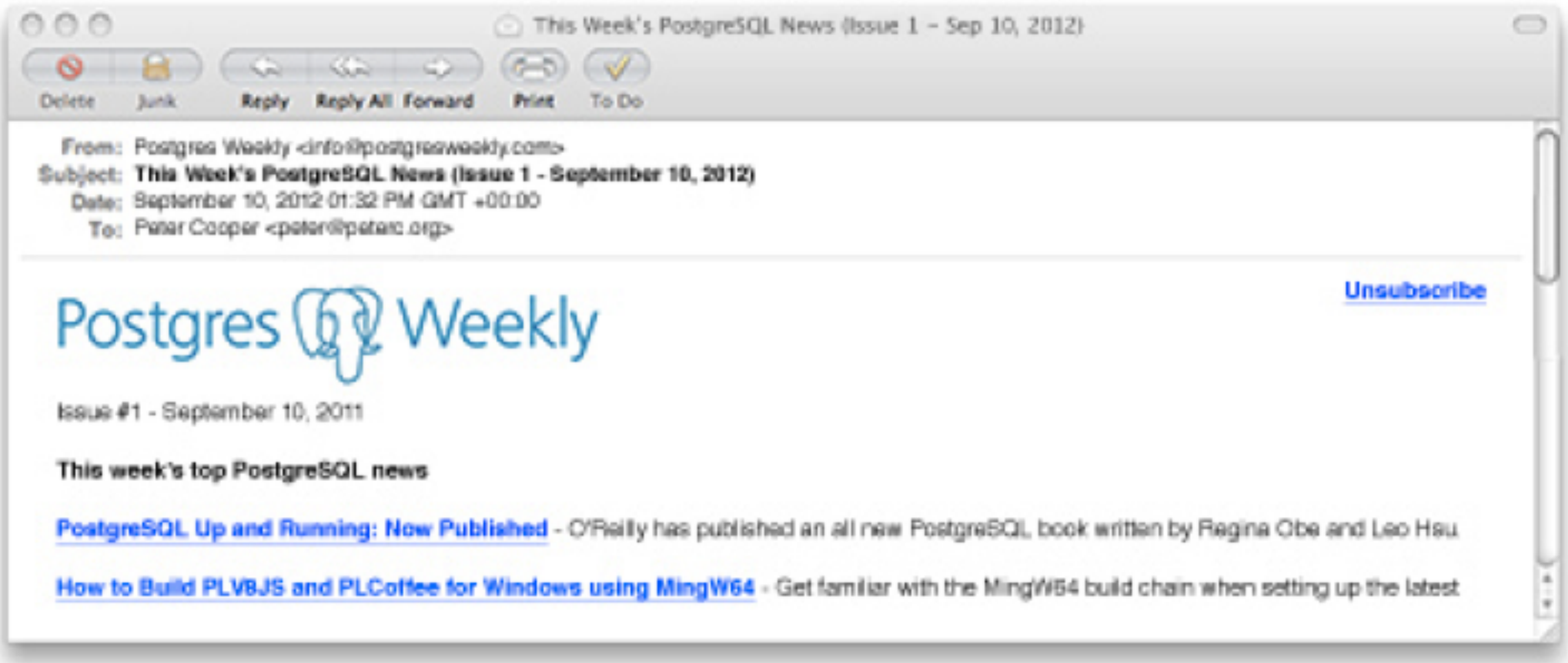
Create an Index on is_active attribute

```
CREATE INDEX CONCURRENTLY
ON auth_user (is_active)
WHERE is_active = false;
```

```
            QUERY PLAN
---------------------------
Index Scan using
auth_user_is_active_idx on
auth_user

(cost=0.00..59.19 rows=2413 width=140)
(actual time=0.129..8.824 rows=2384 loops=1)

Index Cond: (is_active = false)

Total runtime: 9.779 ms

(3 rows)
```

Postgres Weekly

A free, once–weekly e-mail round-up of PostgreSQL news and articles



From: Postgres Weekly <info@postgresweekly.com>
Subject: **This Week's PostgreSQL News (Issue 1 - September 10, 2012)**
Date: September 10, 2012 01:32 PM GMT +00:00
To: Peter Cooper <peter@peterc.org>

Unsubscribe

Postgres Weekly

Issue #1 - September 10, 2011

This week's top PostgreSQL news

PostgreSQL Up and Running: Now Published - O'Reilly has published an all new PostgreSQL book written by Regina Obe and Leo Hsu

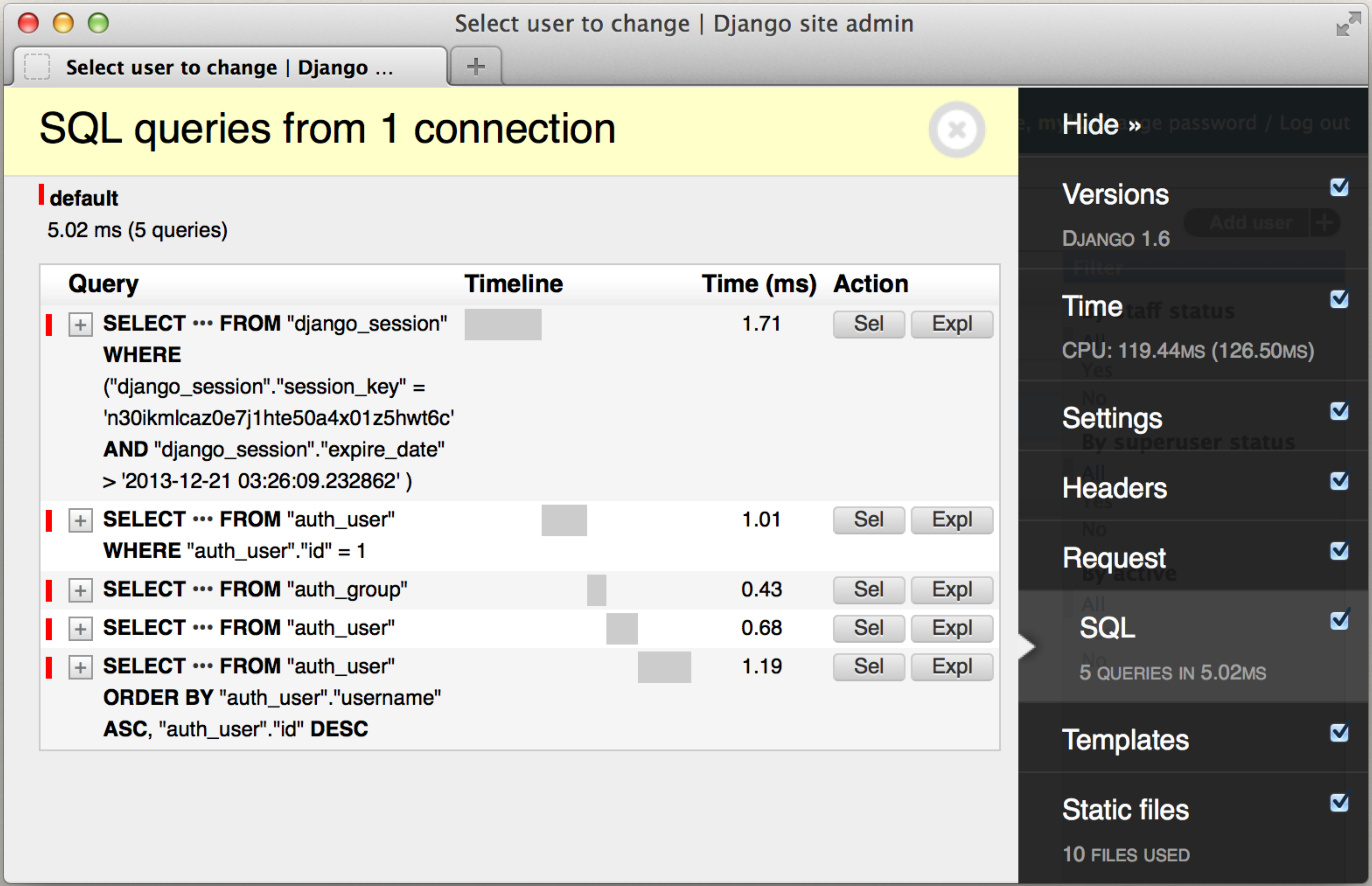
How to Build PLV8JS and PLCoffee for Windows using MingW64 - Get familiar with the MingW64 build chain when setting up the latest

Enter your e-mail address

Sign Me Up!

Wait, how do I view the SQL queries?

Django debug toolbar



Select user to change | Django site admin

SQL queries from 1 connection

default
5.02 ms (5 queries)

Query	Timeline	Time (ms)	Action
SELECT ••• FROM "django_session" **WHERE** ("django_session"."session_key" = 'n30ikmlcaz0e7j1hte50a4x01z5hwt6c' **AND** "django_session"."expire_date" > '2013-12-21 03:26:09.232862')		1.71	Sel Expl
SELECT ••• FROM "auth_user" **WHERE** "auth_user"."id" = 1		1.01	Sel Expl
SELECT ••• FROM "auth_group"		0.43	Sel Expl
SELECT ••• FROM "auth_user"		0.68	Sel Expl
SELECT ••• FROM "auth_user" **ORDER BY** "auth_user"."username" **ASC**, "auth_user"."id" **DESC**		1.19	Sel Expl

Hide »

Versions ✓
Django 1.6

Time ✓
CPU: 119.44ms (126.50ms)

Settings ✓

Headers ✓

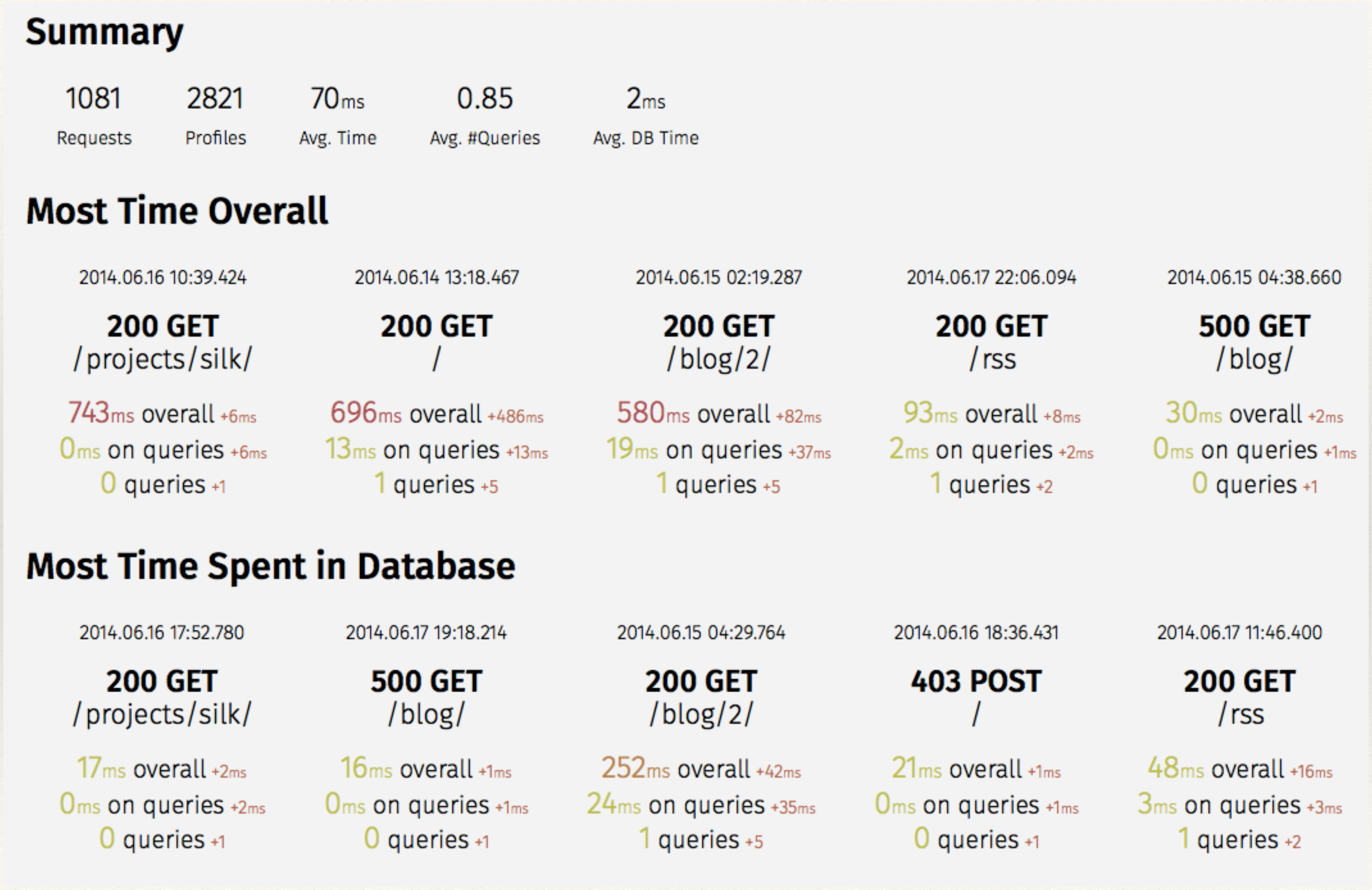
Request ✓

SQL ✓
5 queries in 5.02ms

Templates ✓

Static files ✓
10 files used

Silk



Summary

1081	2821	70ms	0.85	2ms
Requests	Profiles	Avg. Time	Avg. #Queries	Avg. DB Time

Most Time Overall

2014.06.16 10:39.424	2014.06.14 13:18.467	2014.06.15 02:19.287	2014.06.17 22:06.094	2014.06.15 04:38.660
200 GET /projects/silk/	**200 GET** /	**200 GET** /blog/2/	**200 GET** /rss	**500 GET** /blog/
743ms overall +6ms	696ms overall +486ms	580ms overall +82ms	93ms overall +8ms	30ms overall +2ms
0ms on queries +6ms	13ms on queries +13ms	19ms on queries +37ms	2ms on queries +2ms	0ms on queries +1ms
0 queries +1	1 queries +5	1 queries +5	1 queries +2	0 queries +1

Most Time Spent in Database

2014.06.16 17:52.780	2014.06.17 19:18.214	2014.06.15 04:29.764	2014.06.16 18:36.431	2014.06.17 11:46.400
200 GET /projects/silk/	**500 GET** /blog/	**200 GET** /blog/2/	**403 POST** /	**200 GET** /rss
17ms overall +2ms	16ms overall +1ms	252ms overall +42ms	21ms overall +1ms	48ms overall +16ms
0ms on queries +2ms	0ms on queries +1ms	24ms on queries +35ms	0ms on queries +1ms	3ms on queries +3ms
0 queries +1	0 queries +1	1 queries +5	0 queries +1	1 queries +2

Django Query Inspector

```
[SQL] repeated query (6x): SELECT "customer_role"."id",
  "customer_role"."contact_id", "customer_role"."name"
  FROM "customer_role" WHERE "customer_role"."contact_id" = ?
```

Suitable for API projects with no web UI

Search Engine?

PostgreSQL comes with full-text search

Heavy search traffic?
Consider Elasticsearch, Solr

Haystack
Modular search for django
Find the needle you're looking for.
Download

Haystack

Supports Elasticsearch, Solr and more

Easy to get started with manage.py commands

Familiar ORM syntax for searching

Doing a search

```
SearchQuerySet().models(Post).
  filter(content=‘Python’).all()

[
<SearchResult: blog.post (pk=u’1')>,
<SearchResult: blog.post (pk=u’2')>,
…]
```

Happy ORM

Use select/prefetch related to reduce queries

Understand your DB's query planner

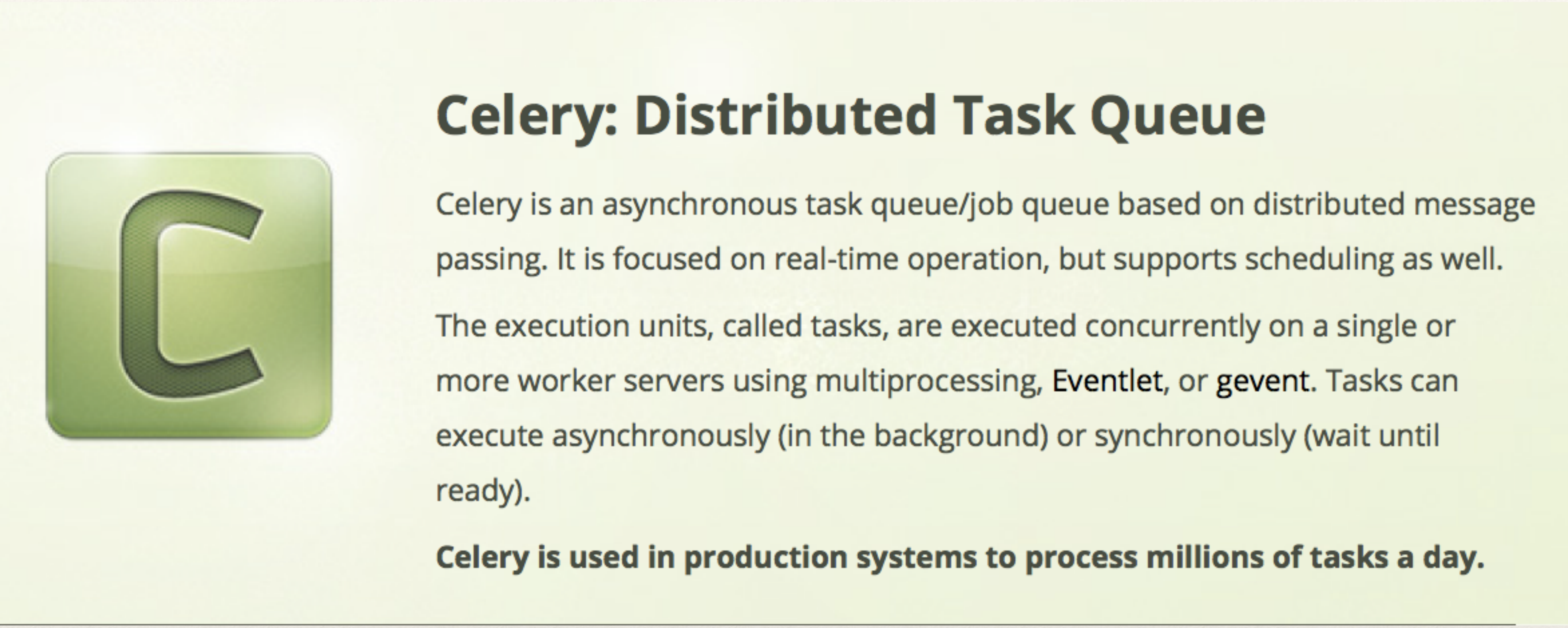
Haystack for search

Caching

ORM / Database

➡ Asynchronous Tasks

Logging / Monitoring



Celery: Distributed Task Queue

Celery is an asynchronous task queue/job queue based on distributed message passing. It is focused on real-time operation, but supports scheduling as well.

The execution units, called tasks, are executed concurrently on a single or more worker servers using multiprocessing, Eventlet, or gevent. Tasks can execute asynchronously (in the background) or synchronously (wait until ready).

Celery is used in production systems to process millions of tasks a day.

View is slow :(

```
reset_pw_email(user.email)

# needs to wait for email to
# be sent before we can send
# a response

return HttpResponse(…)
```

Celery Tasks

```
@task
def reset_pw_email(email):
    …
```

Just add @task decorator!

Calling Tasks

```
# In your Django view

reset_pw_email.delay(user.email)
```

View continues,
without waiting for email to be sent

HTTP Callback tasks

```
HttpDispatchTask.delay(
  url='http://a.com/multiply',
  method=‘GET', x=10, y=10)
```

Awesome for microservices

Caching

ORM / Database

Asynchronous Tasks

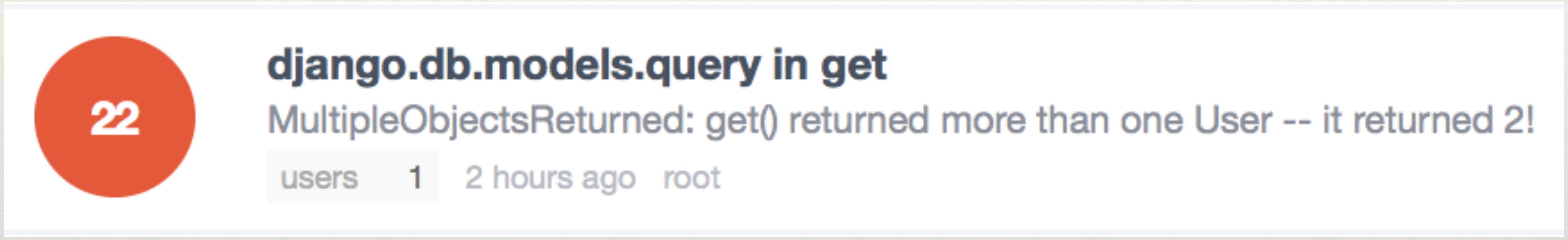
➞ **Logging / Monitoring**

django.db.models.query in get MultipleObjectsReturned: get() returned more than one User -- it returned 2!

Errors Happen

Let’s deal with them

Watch them as they happen



22

django.db.models.query in get

MultipleObjectsReturned: get() returned more than one User -- it returned 2!

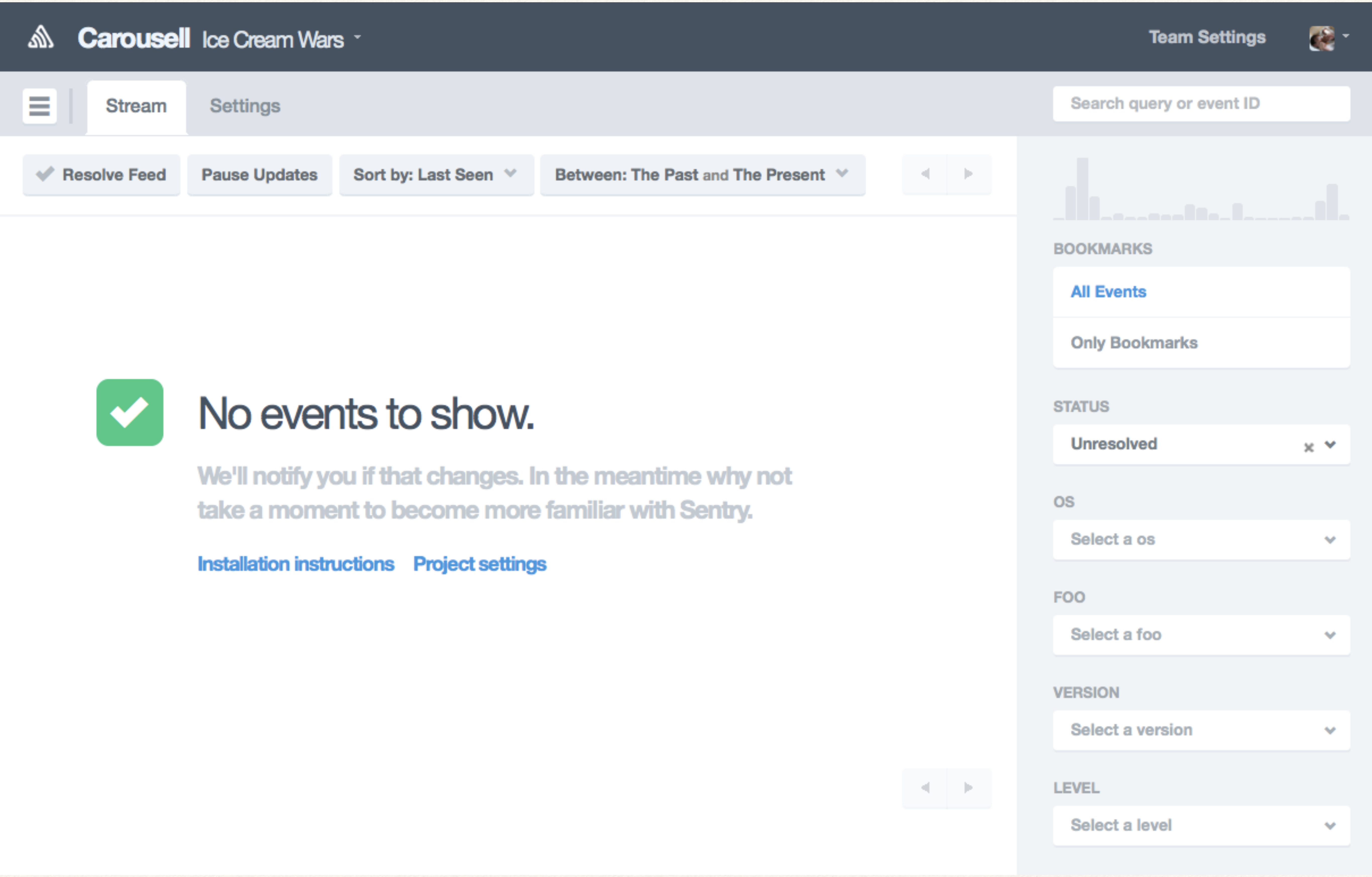
users 1 2 hours ago root

😍 Sentry 😍

Log errors when they happen in production

Open source Django project

Awesome web interface



No events to show.

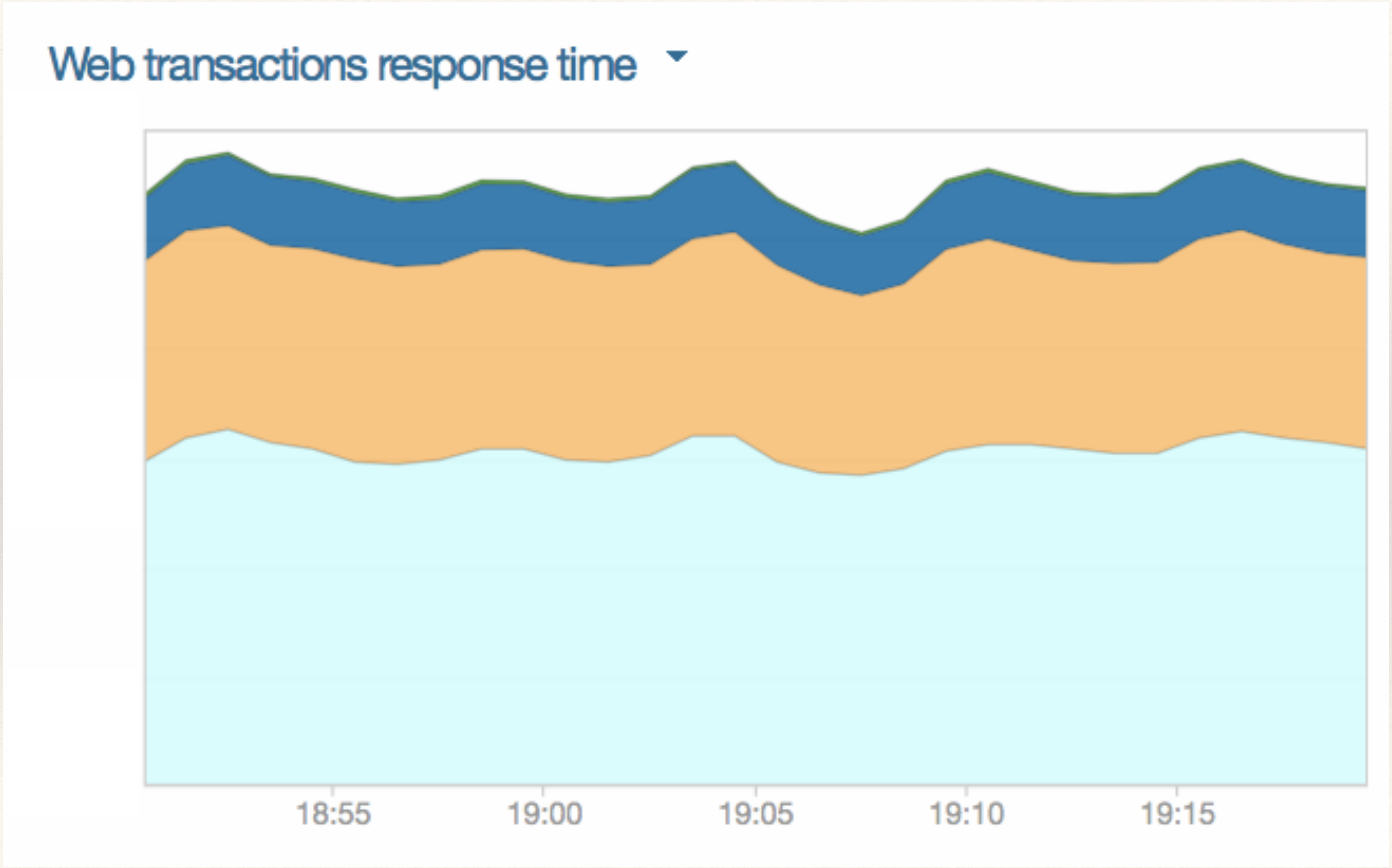
We'll notify you if that changes. In the meantime why not take a moment to become more familiar with Sentry.

Installation instructions   Project settings

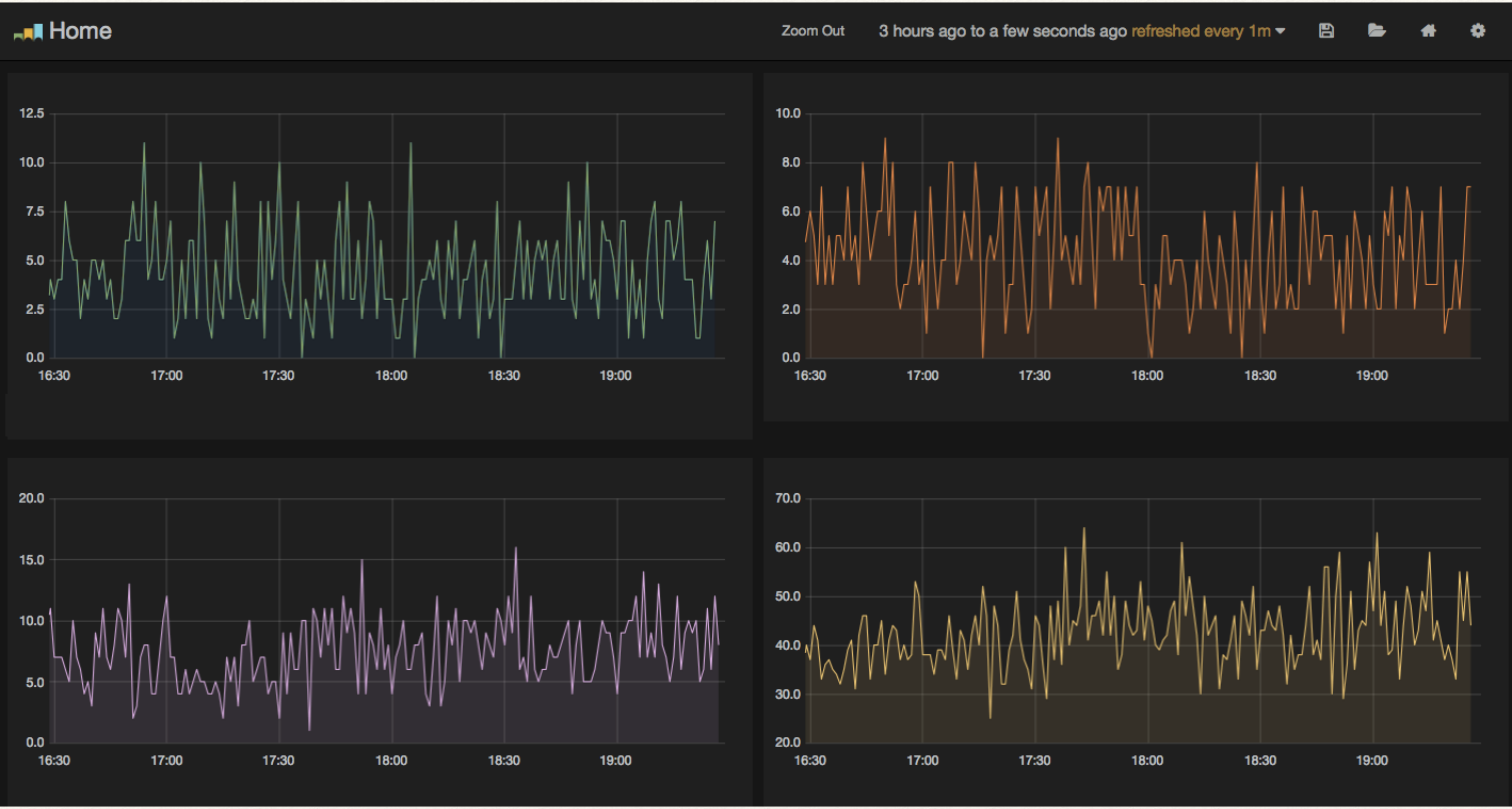
Monitoring

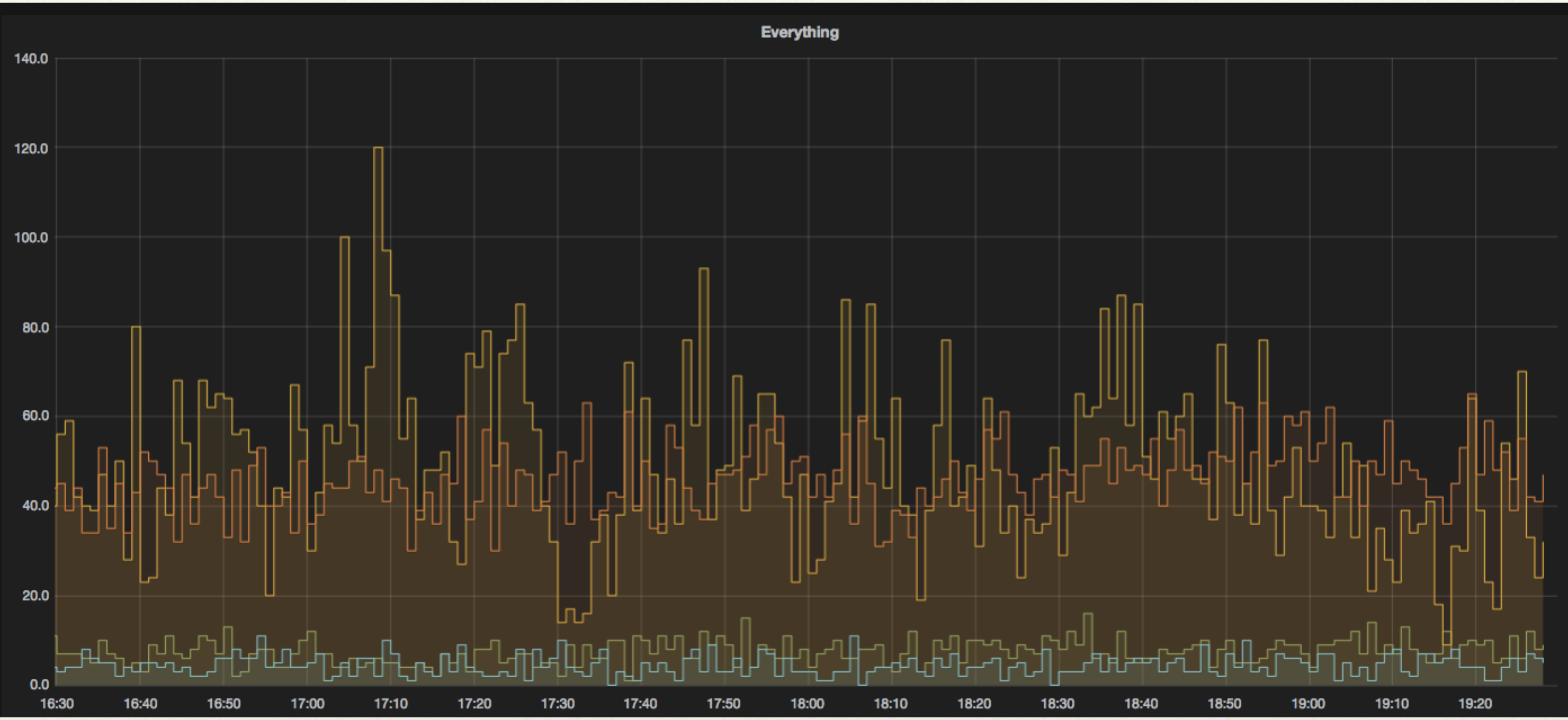
or “is someone using my app”

New Relic



Instrument Everything


Home
Zoom Out
3 hours ago to a few seconds ago refreshed every 1m
12.5
10.0
7.5
5.0
2.5
0.0
16:30
17:00
17:30
18:00
18:30
19:00
10.0
8.0
6.0
4.0
2.0
0.0
20.0
15.0
10.0
5.0
0.0
70.0
60.0
50.0
40.0
30.0
20.0


Everything
140.0
120.0
100.0
80.0
60.0
40.0
20.0
0.0
16:30
16:40
16:50
17:00
17:10
17:20
17:30
17:40
17:50
18:00
18:10
18:20
18:30
18:40
18:50
19:00
19:10
19:20

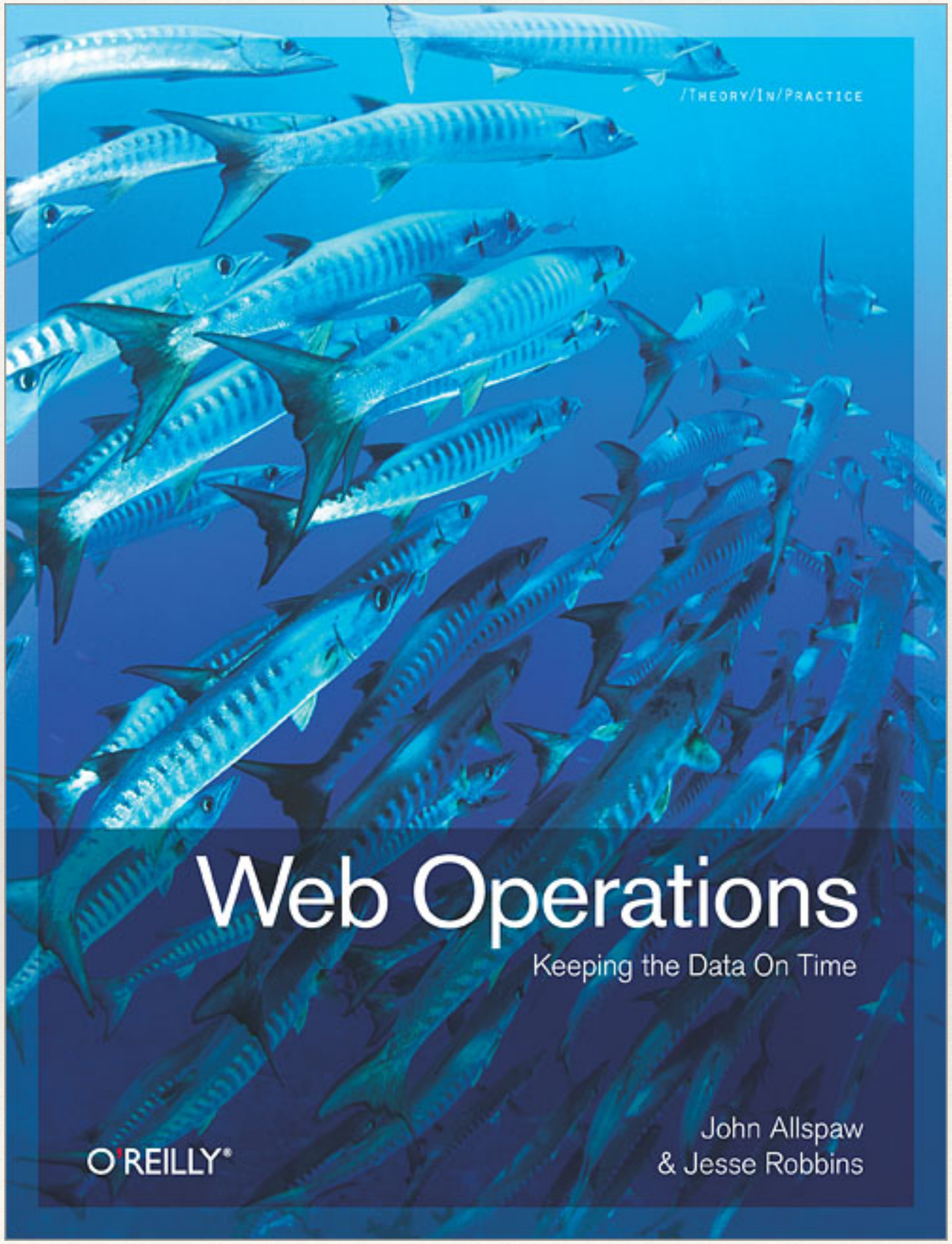
Logging activity

```
c = statsd.Counter('UserSignups')
c.increment()
```

What to track?

Technical Metrics:
Cache misses, images uploaded

Business Metrics:
Number of signups



Web Operations

Keeping the Data On Time

Caching: memcached, redis

ORM / Database: FK keys, DB queries, Haystack

Asynchronous Tasks: Celery

Logging / Monitoring: Sentry, Graphite

Thank You!

come chat with me anytime

@victorneo