INSTAGRAM UNDER THE HOOD

Django Under the Hood 2016

Carl Meyer

Instagram

Search

carl.j.meyer Follow

10 posts **84** followers **110** following

Carl Meyer

4,200,000,000,000

EVERY DAY

2,300,000,000,000,000

DJANGO REINHARDT

AND THE QUINTET OF THE HOT CLUB OF FRANCE

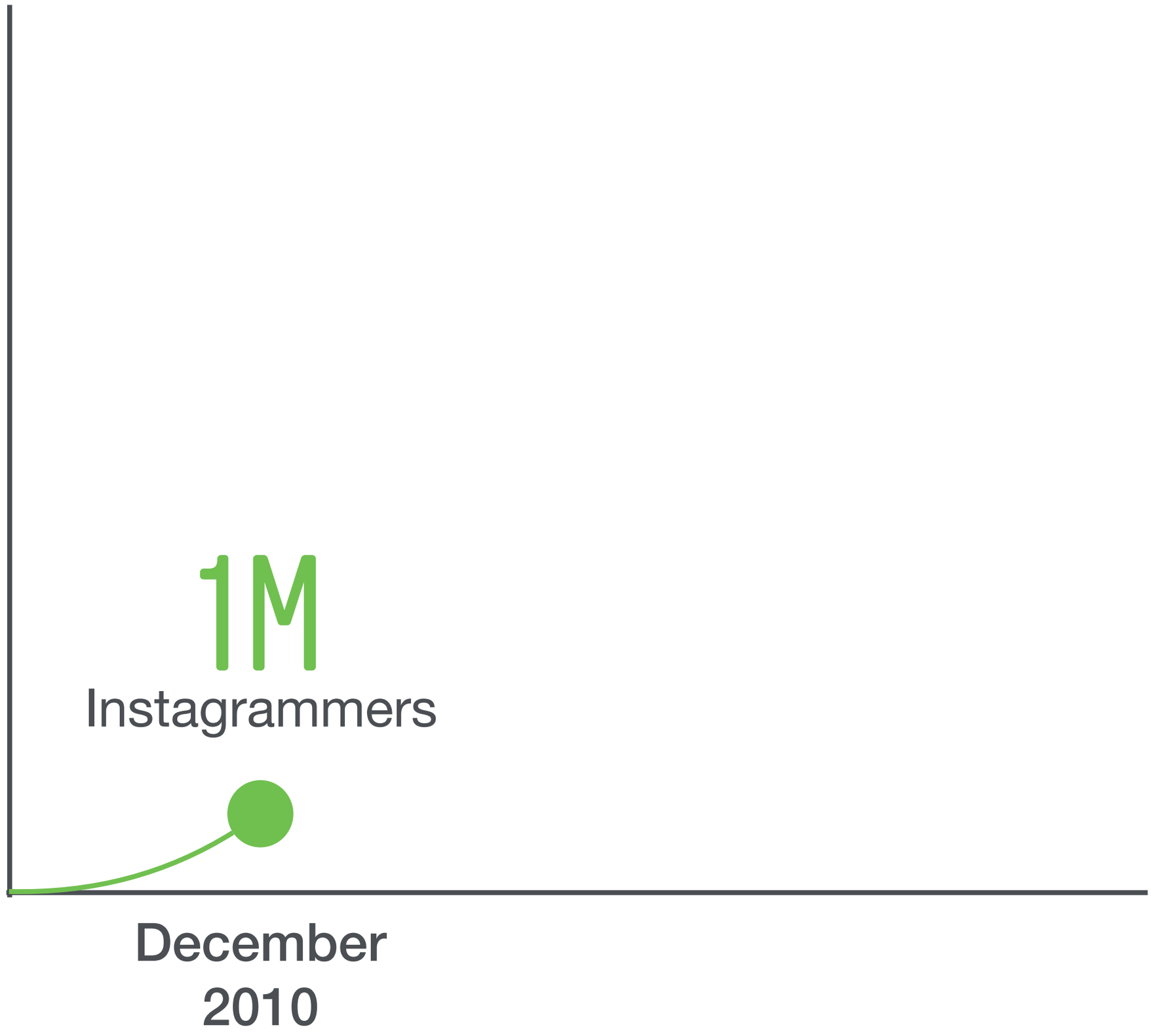
DJANGOLOGY

180g VINYL

October
2010

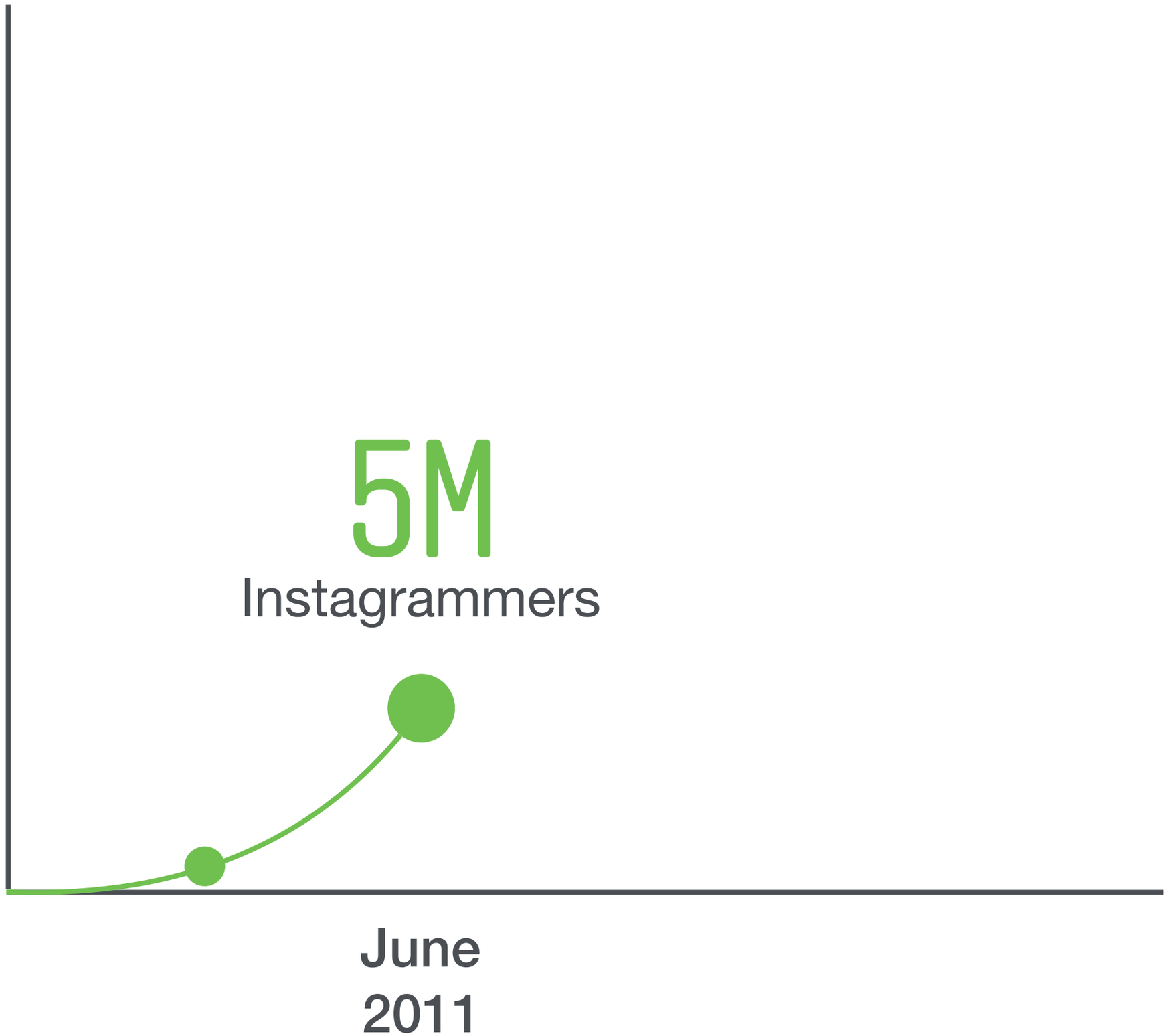
"SUPER EASY SET-UP...
ONE WAY OF DOING THINGS...
EASY TESTING."

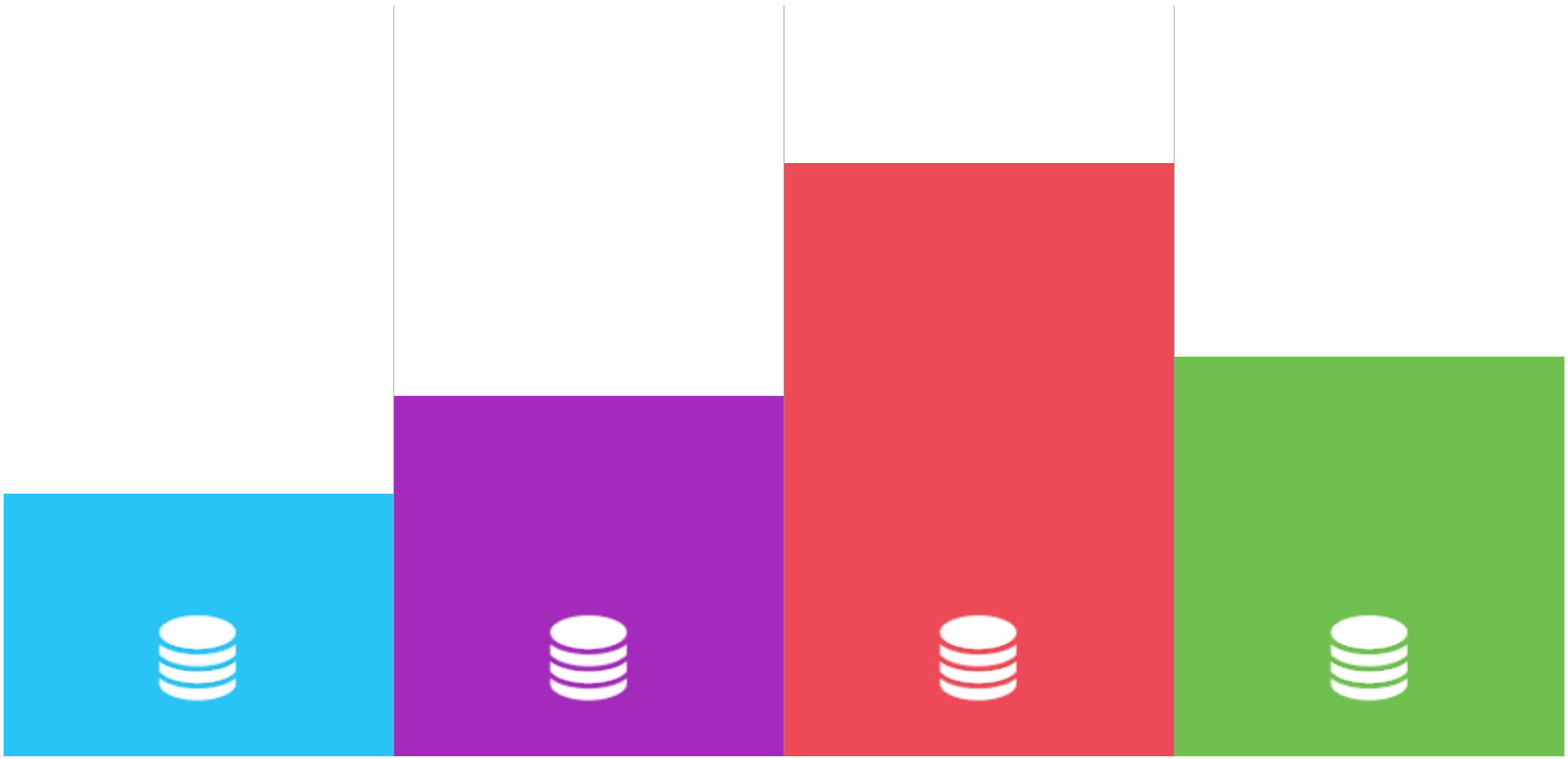
— Mike Krieger


1M
Instagrammers
December
2010


WIEN MUSE


IDLE

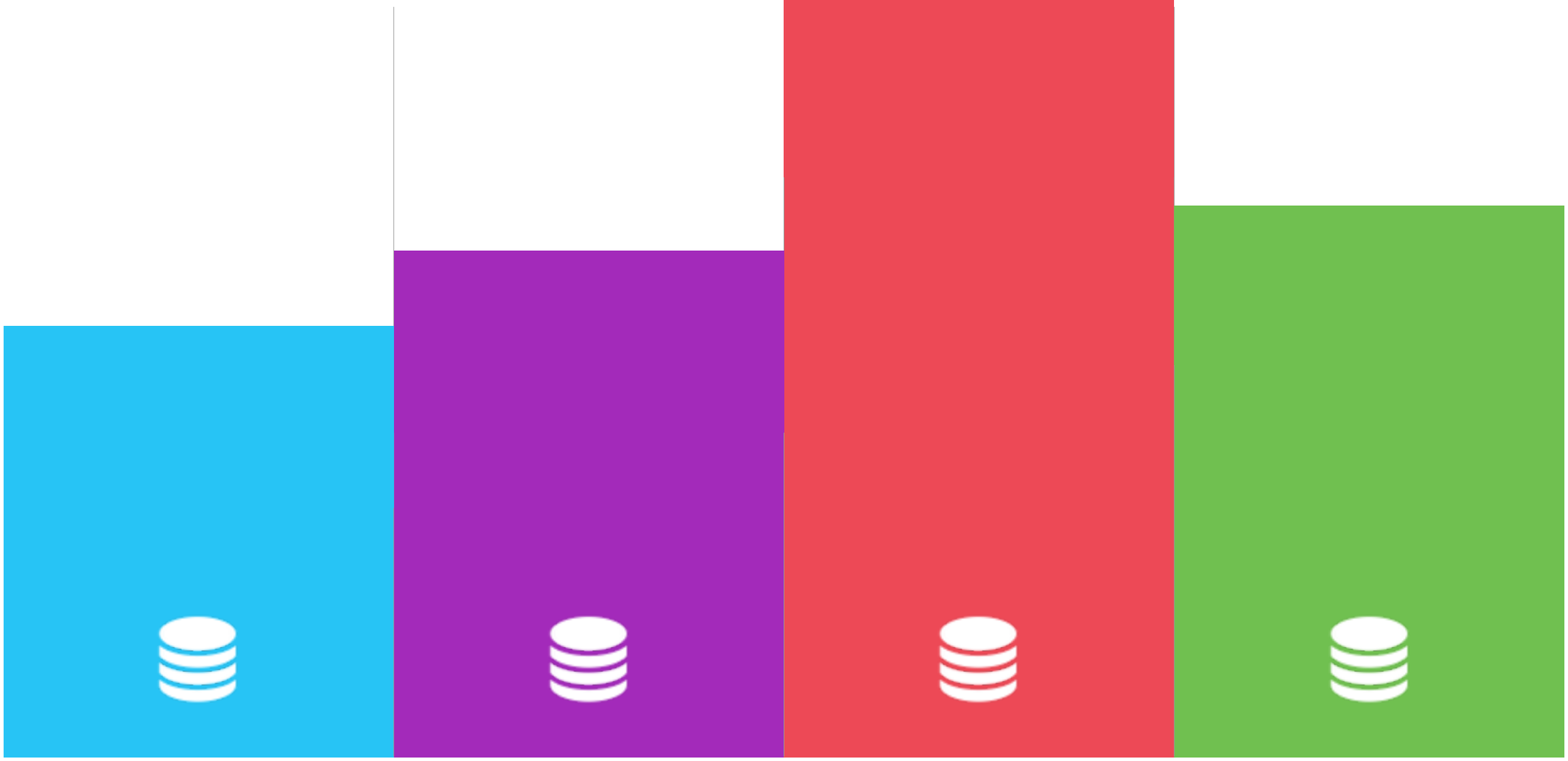

5M
Instagrammers
June
2011



USERS MEDIA LIKES COMMENTS

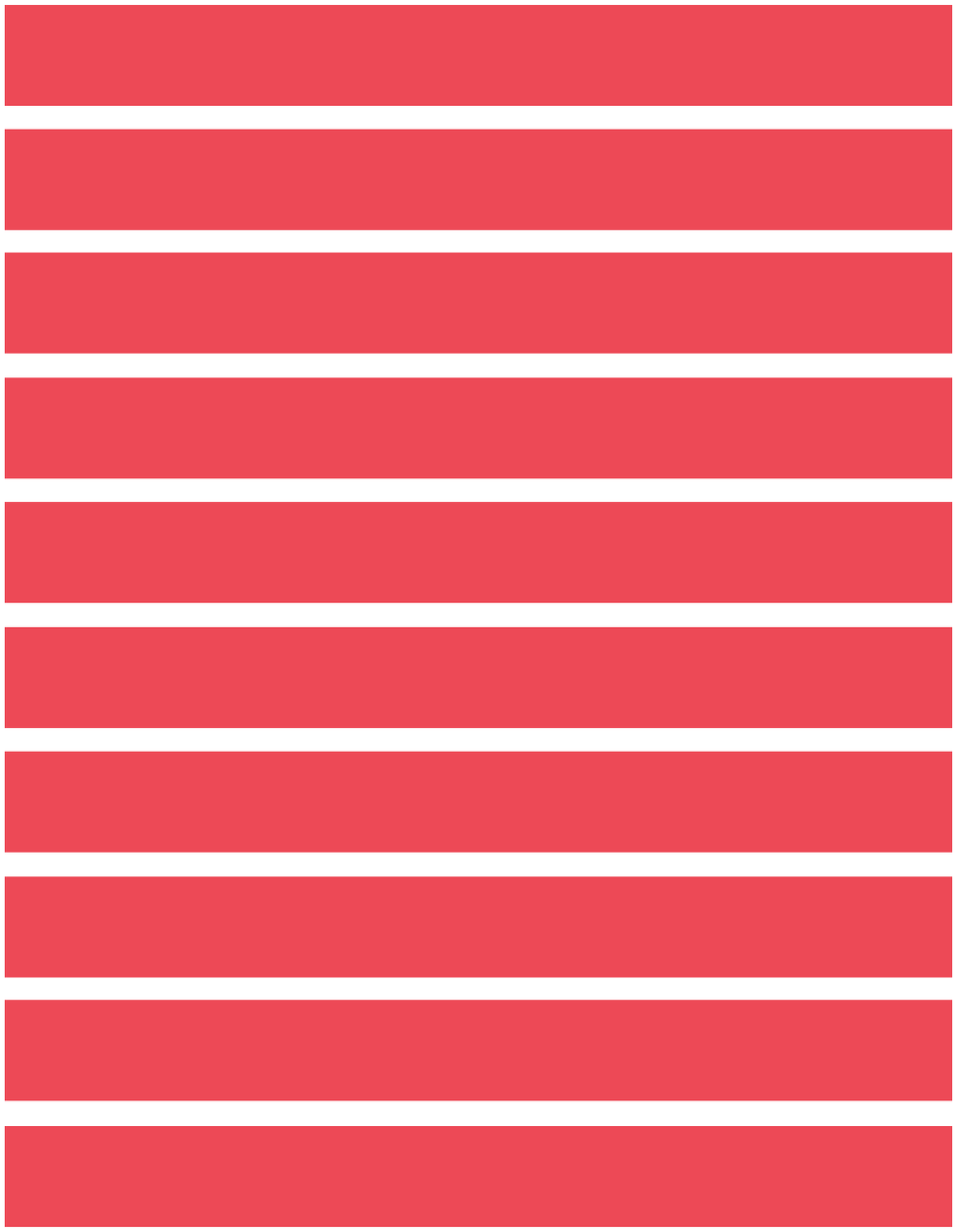
```python
class VerticalPartitionRouter(object):
    DB_FOR_MODEL = {
        'likes.like': 'likes',
        'comments.comment': 'comments',
        'media.media': 'media',
    }

    def _db_for(self, model_or_obj):
        label = model_or_obj._meta.label_lower
        return self.DB_FOR_MODEL.get(label, 'default')

    def db_for_read(self, model, **hints):
        return self._db_for(model)

    def db_for_write(self, model, **hints):
        return self._db_for(model)

    def allow_relation(self, obj1, obj2, **hints):
        return self._db_for(obj_1) == self._db_for(obj_2)
```

```
  501  ihst
  502  history
alex@alex-laptop:~$ hg log
alex@alex-laptop:~/projects/pypy$ ^Cory here (.hg n
alex@alex-laptop:~/projects/pypy$ python pytest.py
ainterp/test/test_optimizeopt.py
========== test session starts ==========
platform linux2 -- Python 2.6.6 -- pytest-2.0.3.dev
pytest-2.0.3.dev1 from /home/alex/projects/pypy/pyt
collected 236 items

pypy/jit/metainterp/test/test_optimizeopt.py ......
.....
.....
.....

========== 235 passed, 1 skipped in 2.06 seconds ==========
lex-laptop:~/projects/pypy$
```



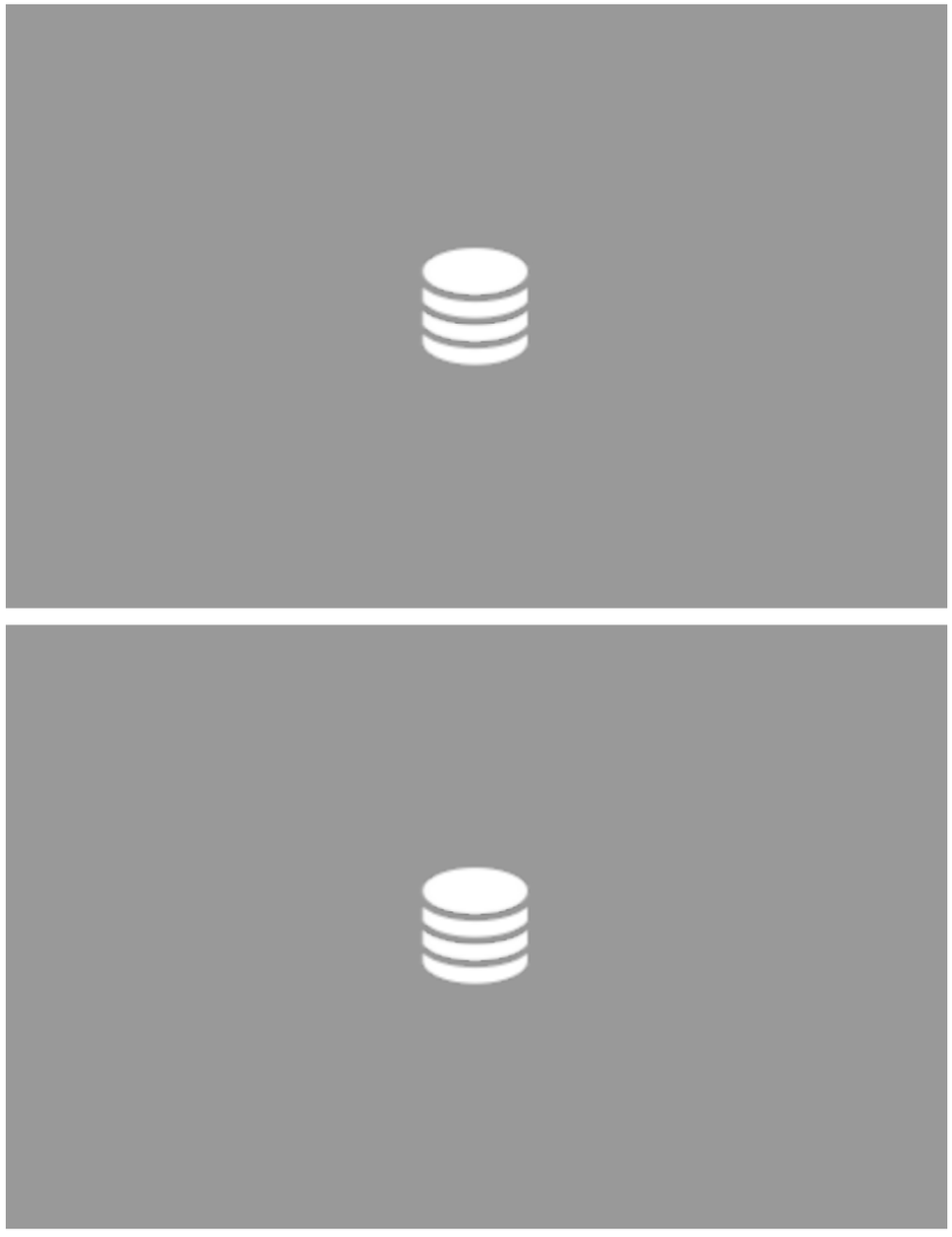
USERS MEDIA LIKES COMMENTS

LOGICAL SHARDS

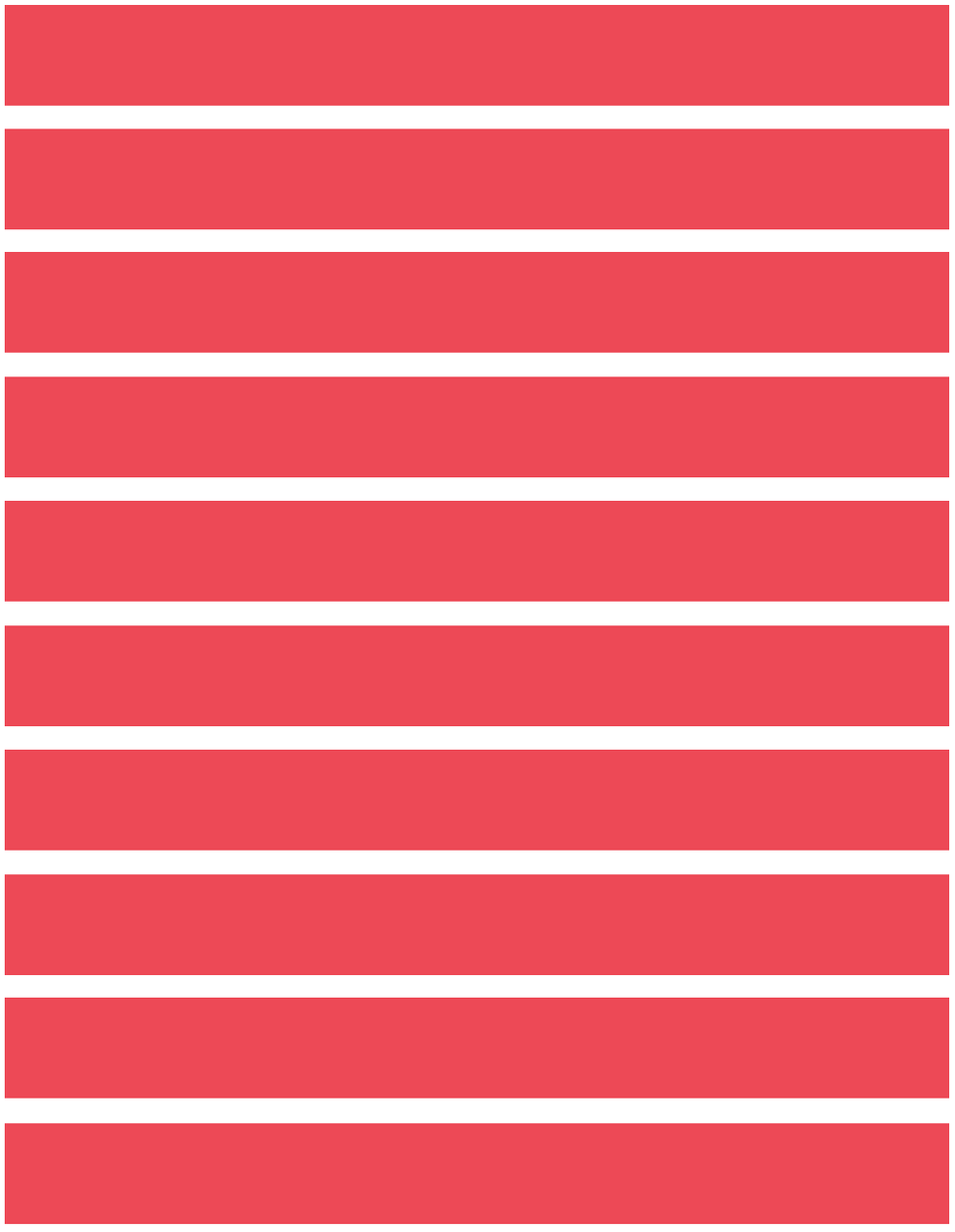


(PG SCHEMAS)

PHYSICAL SERVERS

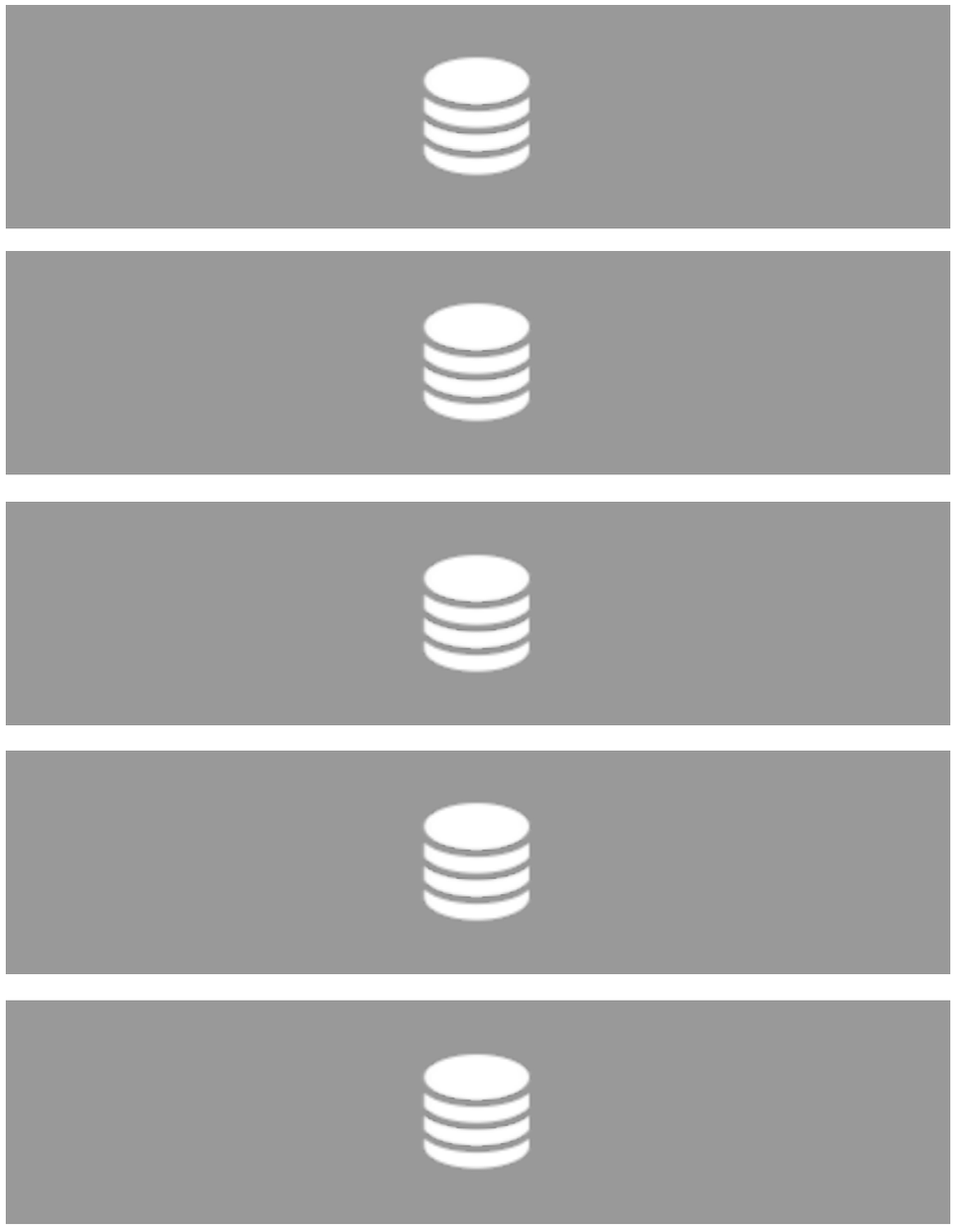


LOGICAL SHARDS



(PG SCHEMAS)

PHYSICAL SERVERS



commit 5c7034fa8b934569cce5c1bf4bb202f2f3f18bc9

Author: Mike Krieger

Date: Tue Jul 19 23:47:26 2011 -0700

WIP

```
class ShardedObject(object):
    def insert(self, shard_on_id, from_table, values):
        shard, db = get_conn_for_shard_key(shard_on_id)
        cursor = db.cursor()
        placeholders = ','.join(
         [("%%(%s)s" % key) for key in values.keys()])

        columns = ','.join(values.keys())
        insert_statement = (
         "INSERT INTO idb%s.%s (%s) VALUES (%s)" %
         (shard, from_table, columns, placeholders)
        )

        cursor.execute(insert_statement, values)
        db.commit()
```

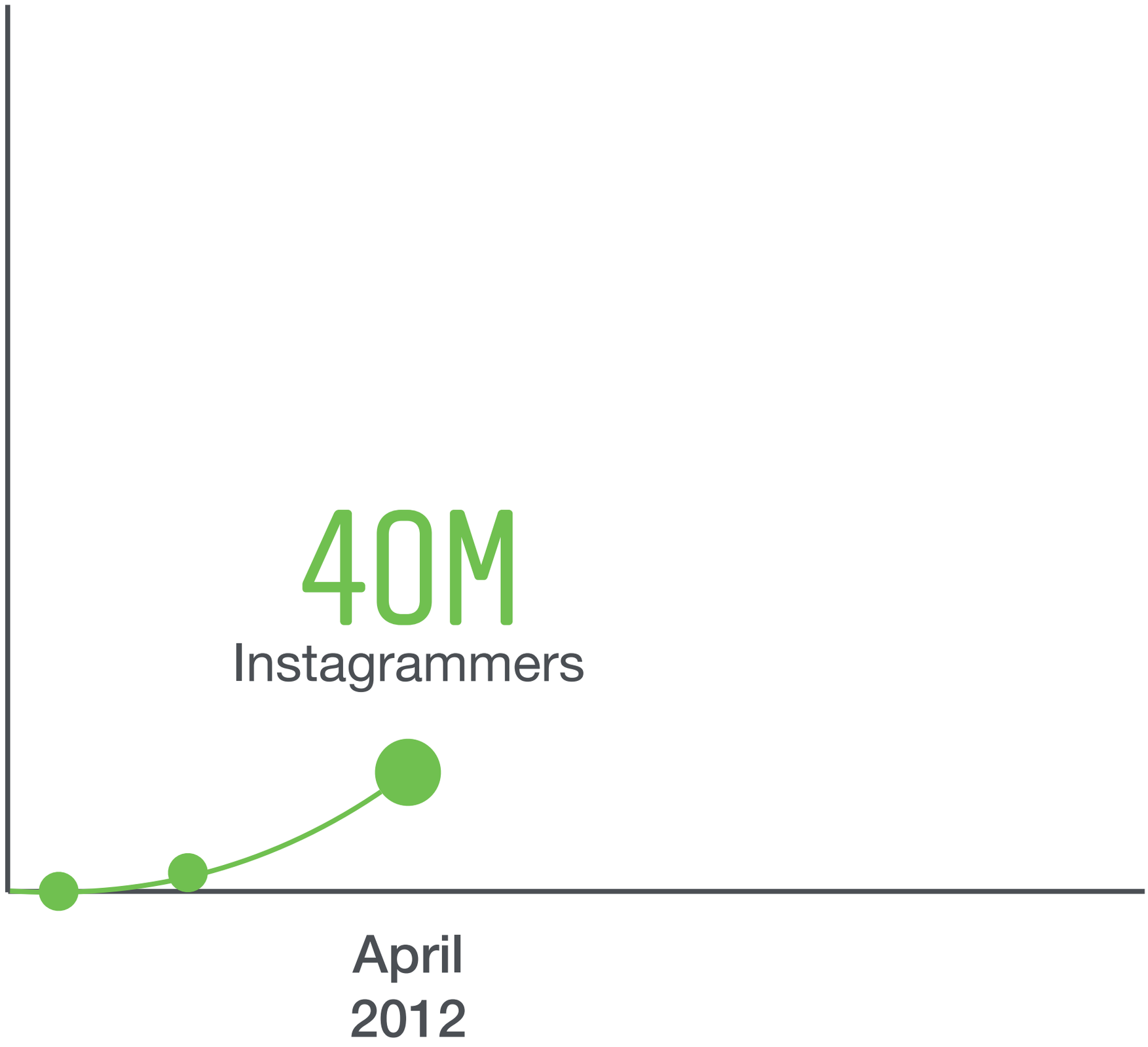
SHARDED UNIQUE IDS

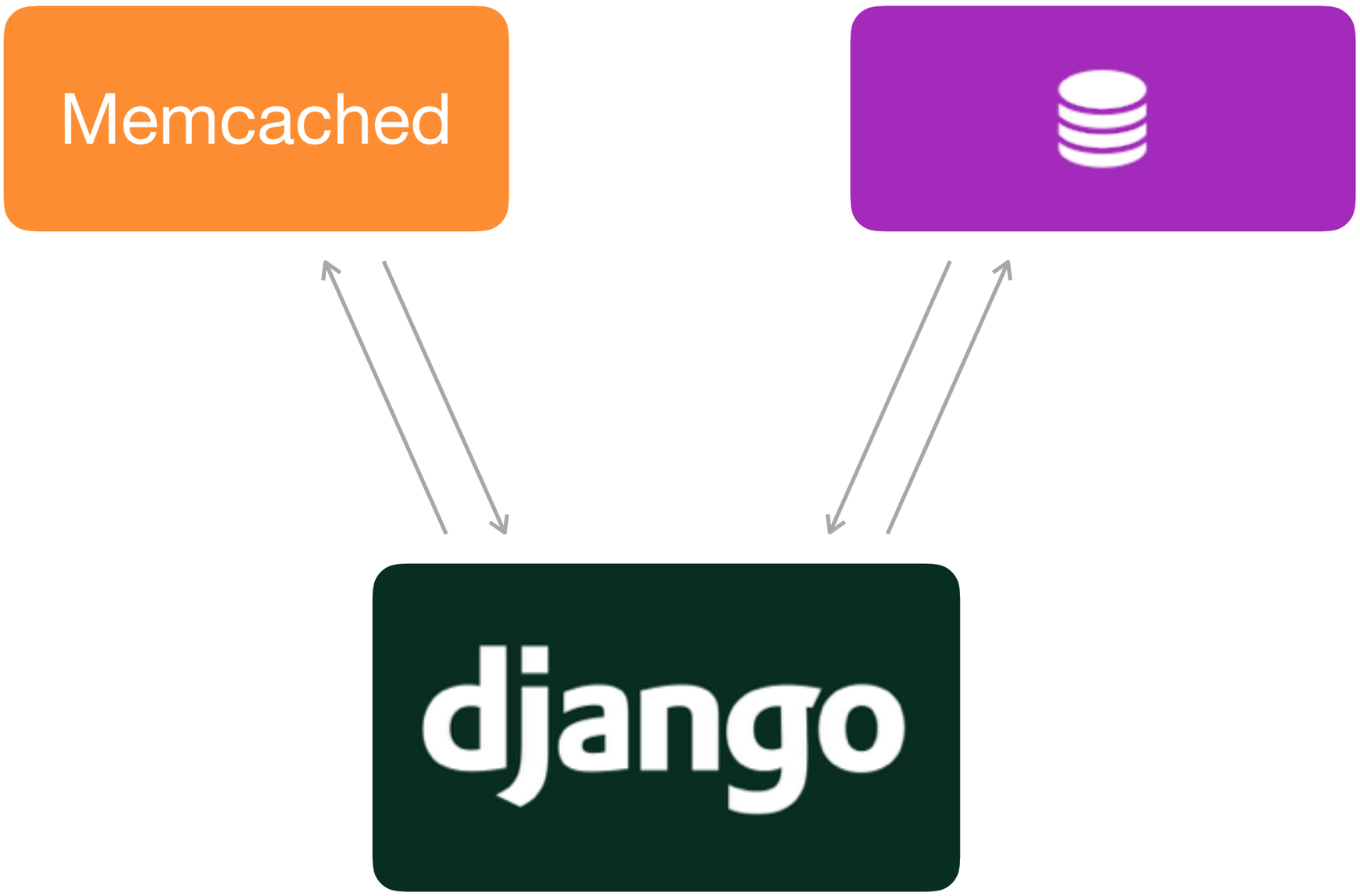
1387263000013410905

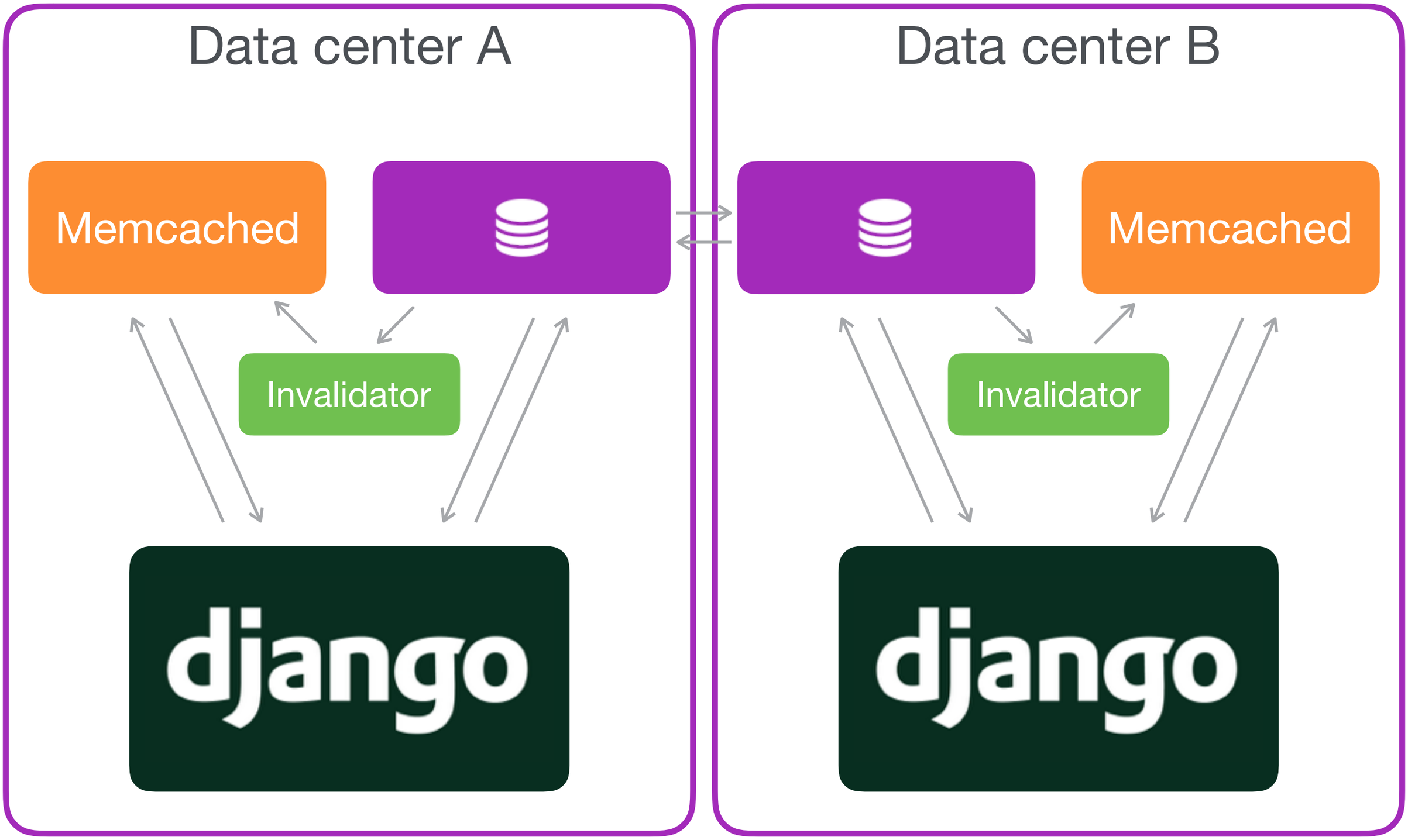
TIMESTAMP SHARD ID SEQUENCE

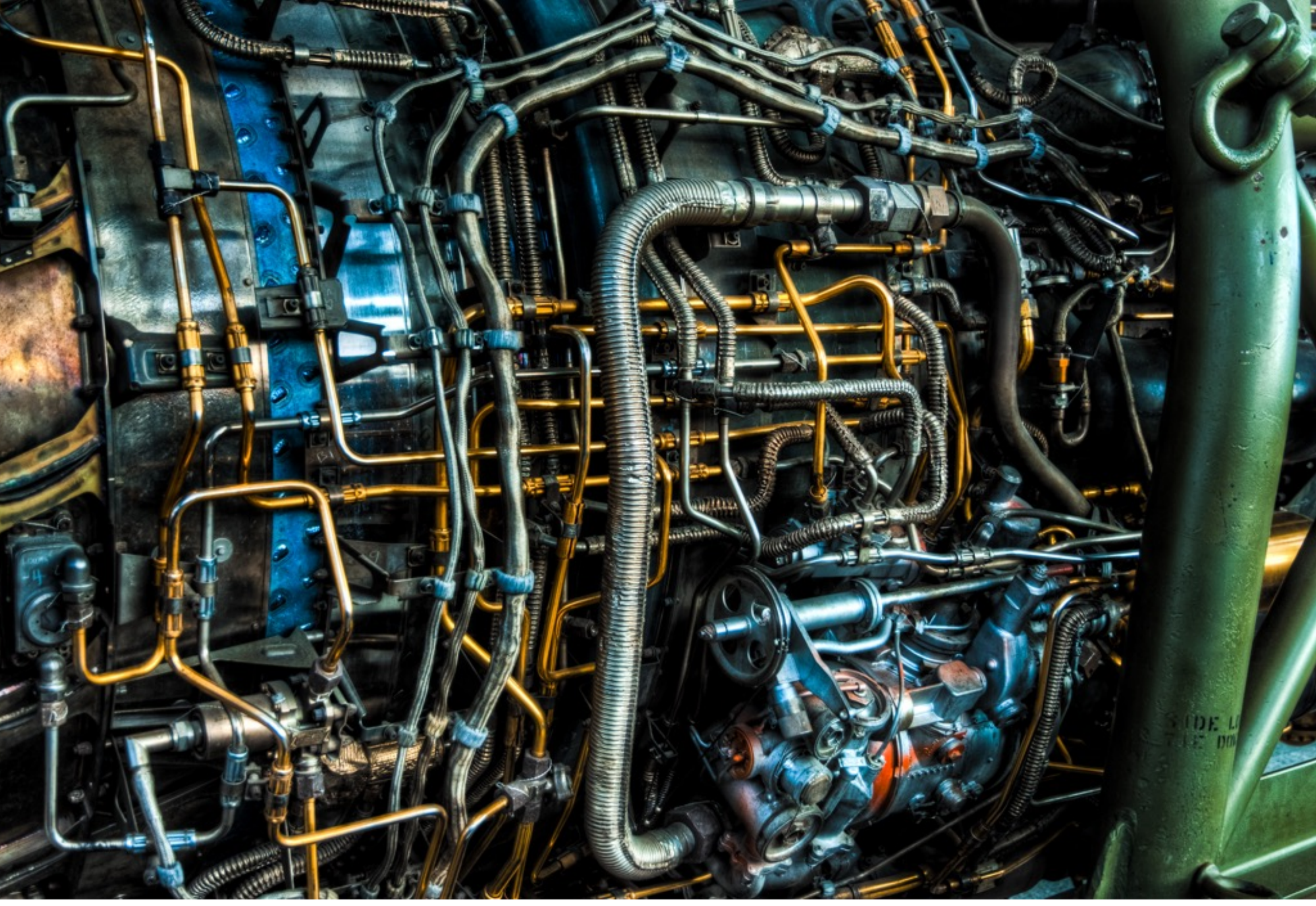
```
CREATE OR REPLACE FUNCTION insta5.next_id...
CREATE TABLE insta5.our_table (
    "id" bigint NOT NULL DEFAULT insta5.next_id(),
    ...rest of table schema...
  )
```


JUSTIN
BIEBER


40M
Instagrammers
April
2012

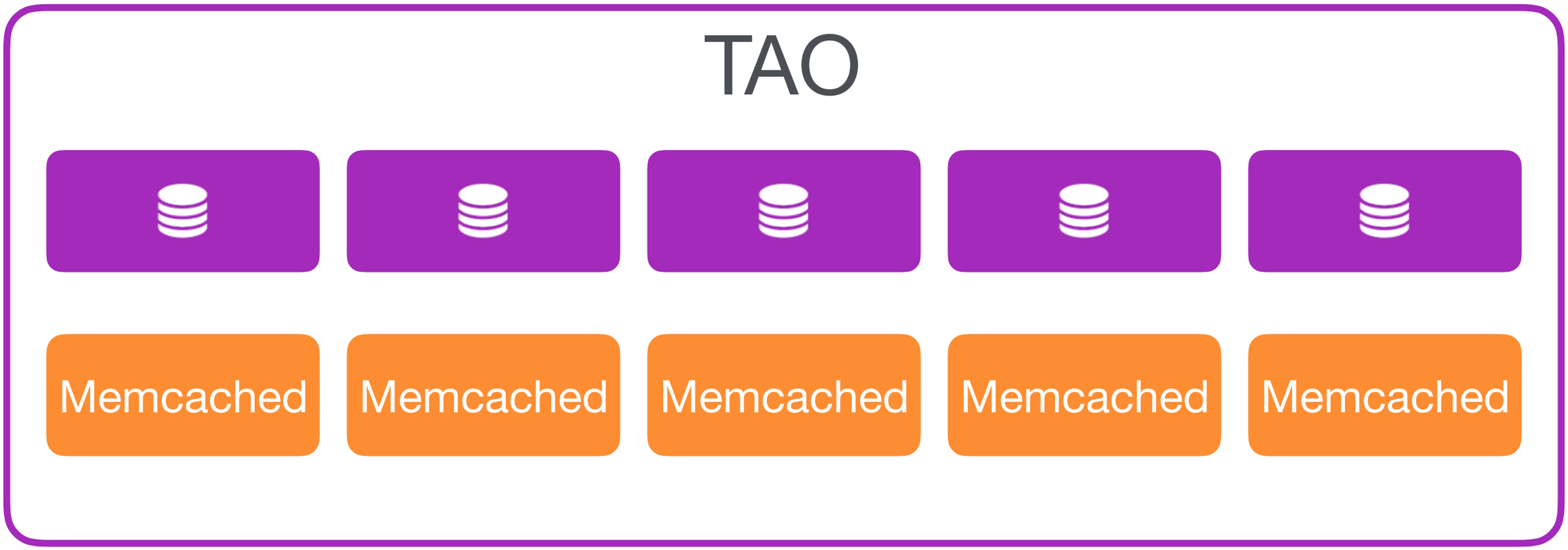

Memcached
django


Data center A
Memcached
Invalidator
django
Data center B
Memcached
Invalidator
django



MULTI-REGION CACHE INVALIDATION

CONTEMPLATING THE TAO


TAO
Memcached
Memcached
Memcached
Memcached
Memcached


django

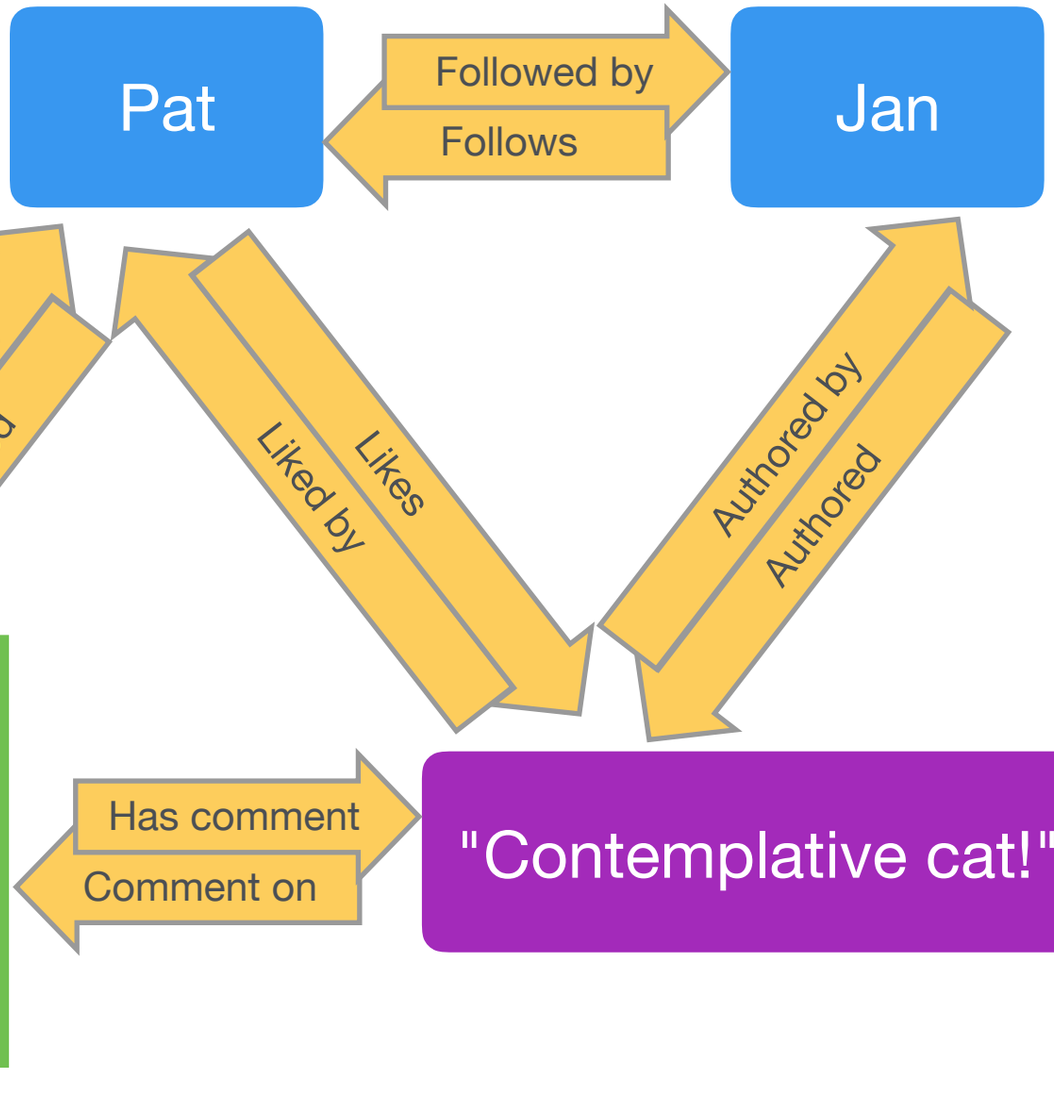
TAO DATA MODEL

Jan follows Pat.

Pat posts a **photo**.

Jan authors a **comment** on the **photo**.

Pat likes the **comment**.



CONTEMPLATING THE TAO

500M
Instagrammers

June
2016

"JUST KEEP FIXING UNTIL THE TESTS PASS."

INSTAGRAM:

Now compatible with

Django 3.1™

(1.3 + 1.8)

INSTAGRAM:

Now compatible with

Django

1.8!

OUR (MONKEY) PATCHES

1. Don't recompile URL regexes for every active language.

---

2. Don't try to load translations from an app with no locale directory.

---

3. Unlazified settings!

UNLAZY ALL THE SETTINGS!

```
from django.conf import settings

def force_unlazified_settings():
    for key in dir(settings):
        settings.__dict__[key] = getattr(settings, key)
```

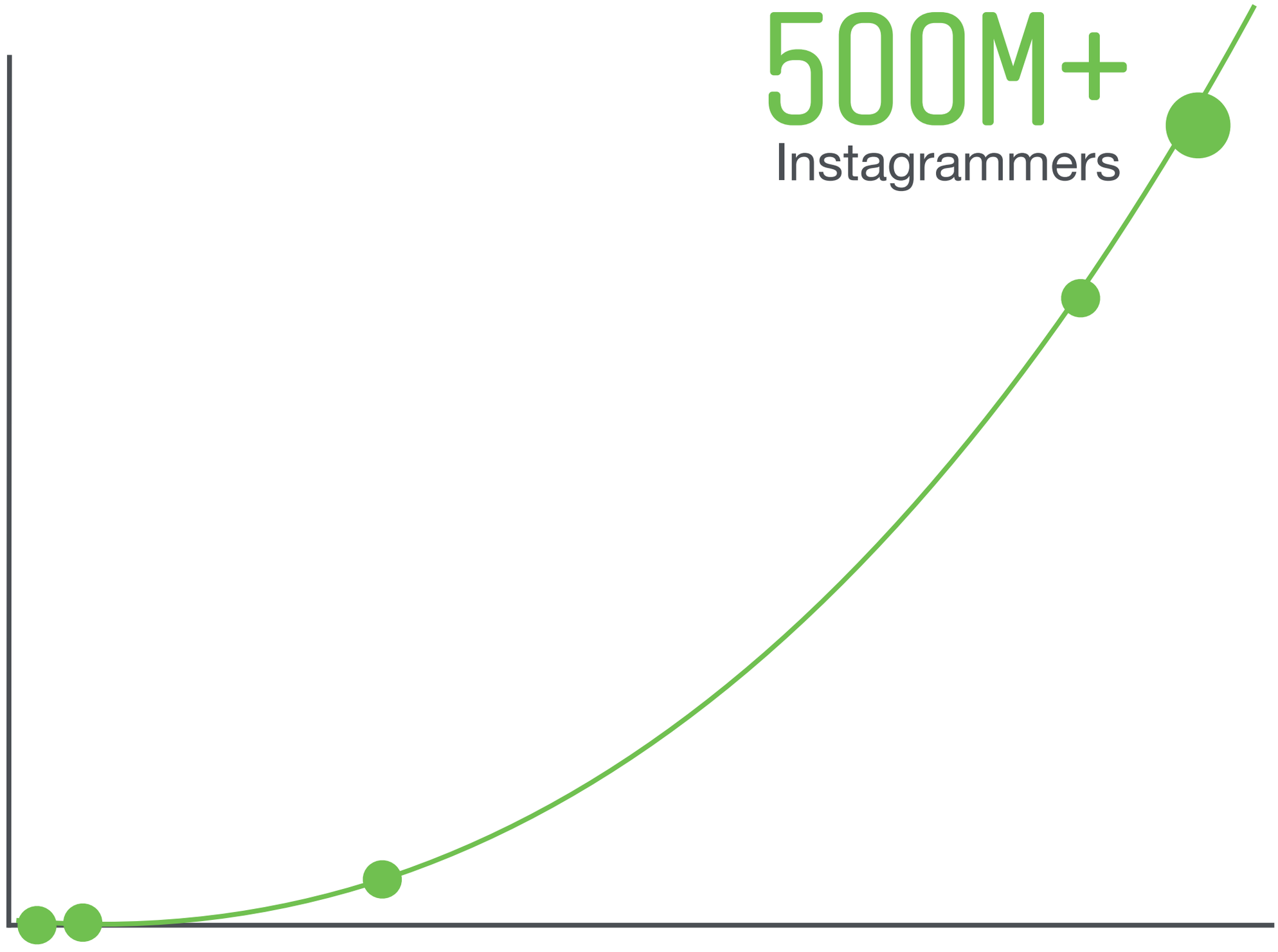
INSTAGRAM:

Now compatible with

Django

1.8!

(and fast as ever)


500M+
Instagrammers
Today!

Proxygen

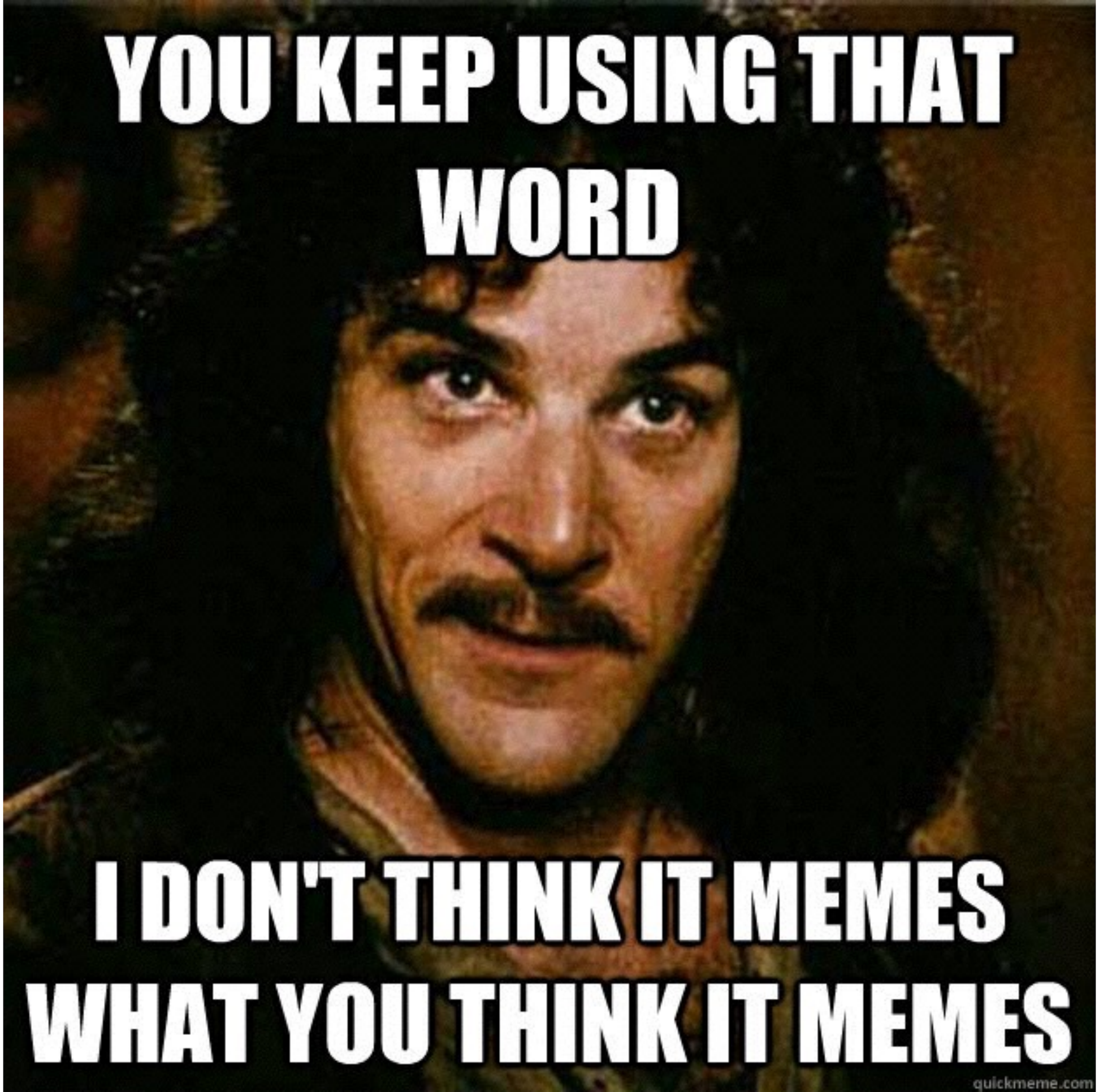
Django & uWSGI

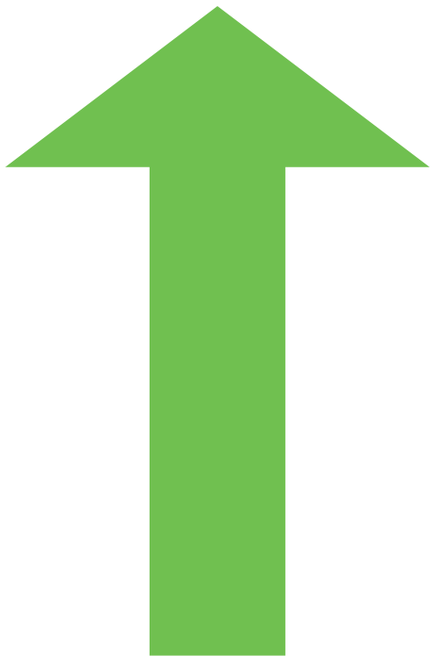
TAO

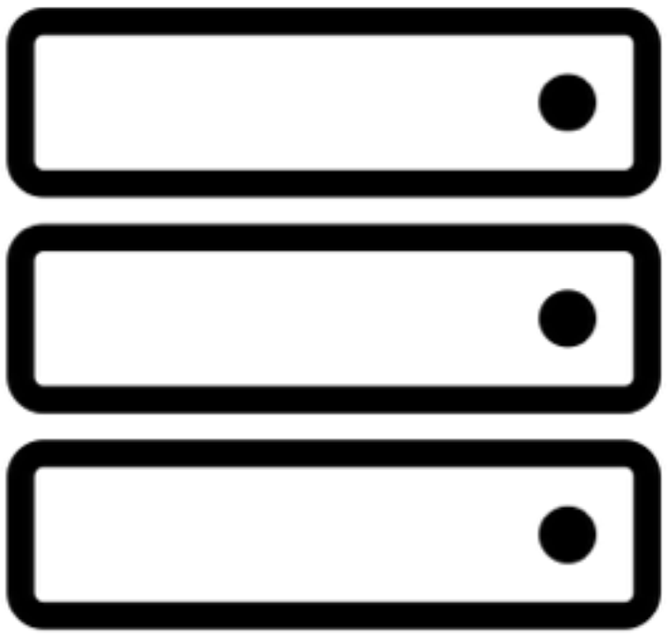
Cassandra

Everstore

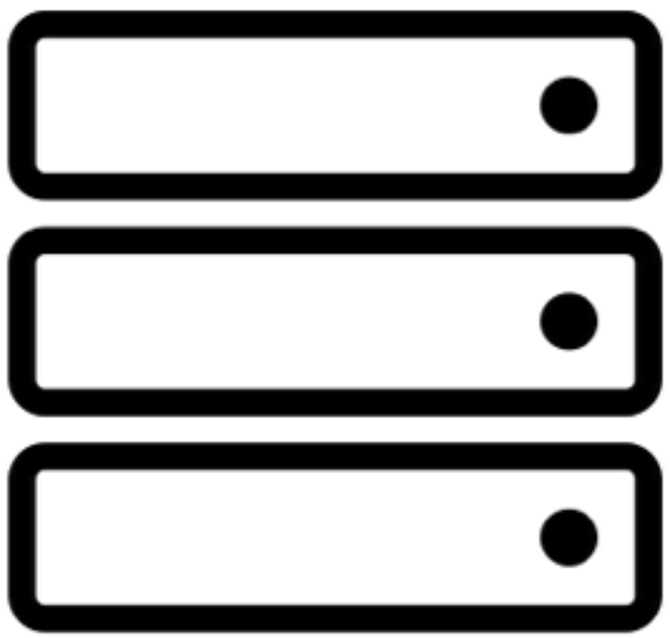
Celery &
RabbitMQ


YOU KEEP USING THAT WORD
I DON'T THINK IT MEMES WHAT YOU THINK IT MEMES
quickmeme.com





Active Last Minute



???

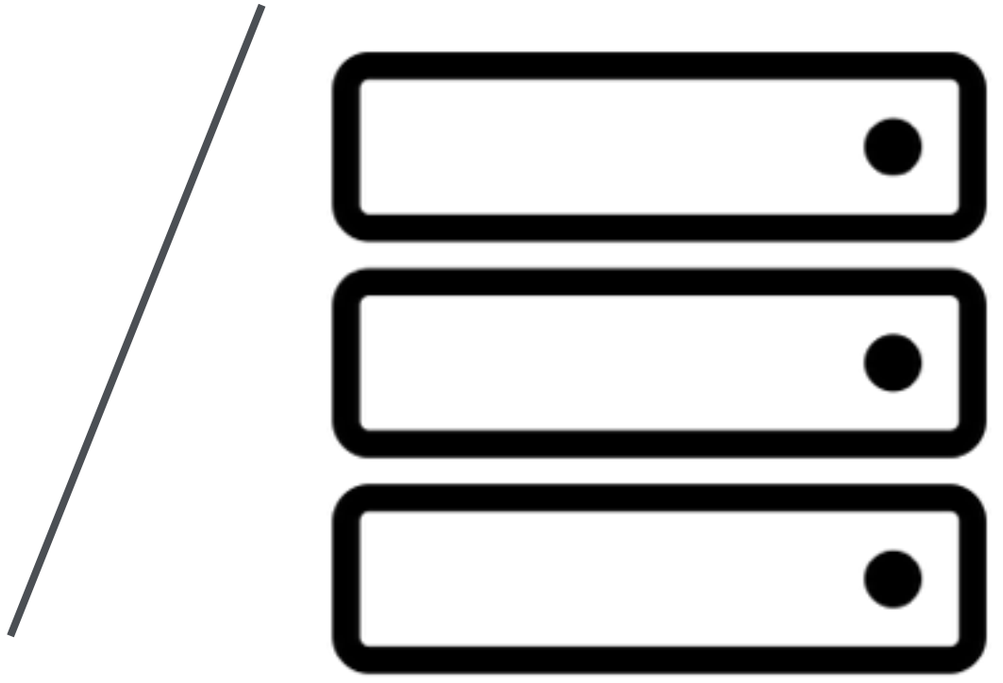
COUNTING CPU INSTRUCTIONS WITH PERF

```
struct perf_event_attr pe;
pe.type = PERF_TYPE_HARDWARE;
pe.config = PERF_COUNT_HW_INSTRUCTIONS;
fd = perf_event_open(&pe, 0, -1, -1, 0);
ioctl(fd, PERF_EVENT_IOC_ENABLE);
// code whose CPU instructions you want to measure
ioctl(fd, PERF_EVENT_IOC_DISABLE);
read(fd, &count, sizeof(long long));
```

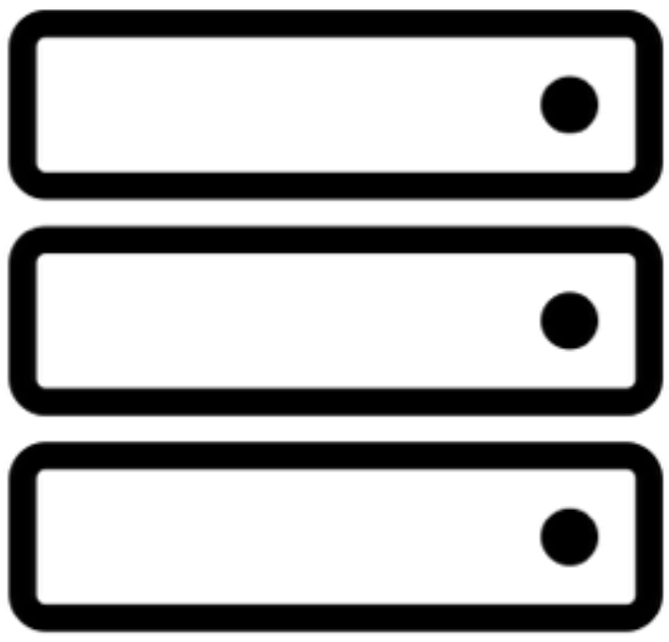
CPU instructions/s

CPU instructions/s

CPU instructions/s



CPU instructions/s



Wed, Oct 26th, 2016 From: 2016

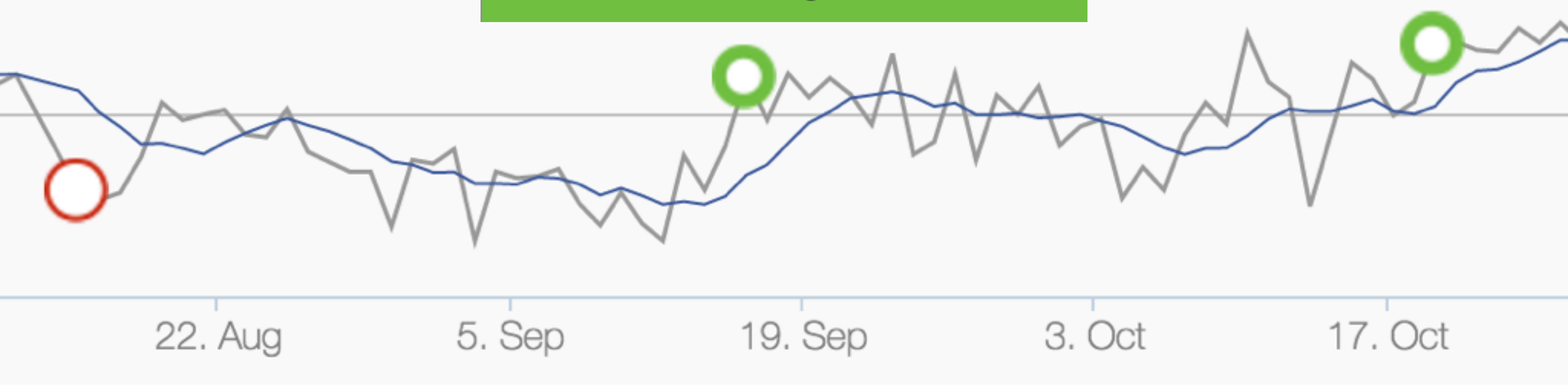
IG Weight

Instagram

17.9%

Improvement

Latest Goal:
≤ 13.1% regression



AppWeight


Continuous deployment
30-50 deploys per day


TURN TO CLEAR VISION
QUARTERS ONLY
25¢
1. DROP COIN IN SLOT
2. PULL HANDLE DOWN ALL THE WAY-LET GO
3. TO CLEAR VISION - TURN RED KNOB
2414

DYNOSTATS

```
class DynostatsMiddleware(object):
  def process_request(self, req):
    req.dynostats_enabled = (
      1 == random.randint(1, settings.DYNO_SAMPLE_RATE))
    if req.dynostats_enabled:
      # uses Linux perf library
      req.dyno_start_cpu_instr = get_cpu_instructions()
      # use clock_gettime from librt
      req.dyno_start_wall_time = get_real_wall_time()
      req.dyno_start_cpu_time = get_process_cpu_time()
      # uses /proc/<pid>/statm
      req.dyno_start_rss_mem = get_process_rss_mem()

  def process_response(self, req, response):
    if req.dynostats_enabled:
      # get end values, send to scribe w/ req details
    return response
```

TimeSeries (10 minutes) ordered by Hits

CPU regression happens here

1400
1200
1000
800
600
400
200
0

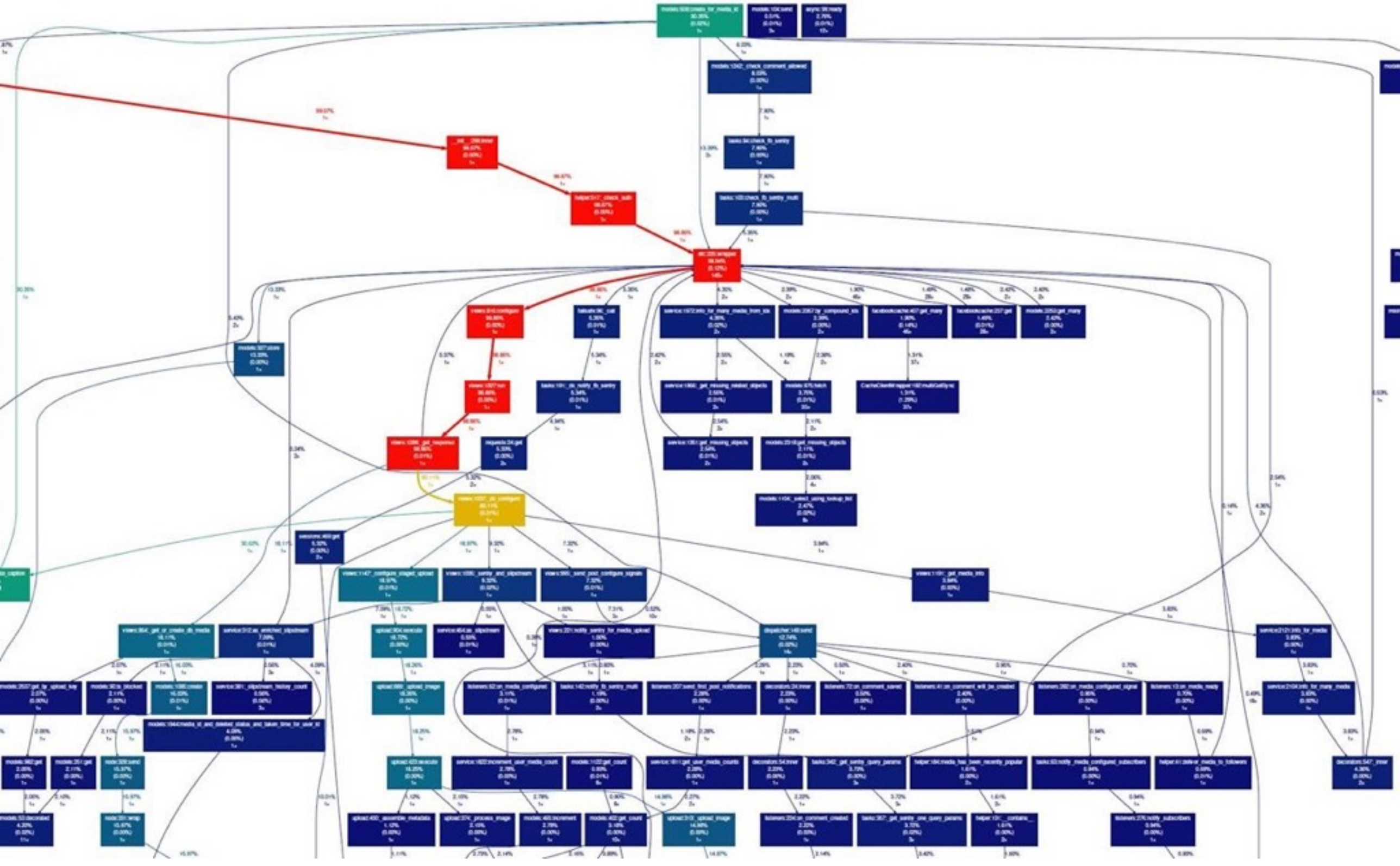
Wed Dec 02
UTC-08:00
Thu Dec 03
Fri Dec 04
Sat Dec 05
Sun Dec 06
Mon


ALERT!

CPROFILE

```
class ProfilerMiddleware(object):
   def process_request(self, req):
      req.cprofile_enabled = (
         1 == randint(1, settings.CPROFILE_SAMPLE_RATE))
      if req.cprofile_enabled:
         req.profiler = cProfile.Profile()
         req.profiler.enable()

   def process_response(self, req, response):
      if req.cprofile_enabled:
         req.profiler.disable()
         req.profiler.create_stats()
         send_to_scribe(msgpack.dumps(profiler.stats))
```

Raw Cumulative CPU Instructions (Sum/Minute) [?]	(compare)	(%)	(delta)
8,034,035,705	8,043,161,111	-0.1%	-9,125,406
7,878,265,891	7,897,286,292	-0.2%	-19,020,400
5,339,241,762	5,336,959,292	0.0%	2,282,470
2,175,683,992	2,193,166,866	-0.8%	-17,482,874
1,478,319,569	1,505,121,058	-1.8%	-26,801,489
637,941,053	375,612,446	69.8%	262,328,607
347,426,732	341,689,652	1.7%	5,737,080
290,604,317	245,609,794	18.3%	44,994,524
118,035,772	118,197,518	-0.1%	-161,745
1,930,530	1,919,097	0.6%	11,433



Global CPU consumption by this function: 0.103% inclusive and 0.015% exclusive .
We need ~[redacted] IG servers for this function!

Drill Down

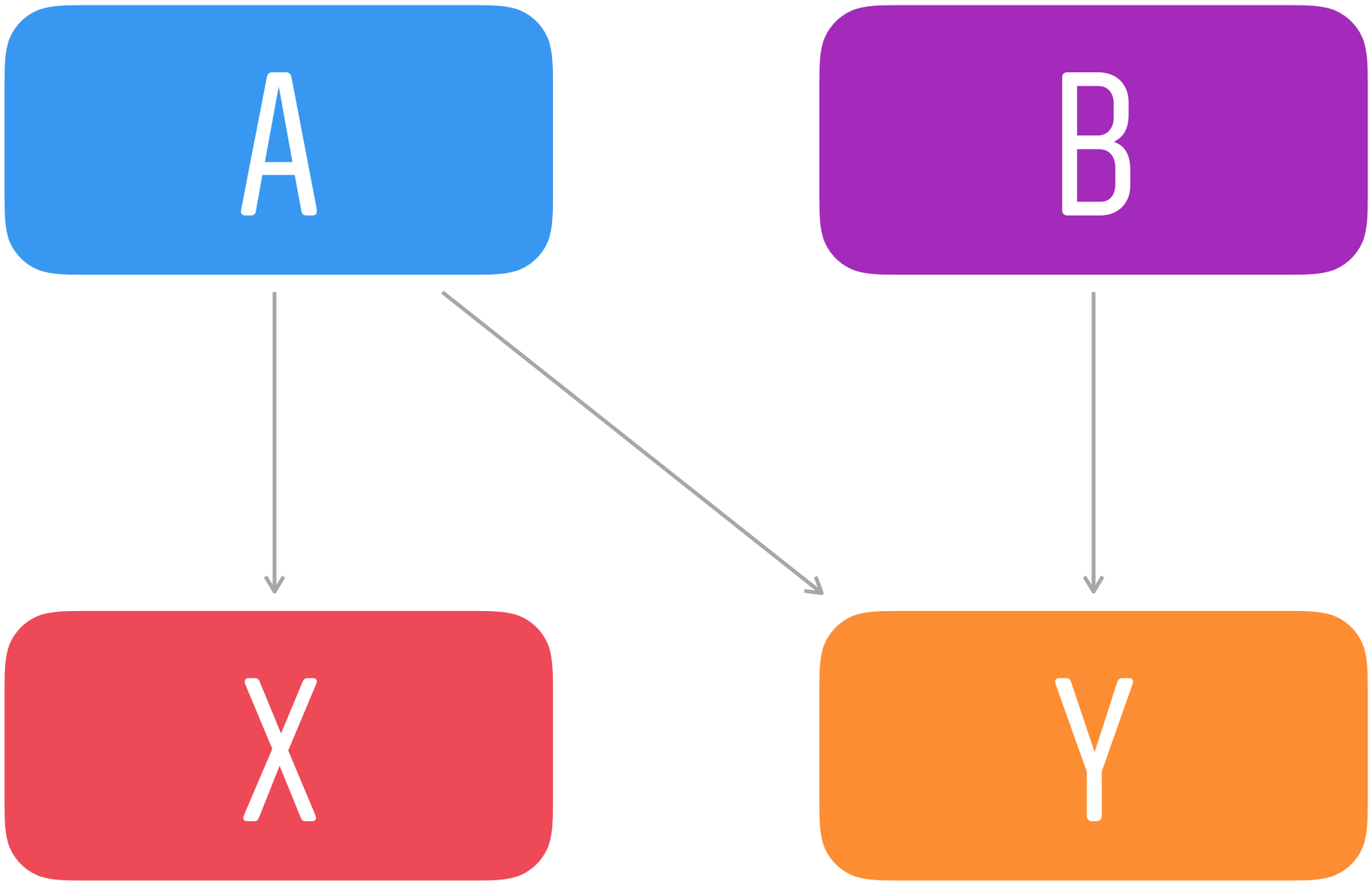
Top Views (see all):
91.4% by discover.api.views.explore
5.1% by discover.api.views.popular
3.5% by channels.api.views.channel_viewer

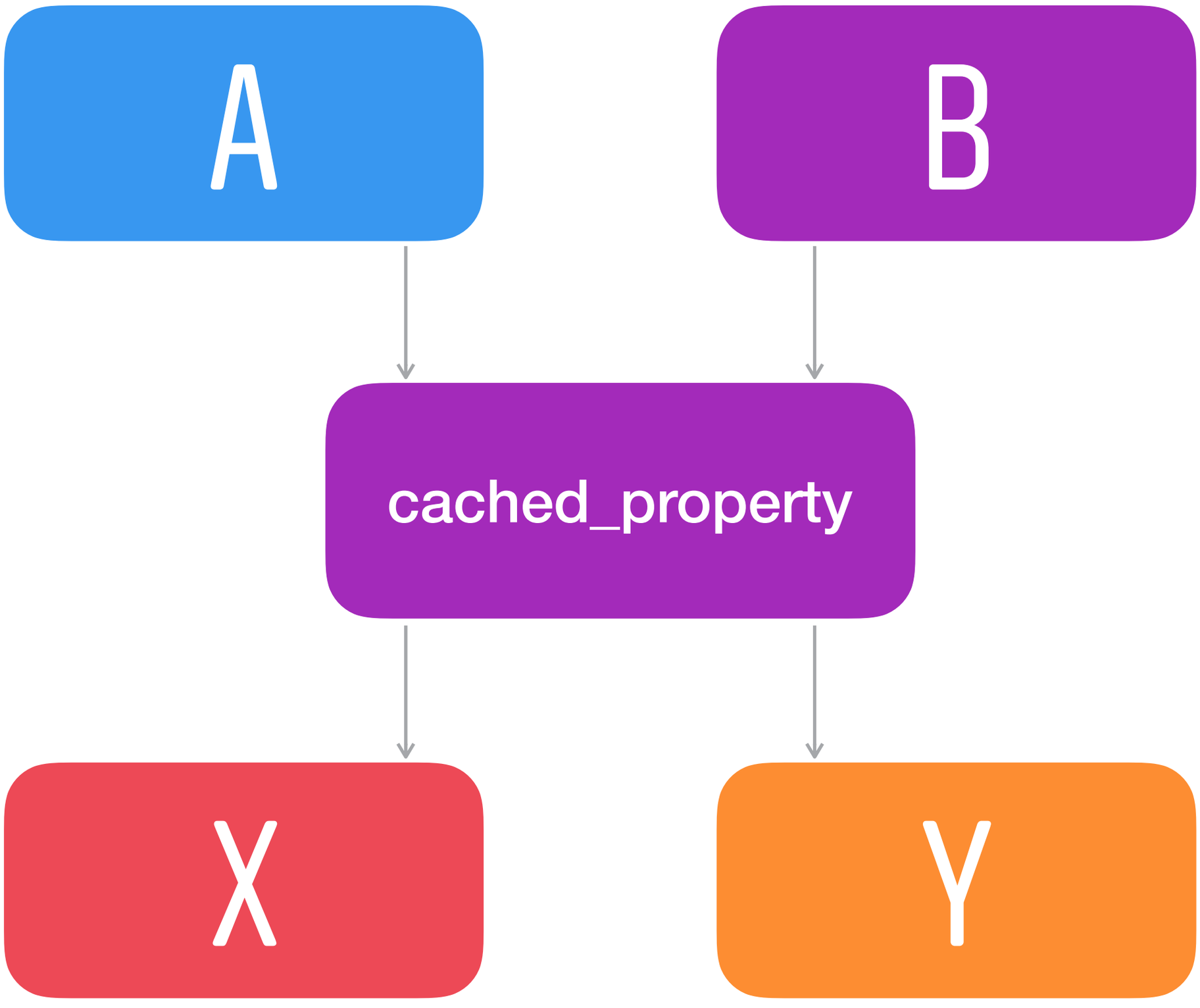
Top Callers (see all):
96.5% by _post_process_explore_media_response in discover/services/service.py
3.5% by _unicorn_response_to_bundles in discover/services/service.py
0.1% by _fetch_videos_from_search in fbsearch/event_video_search.py

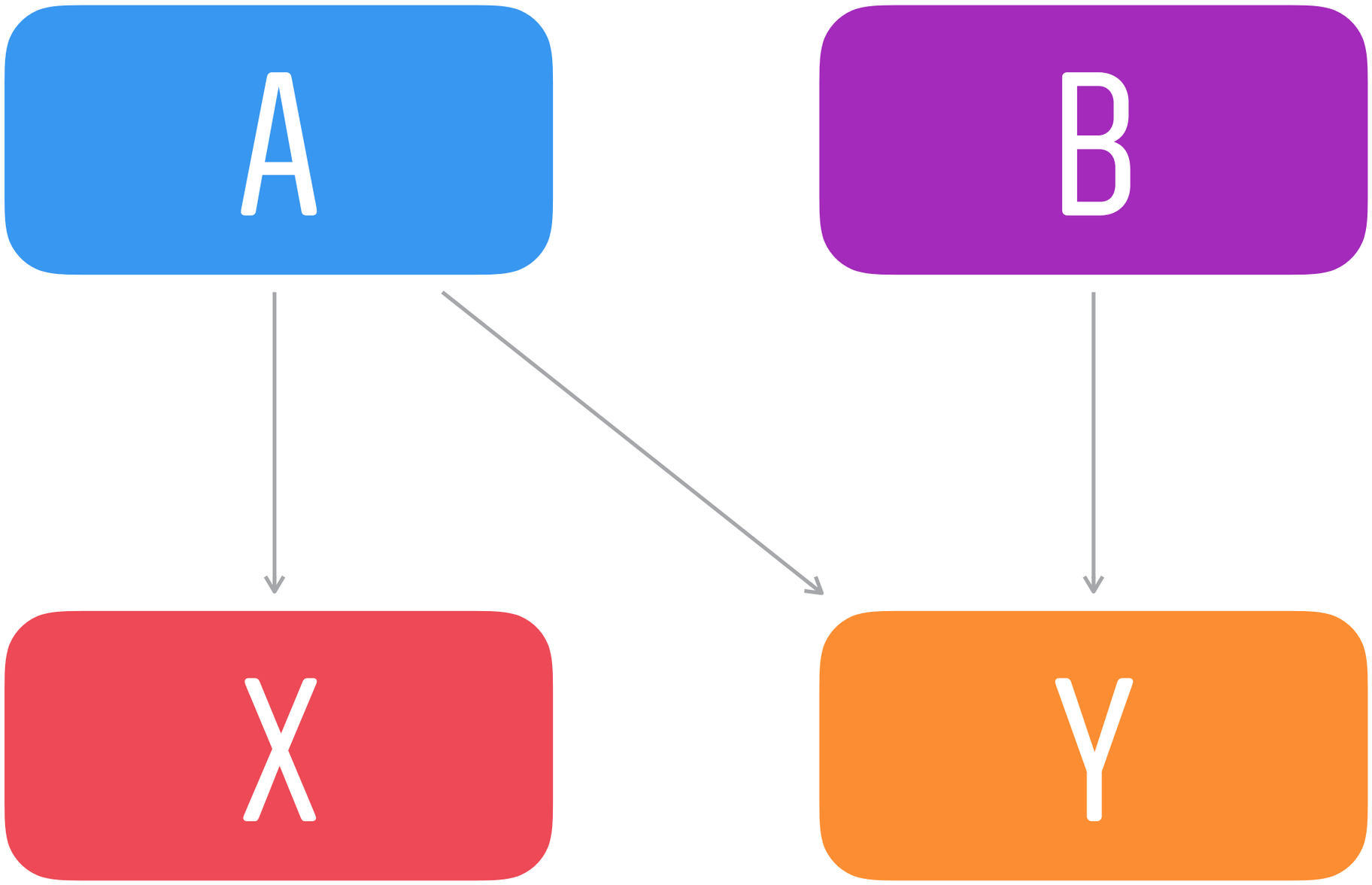
```
# on
@statsd.timer("discover._get_bundles_from_search_response")
def _get_bundles_from_search_response(self, search_response, has_connection_info=False):
    """
    :type search_response: fbsearch.media_search.MediaSearchResponse
```

CUSTOM CPROFILE TIMERS

```
import cProfile
import resource


def get_cpu_instr():
   # use perf to get CPU instructions

cpu_profiler = cProfile.Profile(timer=get_cpu_instr)


def get_rss_mem():
   return resource.getrusage(
     resource.RUSAGE_SELF).ru_maxrss

mem_profiler = cProfile.Profile(timer=get_rss_mem)
```


A
B
X
Y


A
B
cached_property
X
Y


A
B
X
Y

FIXING EFFICIENCY REGRESSIONS

- Fixing the obvious.
- Don't do useless work.
- Cache things that don't change.
- Change a .py to a .pyx: Cython.
- Rewrite in C.


IC
799221
IC
799221
tightly integrated
loosely coupled


make the easy things easy
and the hard things possible

"SUPER EASY SETUP."

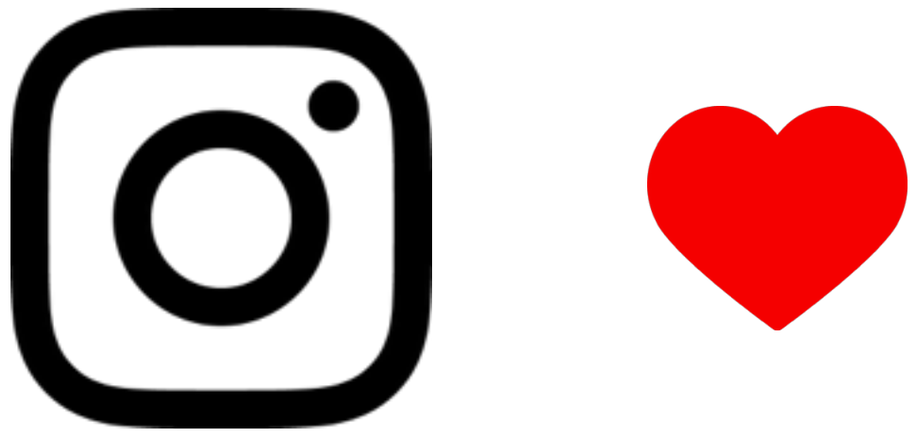
— Mike Krieger

"THE PIECES WERE PLUGGABLE ENOUGH... EVEN WITH OUR OWN ORM WE COULD USE MOST OF THE REST OF DJANGO."

— Mike Krieger

AN INCOMPLETE LIST OF THE DJANGO WE RELY ON

- HTTP stack
- requests and responses
- contrib.sessions
- contrib.auth
- middleware
- url routing
- settings
- forms
- i18n
- contrib.gis
- django.utils
- cache backends
- HTTP decorators
- CSRF
- signals
- management commands




python 3
async(io)
traffic replay
pypy?
CPython JIT?








engineering.instagram.com

carljm@instagram.com

@carljm


T.Hanks

PHOTOS

database by RockIcon, smiley by Vandana Agrawal, server by Alexander Skowalsky, from Noun Project

https://www.flickr.com/photos/yashh/2834704689

https://unsplash.com/photos/KEXUeZlev10

https://unsplash.com/photos/pd4lo70LdbI

https://unsplash.com/photos/jh2KTqHLMjE

https://www.flickr.com/photos/johnsonderman/15144843722

https://www.flickr.com/photos/kennethreitz/5521545772/

https://www.instagram.com/p/mNj4L3OTzj/

https://www.flickr.com/photos/67926342@N08/6175870684

https://www.flickr.com/photos/lytfyre/6489338411

https://unsplash.com/photos/4fQAMZNaGUo

https://unsplash.com/photos/glHJybGNt1M

https://www.flickr.com/photos/nedrichards/51132692

https://www.flickr.com/photos/sophistechate/2913053678

https://www.flickr.com/photos/elviskennedy/6784123582

https://unsplash.com/photos/HkTMcmlMOUQ