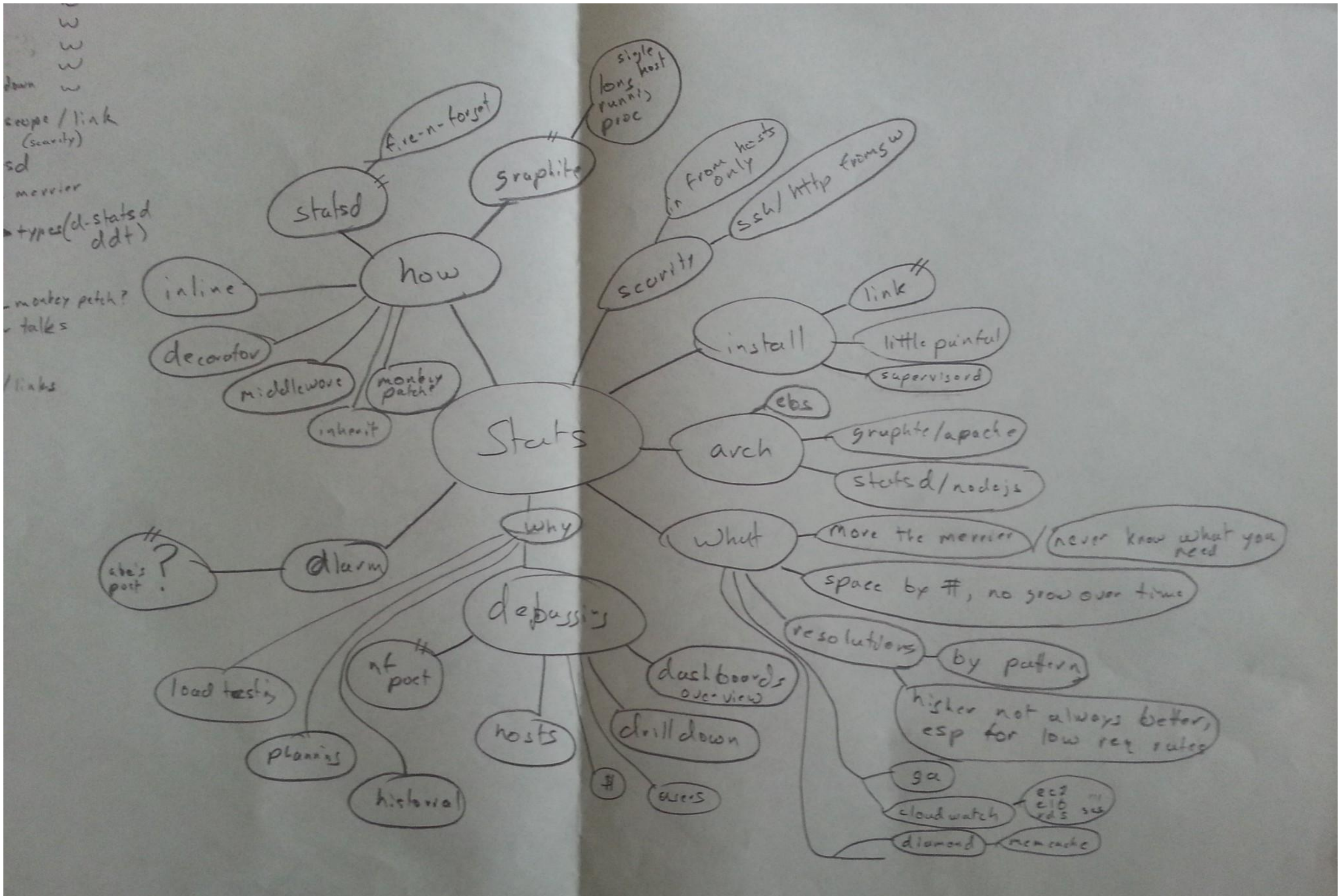
Stats: Keeping Track

with Graphite and Statsd

Outline



Stats
how
statsd
fire-n-forget
graphite
long running proc
single host
inline
decorator
middleware
monkey patch
inherit
security
from hosts only
ssh/http from sw
install
link
little painful
supervisord
arch
ebs
graphite/apache
statsd/nodejs
what
more the merrier
never know what you need
space by #, no grow over time
resolutions
by pattern
higher not always better, esp for low req rates
ga
cloudwatch
diamond
memcache
why
alarm
debugging
load testing
planning
historical
hosts
drill down
dashboards overview
users
#

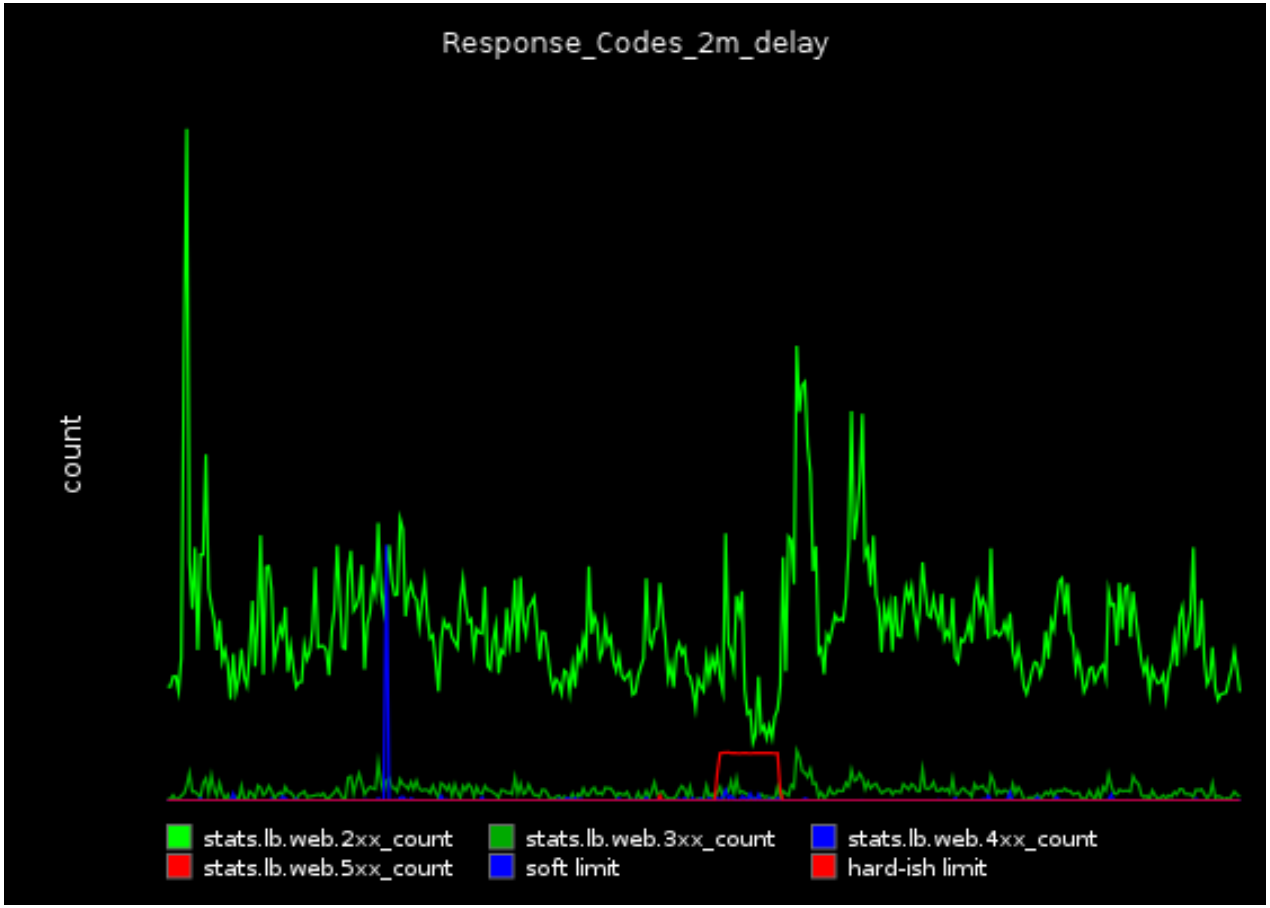
Why Stats

On-call gets "paged" and ALERT email sent by a tripped 500 count from the API

App & site are fine

API calls working when manually checked

Bring up the overview dashboard...



Django 1.4.4's ALLOWED_HOSTS and ELB not getting along.

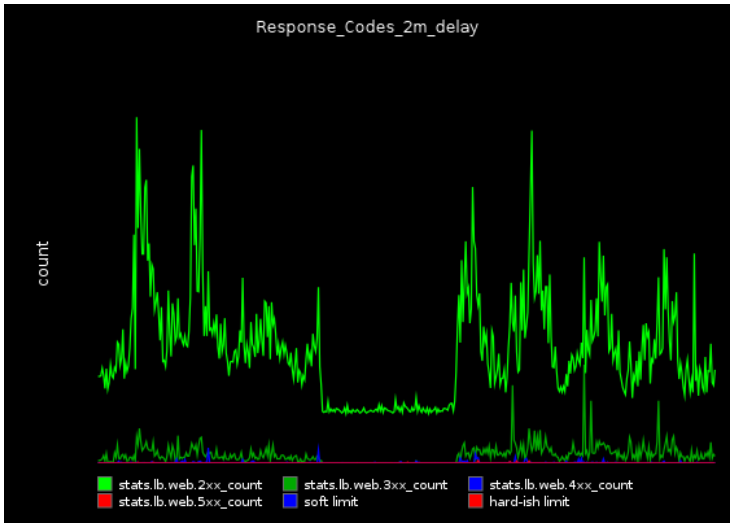
Why Stats

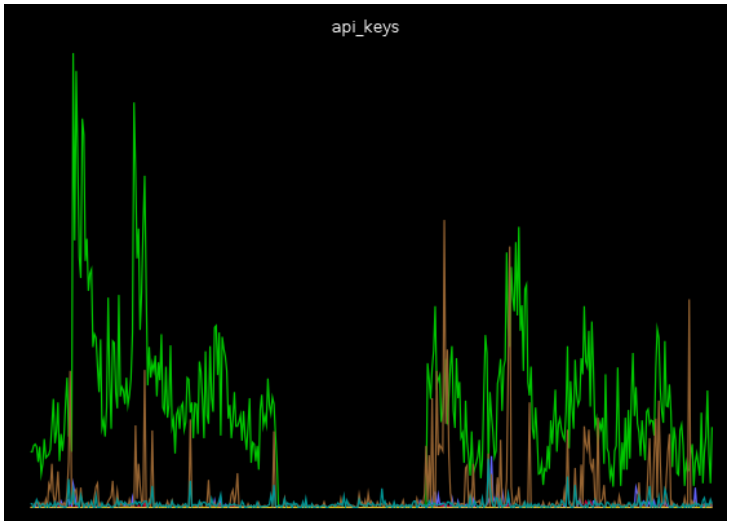
Make an SSL config change

Check everything out via a dev build of the app and it's working fine

Watch response counts lose variability

API key counts drop to almost zero





Dev app uses SSL, but doesn't verify cert

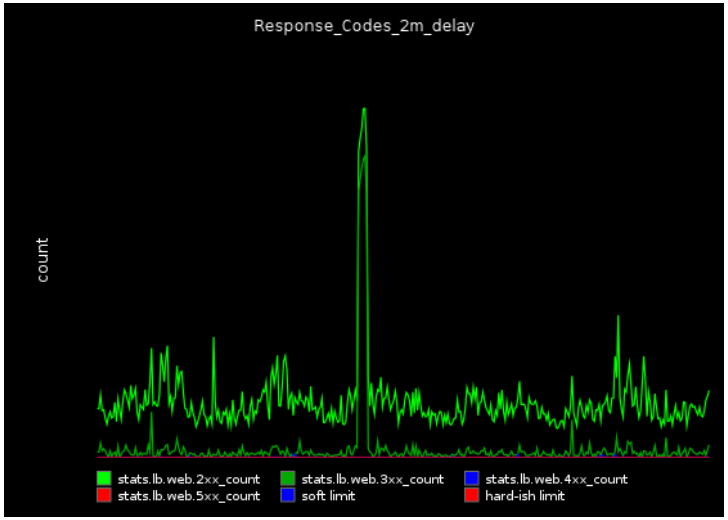
What remains is using HTTP, not HTTPS

Why Stats

On-call gets "paged"

Abnormal count of 200's and 300's

Large spike in a specific API key

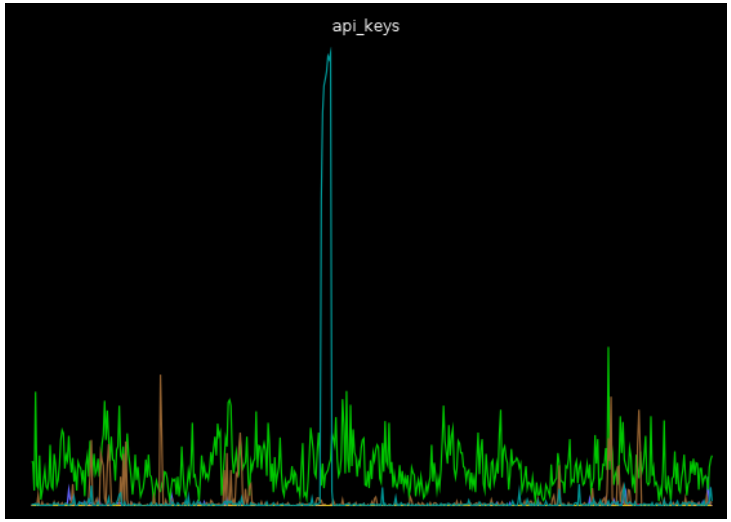


API call accidentally in a loop causing that client to make many calls

We immediately know where to look or at least can rule out a lot of things

Other graphs indicate specific API call counts

* Luckily not at a level that hurt anything, thank the caching gods



Graphite vs Statsd

- Graphite
  - A collection of software for collecting, storing, reporting, and graphing time series data
  - It's a complete car, nothing too fancy, but it'll get you where you're going
- Statsd
  - "Lightweight" server for collecting stat events and passing them along to a backend (Graphite)
  - Can take "raw" event counts and aggregate them across time and submit the results
  - It's like adding a high performance engine and transmission

Installation & Configuration

Unfortunately too involved for a presentation

- http://geek.michaelgrace.org/2011/09/how-to-install-graphite-on-ubuntu/
- where I started, plenty of statsd howto's as well

There is a company offering a SaaS graphite setup

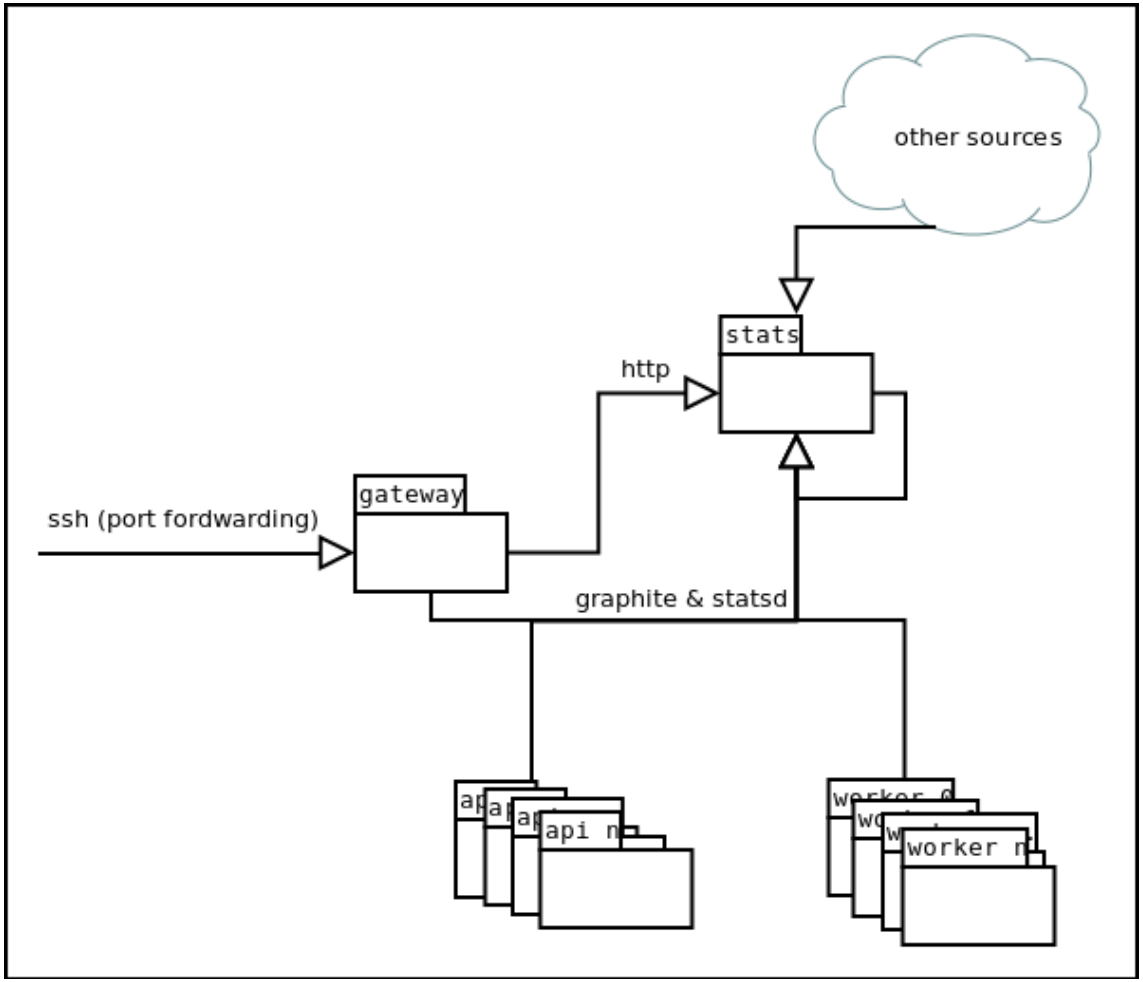
- https://www.hostedgraphite.com/
- haven't used it, might be a quick way to test the waters, ymmv

There are many options

- NewRelic
- AWS Cloudwatch (pretty limited)
- TraceView - http://appneta.com/traceview
- Munin - http://munin-monitoring.org/
- Ganglia - http://ganglia.sourceforge.net/
- ...

Our Architecture

- all web access via ssh tunnel
- data sent to elastic ip of stats box
- Diamond sends system-level stats
- django-statsd & statsd client send app-level event and duration data
- Custom scripts source and send data from CloudWatch, Google Analytics, ...
- Overview/health dashboards for looking in on things.
- Lots of purpose-built dashboards for digging deeper
- ad-hoc graphs as needed



What We Track

- Any and everything, the more the merrier
- Pretty high resolution, 10s, I wouldn't go much higher or lower
- Databases are round-robin, so pretty cheap
- If _____, we track it
  - we're curious how much it'll happen
  - it might indicate system health
  - can impact end user experience
  - we expect it to change over time
  - someone might get pissed if it goes awry
  - it might have even a slight chance of being useful
- When in doubt, track it

What Stats Are Not

- A replacement for logs
- We collect logs to a central host via rsyslogd
- Stats can help tell you what to look for or rule things out, but they seldom give you concrete answers

How We Track

- Scripts
- django-statsd middleware
- Counting Things
- Timing Things
- Inheritance

Scripts

- Run with while True and sleeps
- Important to subtract run time from how long you sleep before next or else it'll skew
- Uses some diamond code as helpers
  - DiamondMetric
  - GraphiteHandler
- I can show you one, but it's not interesting enough to put here

django-statsd middleware

- there are 2 versions out there, best one seems to be django-statsd-mozilla
- a lot for "free"
  - response status codes
  - view stats
    - timers
    - counts
    - individual and aggregate
  - database stats

Counting Things

- For anything else you want to count
- django_statsd provides a configured statsd client

```
from django_statsd.clients import statsd

if user.is_cool:
    # count the number of cool users we see
    statsd.incr('users.cool')
else:
    statsd.incr('users.regular')
```

- A real example might be counting the types of email queued

Timing Things

With a Decorator

```
@statsd.timer('process.something')
def process_something():
    # this takes a while ...
    response = requests.get('http://www.httpbin.com/get')
```

With a Context

```
def process_something():
    with statsd.timer('requests.httpbin'):
        response = requests.get('http://www.httpbin.com/get')
```

Inheritance

Wrap a class you'll be using to add tracking

```
class StatsSession(requests.Session):

    def request(self, method, url, *args, **kwargs):
        host = urlparse(url).netloc.replace('.', '_')
        key = 'requests.%s.%s' % (host, method)
        statsd.incr(key)
        with statsd.timer(key):
            return super(StatsSession, self).request(method, url, *args,
**kwargs)
```

Sometimes it requires monkeypatching :(

Demo

repo - https://github.com/ross/statsd-demo

Links

- Graphite - http://graphite.wikidot.com/
- statsd - https://github.com/etsy/statsd/
- django-statsd - https://django-statsd.readthedocs.org/en/latest/
- python statsd client - https://statsd.readthedocs.org/en/latest/
- Diamond - https://github.com/BrightcoveOS/Diamond
- howto - https://gist.github.com/jgeurts/3112065

Ross McFarland

http://www.xormedia.com/
http://github.com/ross/
rwmcfa1@neces.com