Continuous Delivery with Docker

Tobias Schwab

Myself

- Tobias Schwab
- tobias.schwab@dynport.de
- www.dynport.de
- twitter.com/tobstarr
- github.com/tobstarr

Philosophie

- continuous delivery: deploy multiple times a day
- canary releases
- “never touch a running system”
- “Immutable Infrastructure and Disposable Components"
- don’t fix it, if it can be replaced

Theory

- AWS
- AMI based deployments
- Elastic Load Balancer
- AutoScaling Groups
- S3, RDS, …

Reality

- privacy concerns: AWS not an option
- hoster we could not pick
- first no, then proprietary and unreliable API
- flash based infrastructure management
- limited capacity
- we were the biggest customer

Build

- manual
  - start and attach container
  - install required packages
  - checkout application code
  - run build management tool
  - bad: not reproducible
  - bad: does not utilise caching

Build

- chef/puppet/…
    - start an attach container
    - run chef/puppet/… client
    - good: automated and documented
    - bad: does not utilise caching

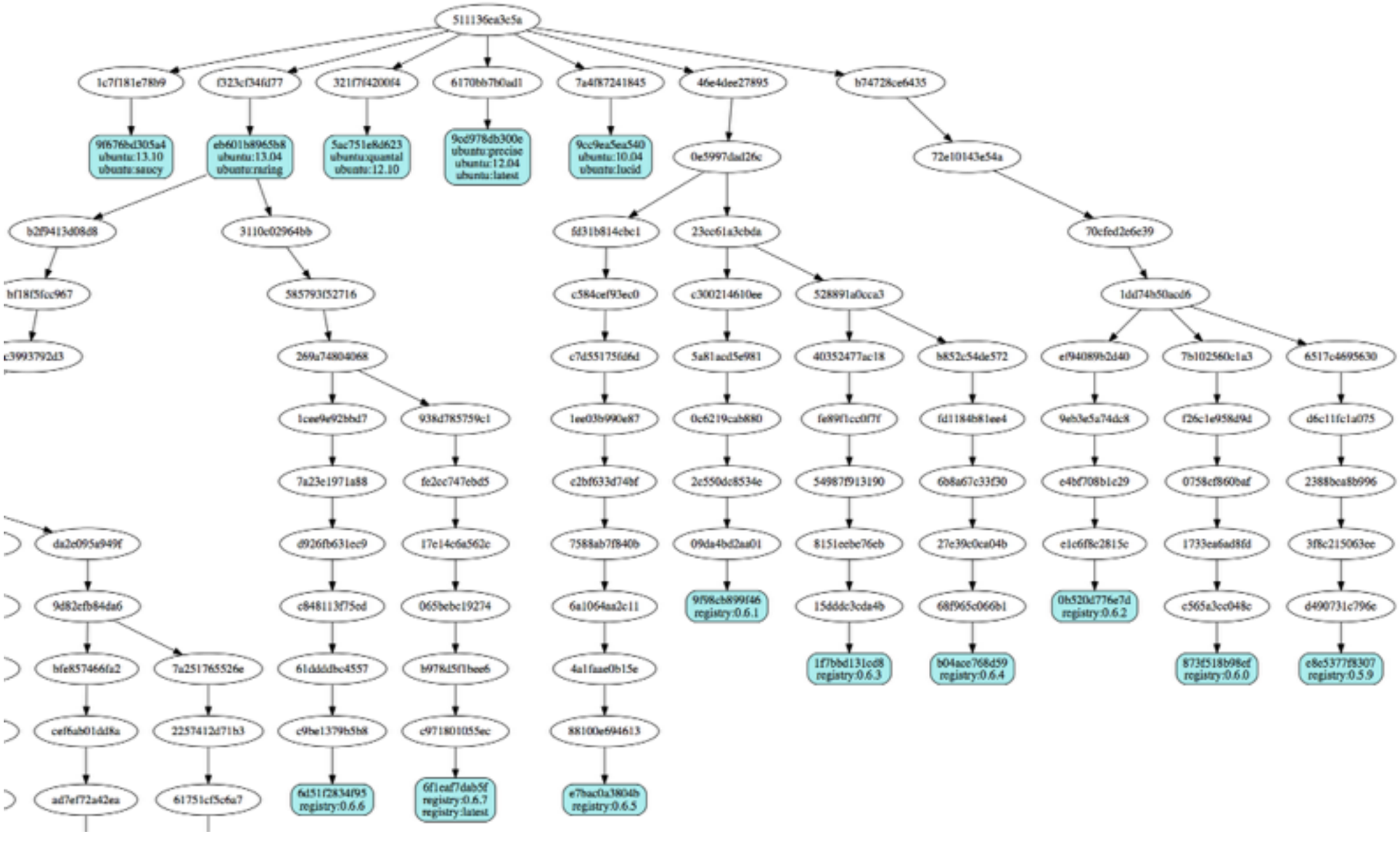
Dockerfile

- simple, plain text script to create images
- commands:
    - FROM: base image to use
    - RUN: execute shell command
    - ENV: set environment variable
    - ADD: write local file to image
    - ENTRYPOINT: start command for containers
- others: MAINTAINER, EXPOSE, CMD, USER, VOLUME, WORKDIR, ONBUILD

Caching

- statement based: each step creates a new image
- existing steps (command tree exists) are re-used
- tricky: “non functional” commands (e.g. apt-get update/upgrade)
- use ENV or comments to break caching of non functional commands

Caching


511136ea3c5a
1c7f181e78b9
f323cf34fd77
321f7f4200f4
6170bb7f0ad1
7a4f87241845
46e4dee27895
b74728ce6435
9f676bd305a4
ubuntu:13.10
ubuntu:saucy
eb601b8965b8
ubuntu:13.04
ubuntu:raring
5ac751e8d623
ubuntu:quantal
ubuntu:12.10
9cd978db300e
ubuntu:precise
ubuntu:12.04
ubuntu:latest
9cc9ea5ea540
ubuntu:10.04
ubuntu:lucid
0e5997dad2fc
72e10143e54a
b2f9413d08d8
3110c02964bb
fd31b814cbc1
23cc61a3cbda
70cfed2e6c39
bf18f5fcc967
585793f52716
c584cef93ec0
c300214610ee
528891a0cca3
1dd74b50acd6
c3993792d3
269a74804068
c7d55175fd6d
5a81acd5e981
40352477ac18
b852c54de572
ef94089b2d40
7b102560c1a3
6517c4695630
1cee9e92bbd7
938d785759c1
1ee03b990e87
0c6219cab880
fe89f1cc0f7f
fd1184b81ee4
9eb3e5a74dc8
f26c1e958d9d
d6c11fc1a075
7a23e1971a88
fe2cc747ebd5
e2bf633d74bf
2e550dc8534e
54987f913190
6b8a67c33f30
e4bf708b1e29
0758cf860baf
2388bca8b996
da2e095a949f
d926fb631ec9
17e14c6a562c
7588ab7f840b
09da4bd2aa01
8151eebe76eb
27e39c0ca04b
e1c6f8c2815c
1733ea6ad8fd
3f8c215063ee
9d82cfb84da6
c848113f75cd
065bebc19274
6a1064aa2c11
9f98cb899f46
registry:0.6.1
15dddc3cda4b
68f965c066b1
0b520d776e7d
registry:0.6.2
c565a3cc048c
d490731c796c
bfe857466fa2
61ddddbc4557
b978d5f1beee6
4a1faae0b15e
1f7bbd131cd8
registry:0.6.3
b04ace768d59
registry:0.6.4
873f518b98ef
registry:0.6.0
e8e5377f8307
registry:0.5.9
cef6ab01dd8a
c9be1379b5b8
c971801055ec
88100e694613
7a25176526e
2257412d71b3
6d151f2834f95
registry:0.6.6
6f1caaf7dab5f
registry:0.6.7
registry:latest
e7bac0a3804b
registry:0.6.5
ad7ef72a42ea
61751cf5c6a7

Configuration Management

- “store config in the environment” (http://12factor.net/config)
- dependency injected with start of container
- same image for
  - development
  - testing
  - staging
  - production

Don’ts

- full blown VMs
- ssh daemon inside containers
- syslog daemon inside containers (sometimes needed)
- user management: everything can run as root
- chef/puppet/… => makes caching useless

Build Management Tools

- candidates: bundler, pip, mvn, carton, composer, …
- problem with caching: bmt are slow when started with “clean slate”
- option 1: add bmt manifest before code
    - bmt needs to run only when manifest changes
- option 2: use pre-bundled base images
    - bmt only needs to work the delta
    - re-build base images from time to time
- option 3: combine option 1 and option 2

Use Case: Ruby on Rails

- Problems
  - unicorn: Rack HTTP server for fast clients
  - static assets
  - logging: default ruby syslog library uses syscall (needs local syslog daemon)
- Solution
  - run 3 daemons in 1 container: unicorn, nginx and rsyslogd
  - upstart
    - ENTRYPOINT [“/sbin/init”]
    - load ENV from /proc/1/environ
  - foreman

Multi-Host

- image distribution via docker registry
- weighted load balancing via HAProxy
- SSL termination via nginx in front of HAProxy

Registry

- push and pull images
- public
- private
- backends: local, S3, Elliptics, Google Cloud Storage, hosted

Load Balancing

- HAProxy
- pool configuration stored in redis/etcd
- config update
  - compile config files from stored configuration
  - upload via ssh
  - verify on remote hosts
  - replace current config with verified one
  - reload

HAProxy

```
global
  daemon
  pidfile /var/run/haproxy.pid
  stats socket /tmp/haproxy.socket level admin
  log 192.168.0.4 user info

  defaults
    mode http
    maxconn 50000
    timeout connect 5000ms
    timeout client 50000ms
    timeout server 50000ms
    option http-server-close
    option http-pretend-keepalive
    option httplog
    log global

frontend in-stats
  bind *:20002
  default_backend stats

backend stats
  stats uri /_stats
  stats refresh 5

frontend in-rails
  bind 192.168.0.6:8080
  default_backend rails

backend rails
  rspadd X-Proxy:\ cnc-5f22a980
  option httpchk GET /_status

  # server cnc-369bf803:f753afc24d79:192c3f992501
  acl cnc-369bf803:f753afc24d79:192c3f992501 srv_id 1
  rspadd X-Host:\ cnc-369bf803:f753afc24d79:192c3f992501 if cnc-369bf803:f753afc24d79:192c3f992501
  server cnc-369bf803:f753afc24d79:192c3f992501 192.168.0.19:49382 check inter 5s id 1 weight 100

  # server cnc-369bf803:f753afc24d79:5dd19b1da07a
  acl cnc-369bf803:f753afc24d79:5dd19b1da07a srv_id 2
  rspadd X-Host:\ cnc-369bf803:f753afc24d79:5dd19b1da07a if cnc-369bf803:f753afc24d79:5dd19b1da07a
  server cnc-369bf803:f753afc24d79:5dd19b1da07a 192.168.0.19:49383 check inter 5s id 2 weight 100
```

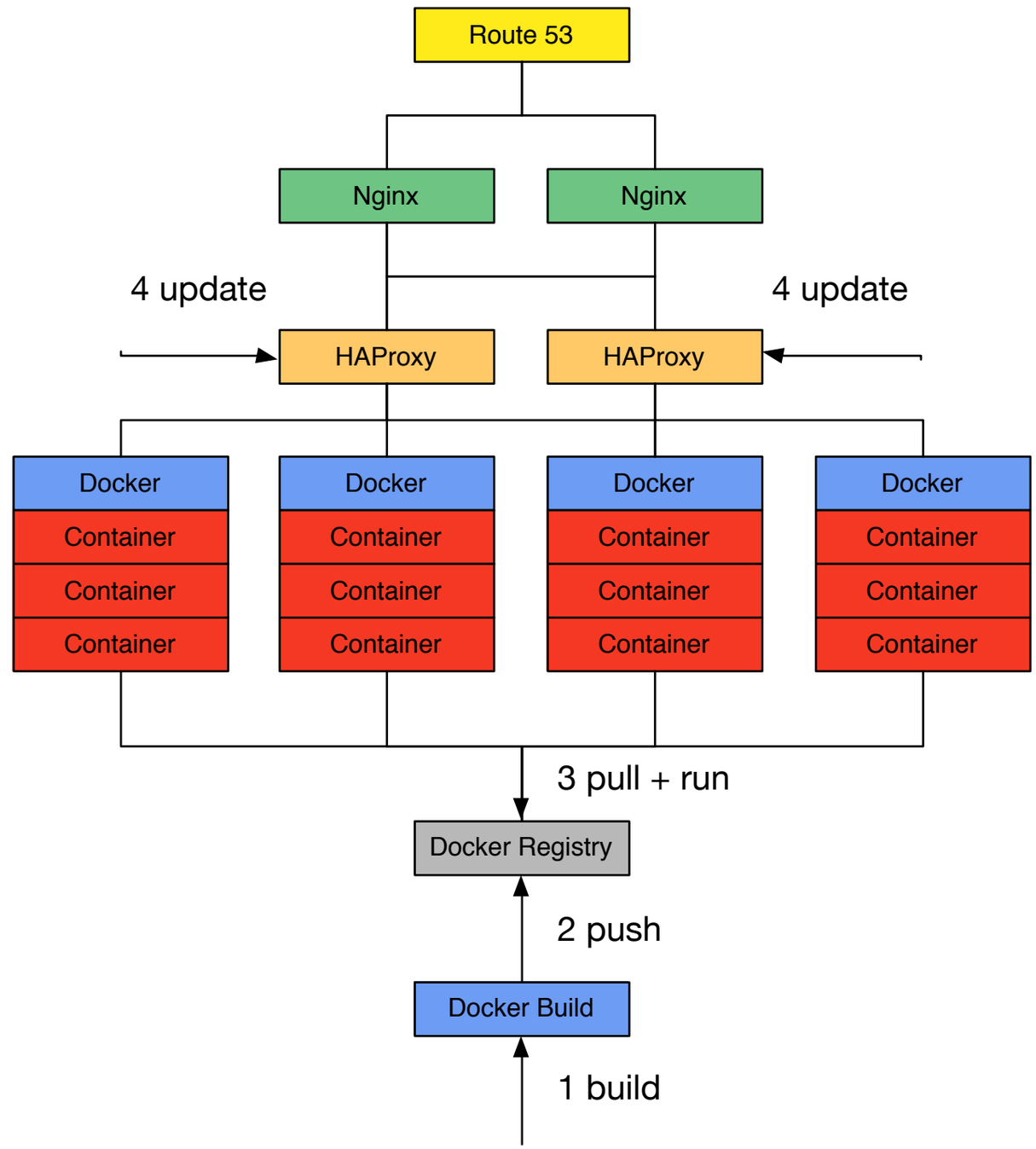
HAProxy

	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings		
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status
cnc-535934d7:b2b46b4425a7:4074cf7ba135	0	0	-	2	6		0	29	-	1 215	1 215	38 759 651	26 592 153		0		0	0	0	0	3m28s UP
cnc-82f8a2c8:b2b46b4425a7:b2674302a38c	0	0	-	2	6		1	50	-	1 170	1 170	28 607 397	29 647 058		0		0	0	0	0	1m10s UP
cnc-87c73527:b2b46b4425a7:e249016faa9a	0	0	-	2	8		0	8	-	1 283	1 283	9 625 072	24 274 168		0		0	0	0	0	12m50s UP
cnc-c1b3dd12:b2b46b4425a7:26e76d513ce4	0	0	-	2	8		8	31	-	1 257	1 257	17 259 361	33 735 903		0		0	0	0	0	2m15s UP
Backend	0	0		9	22		9	59	0	4 925	4 925	94 251 481	114 249 282	0	0		0	0	0	0	12m50s UP

Deployment Pipeline

- commit triggers new image build
- build suite executed with image
- image is pushed to registry if tests passed
- optional: start image with staging ENV settings for manual testing
- start image with production ENV for last pre-flight tests
- deploy image to more hosts
- update load balancer (canary or green/blue)
- monitor new containers/image

Deployment Pipeline



Logging

- NOT inside containers
- remote syslog (when possible)
- alternative: local syslog relay inside container

```
2013-11-20T16:00:00.003829+00:00 1d63a4fc82a1 nginx: [redacted] method=GET s
tatus=200 length=38173 pid=27771 rev=5f3cf70cc90c image_id=b8d7736b9986 uuid=da7d5065-a544-4c6a-88dd-7c0dbc57fdb6 action=hom
e#index etag=2dffe9baf37402334042dde91c055db5 rack=0.140591 db=0.035836/12 db_cache=0.002128/3 total=0.146 upstream_time=0.1
43 ua="Mozilla/5.0 (Linux; Android 4.2.2; de-de; SAMSUNG GT-I9195 Build/JDQ39) AppleWebKit/535.19 (KHTML, like Gecko) Versio
n/1.0 Chrome/18.0.1025.308 Mobile Safari/535.19" uri="/" ref="-"
```

- host: docker host, container_id
- code: image_id, revision
- request: request_id, action, status_code, etag, times, calls

Metrics

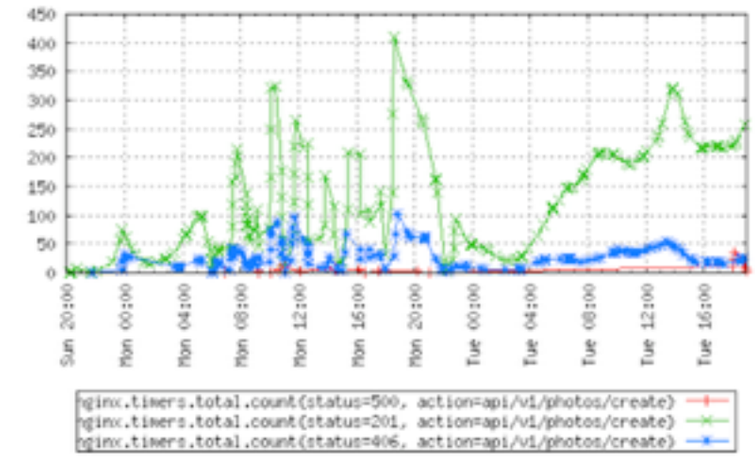
- OpenTSDB
  - “distributed, scalable Time Series Database”
  - HBase
    - Tags / Dimensions
- from syslog via udp (StatsD “like”)
- rickshaw.js for graphs
- compare status codes, counts and times between actions of two revisions

OpenTSDB

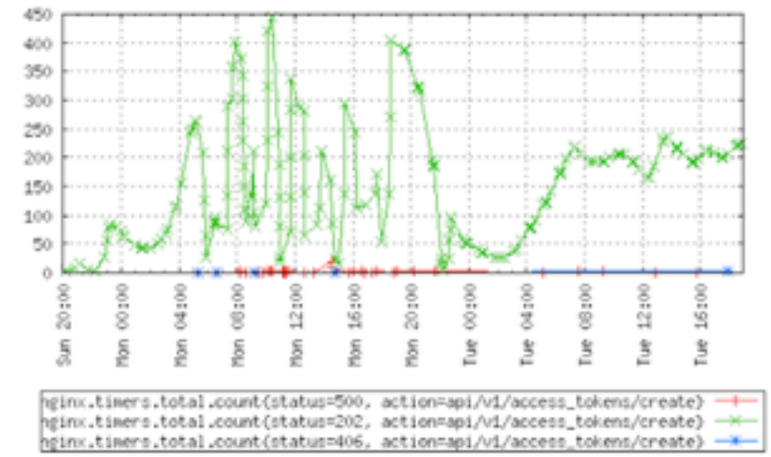
```
nginx.timers.total.count 1389589200 1 revision=bb103bbf37f3 image_id=a0c79553f35f
nginx.timers.total.count 1389589201 2 revision=bb103bbf37f3 image_id=a0c79553f35f
nginx.timers.total.count 1389589202 3 revision=bb103bbf37f3 image_id=a0c79553f35f
nginx.timers.total.count 1389589204 4 revision=bb103bbf37f3 image_id=a0c79553f35f
nginx.timers.total.count 1389589208 6 revision=bb103bbf37f3 image_id=a0c79553f35f
nginx.timers.total.count 1389589209 7 revision=bb103bbf37f3 image_id=a0c79553f35f
nginx.timers.total.count 1389589210 9 revision=bb103bbf37f3 image_id=a0c79553f35f
nginx.timers.total.count 1389589212 11 revision=bb103bbf37f3 image_id=a0c79553f35f
nginx.timers.total.count 1389589213 14 revision=bb103bbf37f3 image_id=a0c79553f35f
nginx.timers.total.count 1389589215 15 revision=bb103bbf37f3 image_id=a0c79553f35f
nginx.timers.total.count 1389589216 18 revision=bb103bbf37f3 image_id=a0c79553f35f
nginx.timers.total.count 1389589217 20 revision=bb103bbf37f3 image_id=a0c79553f35f
nginx.timers.total.count 1389589218 21 revision=bb103bbf37f3 image_id=a0c79553f35f
nginx.timers.total.count 1389589221 24 revision=bb103bbf37f3 image_id=a0c79553f35f
nginx.timers.total.count 1389589224 27 revision=bb103bbf37f3 image_id=a0c79553f35f
nginx.timers.total.count 1389589225 28 revision=bb103bbf37f3 image_id=a0c79553f35f
nginx.timers.total.count 1389589226 32 revision=bb103bbf37f3 image_id=a0c79553f35f
nginx.timers.total.count 1389589227 33 revision=bb103bbf37f3 image_id=a0c79553f35f
nginx.timers.total.count 1389589228 35 revision=bb103bbf37f3 image_id=a0c79553f35f
nginx.timers.total.count 1389589229 40 revision=bb103bbf37f3 image_id=a0c79553f35f
nginx.timers.total.count 1389589230 64 revision=bb103bbf37f3 image_id=a0c79553f35f
nginx.timers.total.count 1389589231 129 revision=bb103bbf37f3 image_id=a0c79553f35f
nginx.timers.total.count 1389589232 149 revision=bb103bbf37f3 image_id=a0c79553f35f
nginx.timers.total.count 1389589233 157 revision=bb103bbf37f3 image_id=a0c79553f35f
nginx.timers.total.count 1389589234 159 revision=bb103bbf37f3 image_id=a0c79553f35f
```

Metrics

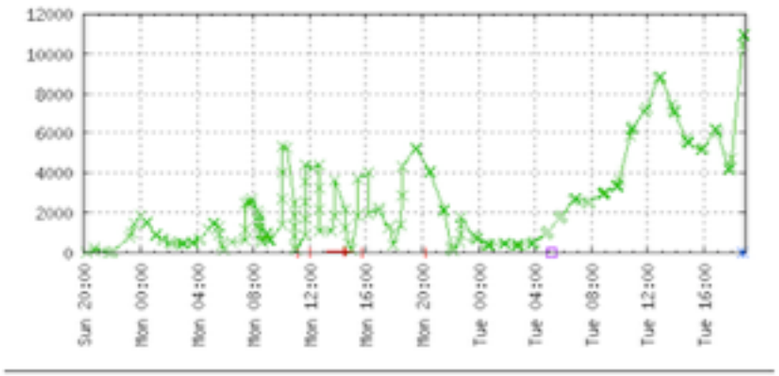
api/v1/photos/create



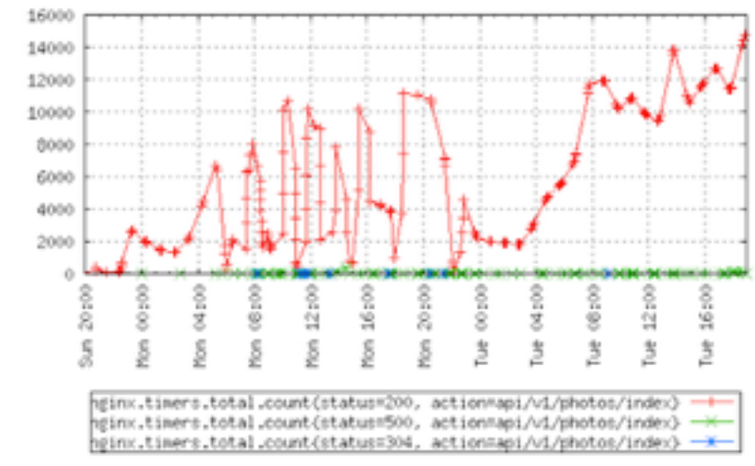
api/v1/access_tokens/create



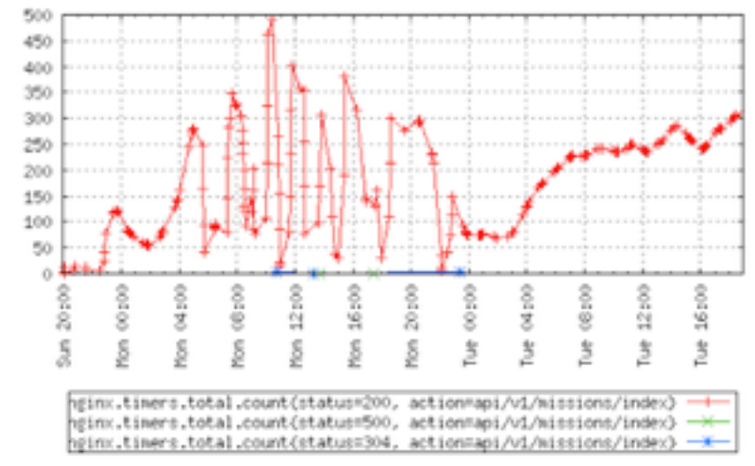
api/v1/ratings/update



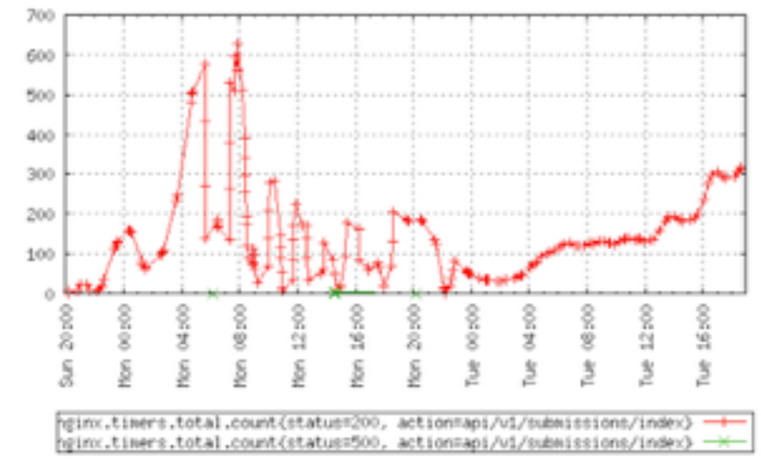
api/v1/photos/index



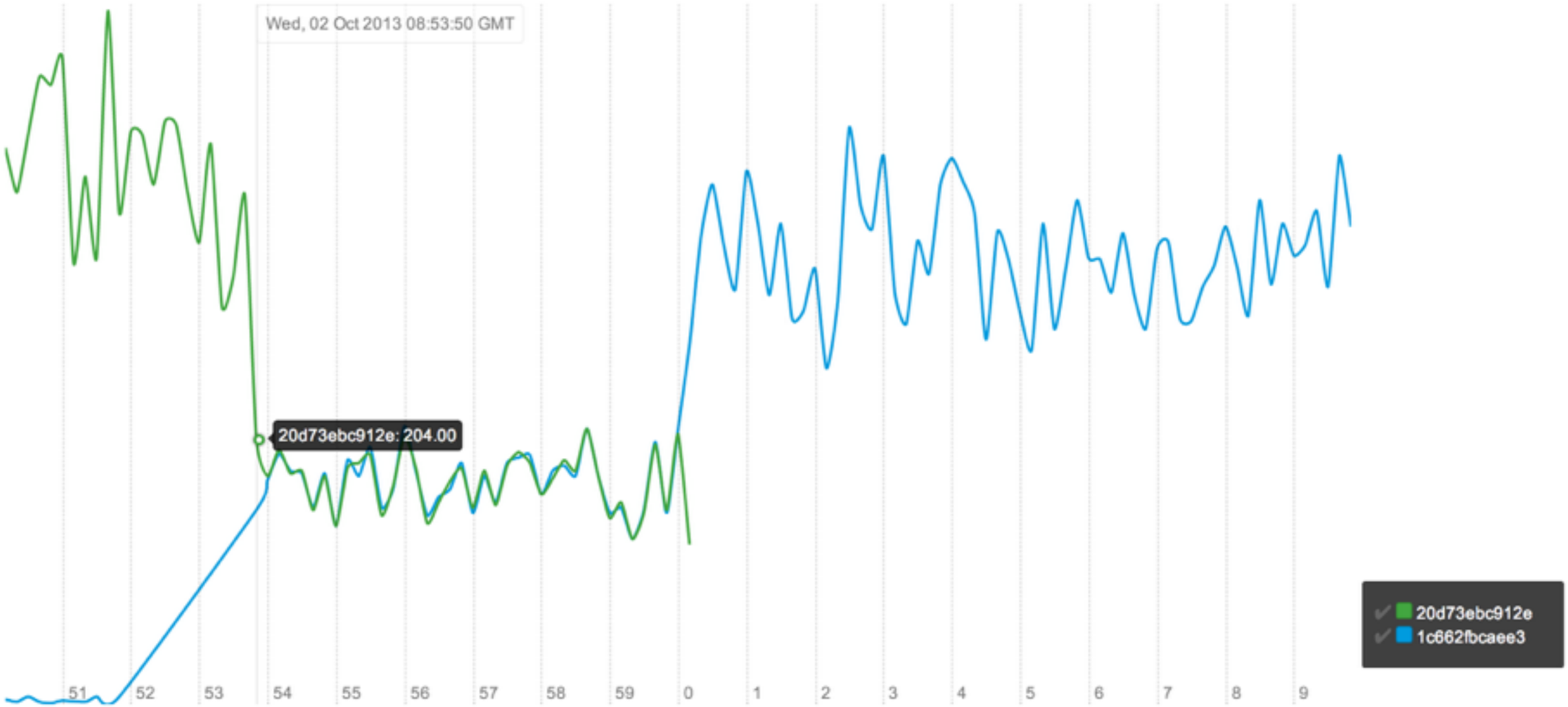
api/v1/missions/index



api/v1/submissions/index

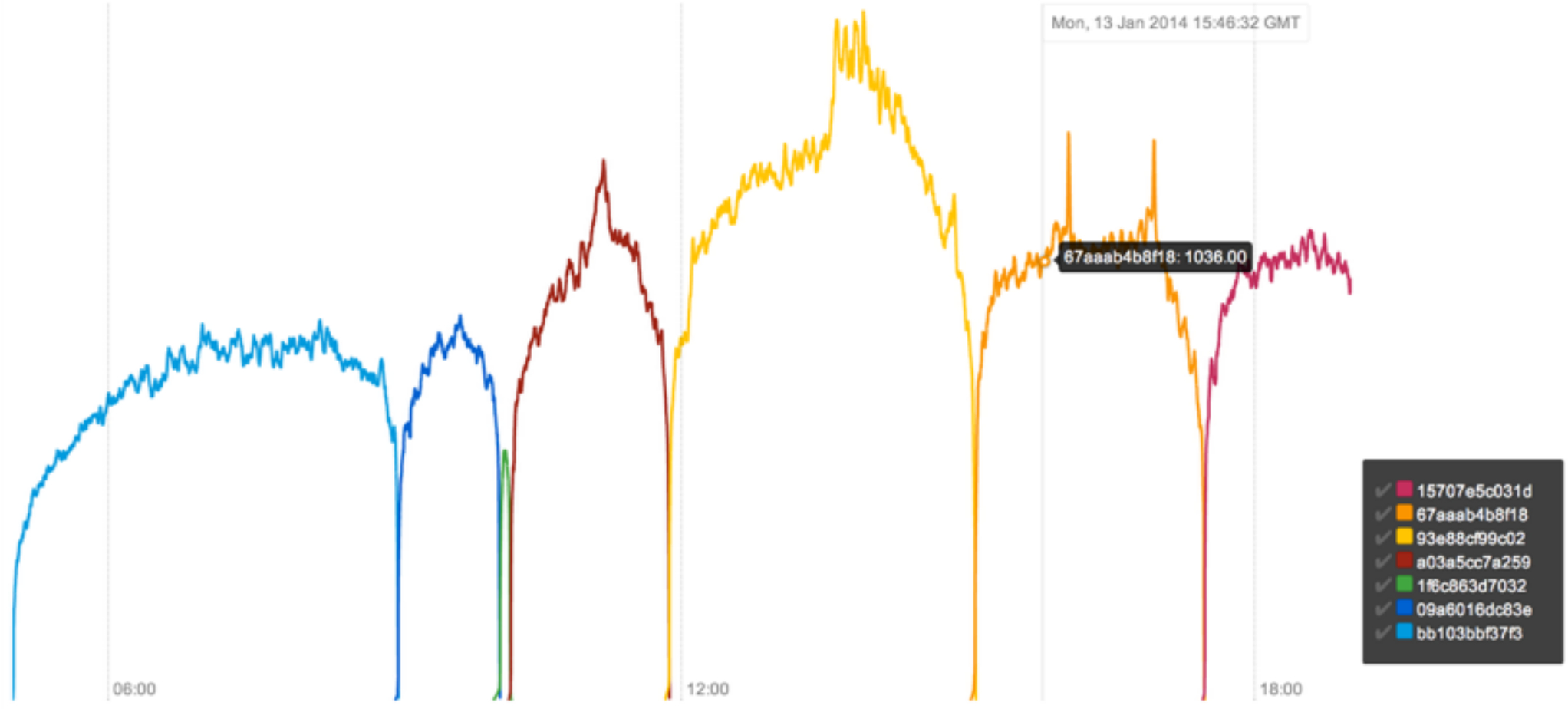


Metrics



request counts by revision

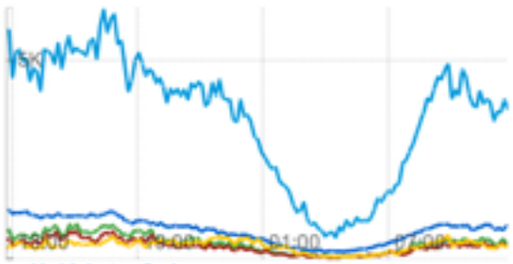
Metrics


Mon, 13 Jan 2014 15:46:32 GMT
67aaab4b8f18: 1036.00
06:00
12:00
18:00
15707e5c031d
67aaab4b8f18
93e88cf99c02
a03a5cc7a259
1f6c863d7032
09a6016dc83e
bb103bbf37f3

Metrics

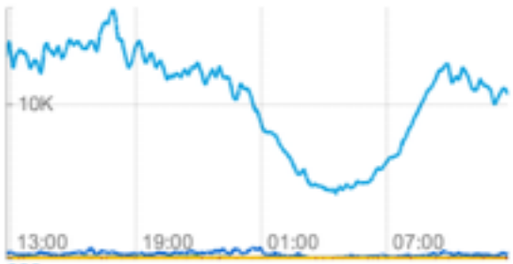
Requests

Action



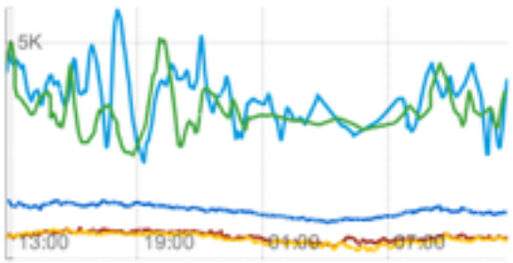
api/v1/photos/index
photos/iframe
api/v1/my/access_tokens/show
photos/show
api/v1/photos/submitted

Status Total



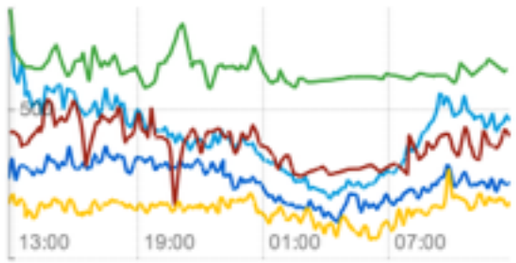
200
201
202
302
401

Total 95%



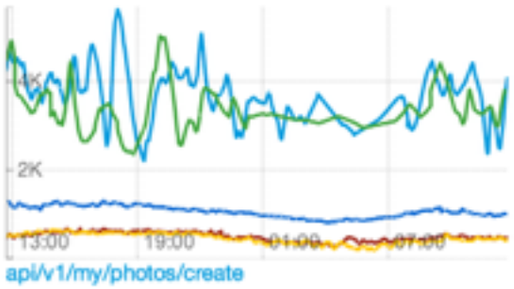
api/v1/my/photos/create
api/v1/photos/index
api/v1/photos/create
api/v1/photos/submitted
api/v1/notifications/index

DB 95%



api/v1/photos/index
api/v1/photos/submitted
api/v1/photos/rate
api/v1/photos/leserreporter
api/v1/notifications/index

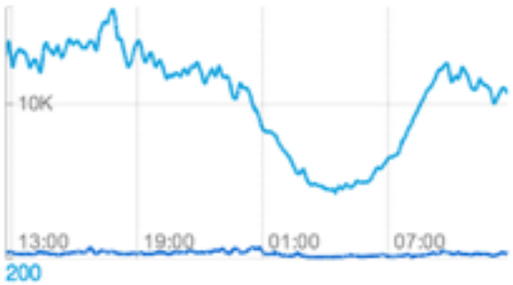
Rack 95%



api/v1/my/photos/create
api/v1/photos/index
api/v1/photos/create
api/v1/photos/submitted
api/v1/notifications/index

3xx

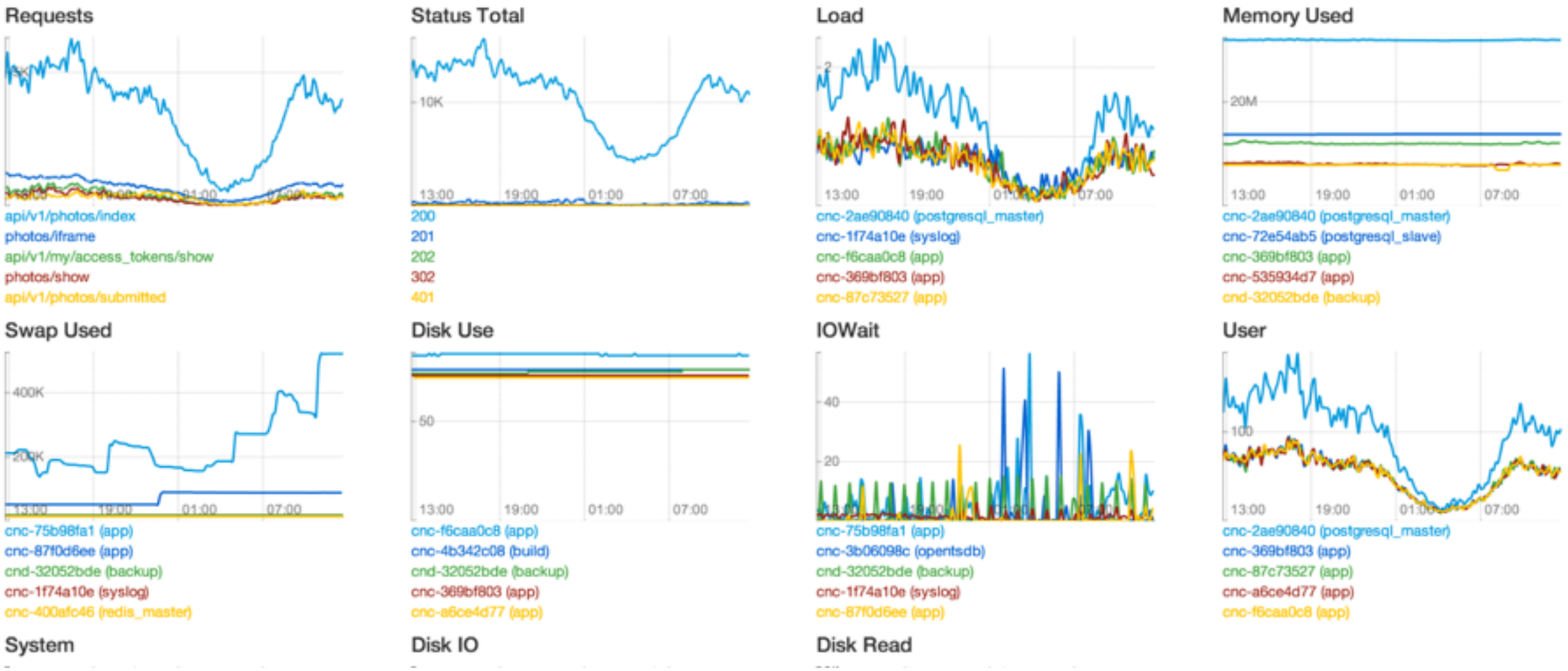
2xx



200
201

5xx

Metrics


Requests
13:00
19:00
01:00
07:00
api/v1/photos/index
photos/iframe
api/v1/my/access_tokens/show
photos/show
api/v1/photos/submitted
Status Total
10K
13:00
19:00
01:00
07:00
200
201
202
302
401
Load
2
13:00
19:00
01:00
07:00
cnc-2ae90840 (postgresql_master)
cnc-1f74a10e (syslog)
cnc-f6caa0c8 (app)
cnc-369bf803 (app)
cnc-87c73527 (app)
Memory Used
20M
13:00
19:00
01:00
07:00
cnc-2ae90840 (postgresql_master)
cnc-72e54ab5 (postgresql_slave)
cnc-369bf803 (app)
cnc-535934d7 (app)
cnd-32052bde (backup)
Swap Used
400K
200K
13:00
19:00
01:00
07:00
cnc-75b98fa1 (app)
cnc-87f0d6ee (app)
cnd-32052bde (backup)
cnc-1f74a10e (syslog)
cnc-400afc46 (redis_master)
Disk Use
50
13:00
19:00
01:00
07:00
cnc-f6caa0c8 (app)
cnc-4b342c08 (build)
cnd-32052bde (backup)
cnc-369bf803 (app)
cnc-a6ce4d77 (app)
IOWait
40
20
cnc-75b98fa1 (app)
cnc-3b06098c (opentsdb)
cnd-32052bde (backup)
cnc-1f74a10e (syslog)
cnc-87f0d6ee (app)
User
100
13:00
19:00
01:00
07:00
cnc-2ae90840 (postgresql_master)
cnc-369bf803 (app)
cnc-87c73527 (app)
cnc-a6ce4d77 (app)
cnc-f6caa0c8 (app)
System
Disk IO
Disk Read

Docker reduces

- external dependencies (“rubygems/github slow/unreliable/down”) after image is built
- “did work on my machine/staging”: same OS package versions, configuration and code in all stages
- unused CPU cycles
- number of hosts
- feedback times
- time to get new host online
- bottlenecks: hosts are more flexible

VS. AWS

- HAProxy much more flexible
  - multiple containers per host
    - balancing weights
- faster build process
- faster deployments
- instance flexibility

Resources

- docker.io
- opentsdb.net
- haproxy.1wt.eu
- continuousdelivery.com
- chadfowler.com/blog/2013/06/23/immutable-deployments/
- 12factor.net

Questions?!?

Thank you!