How TubeMogul Handles over One Trillion HTTP Requests a Month

November 12th, 2015

Nicolas Brousse | Sr. Director Of Operations Engineering | nicolas@tubemogul.com

usenix LISA15
November 8–13, 2015 | Washington, D.C.
www.usenix.org/lisa15 #lisa15

Who are we?

TubeMogul

- Enterprise software company for digital branding
- Over **27 Billions** Ads served in 2014
- Over **30 Billions** Ad Auctions per day
- Bid processed in less than **50 ms**
- Bid served in less than **80 ms** (include network round trip)
- **5 PB** of monthly video traffic served

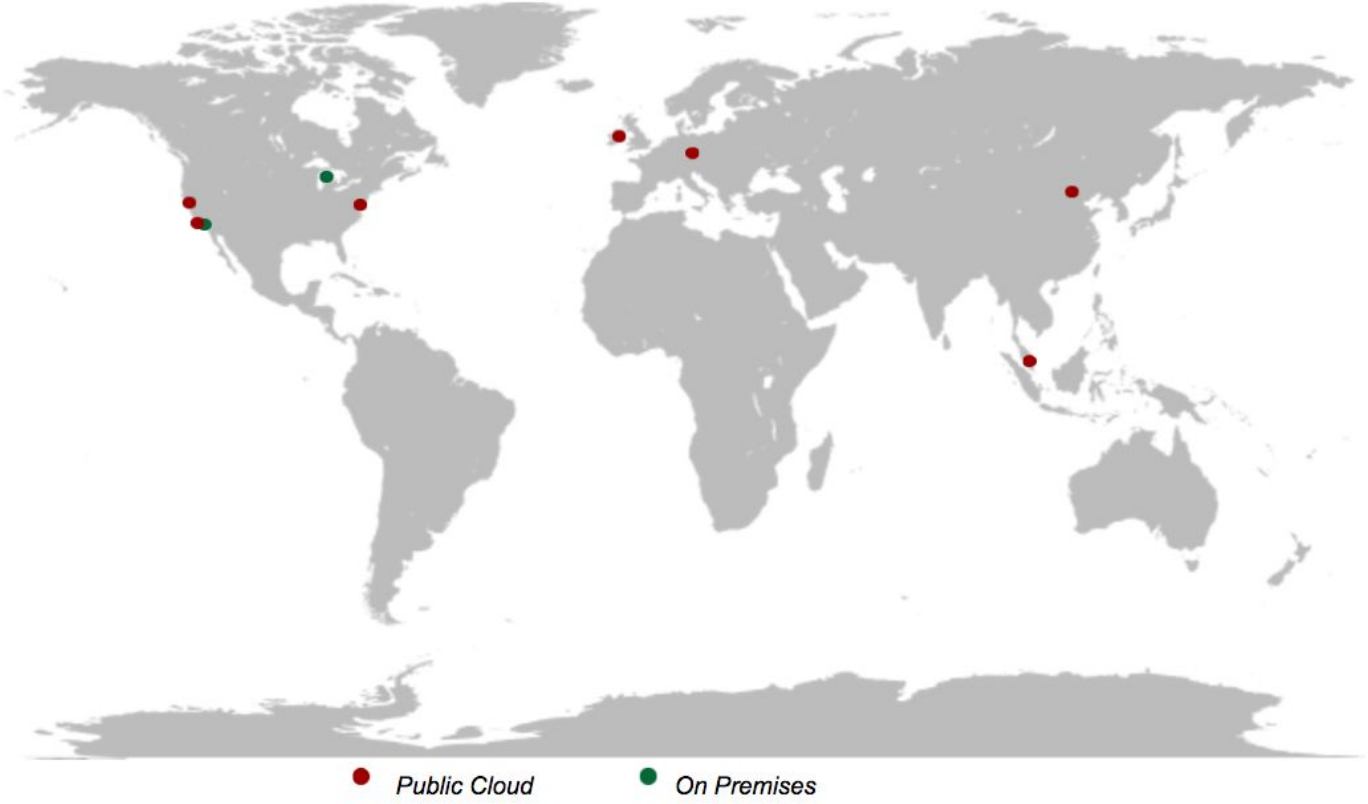
Who are we?

Operations Engineering

- Ensure the **smooth day to day** operation of the platform infrastructure
- Provide a **cost effective** and **cutting edge** infrastructure
- Team composed of SREs, SEs and DBAs
- Managing over **2,500 servers** (virtual and physical)

Our Infrastructure

Multiple locations with a mix of Public Cloud and On Premises

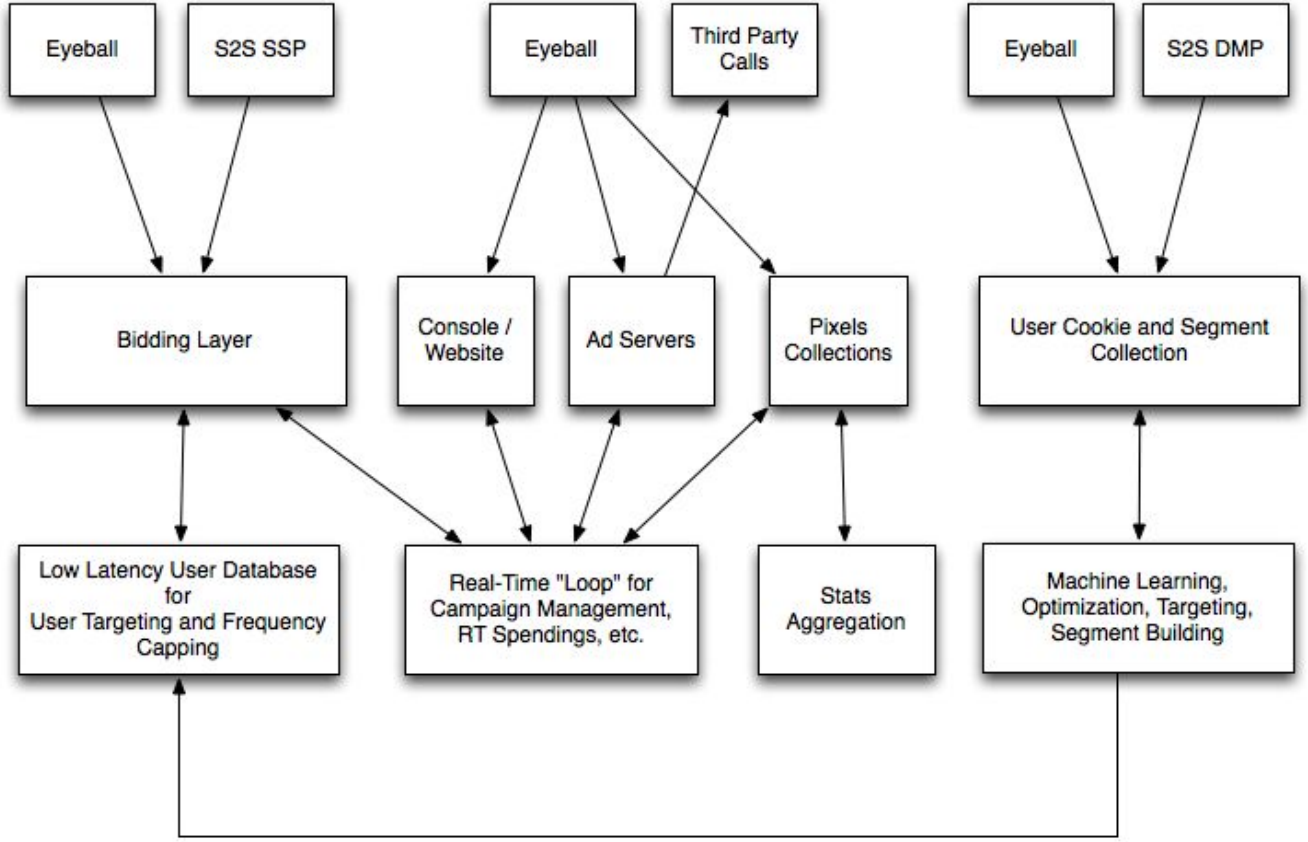


TubeMogul

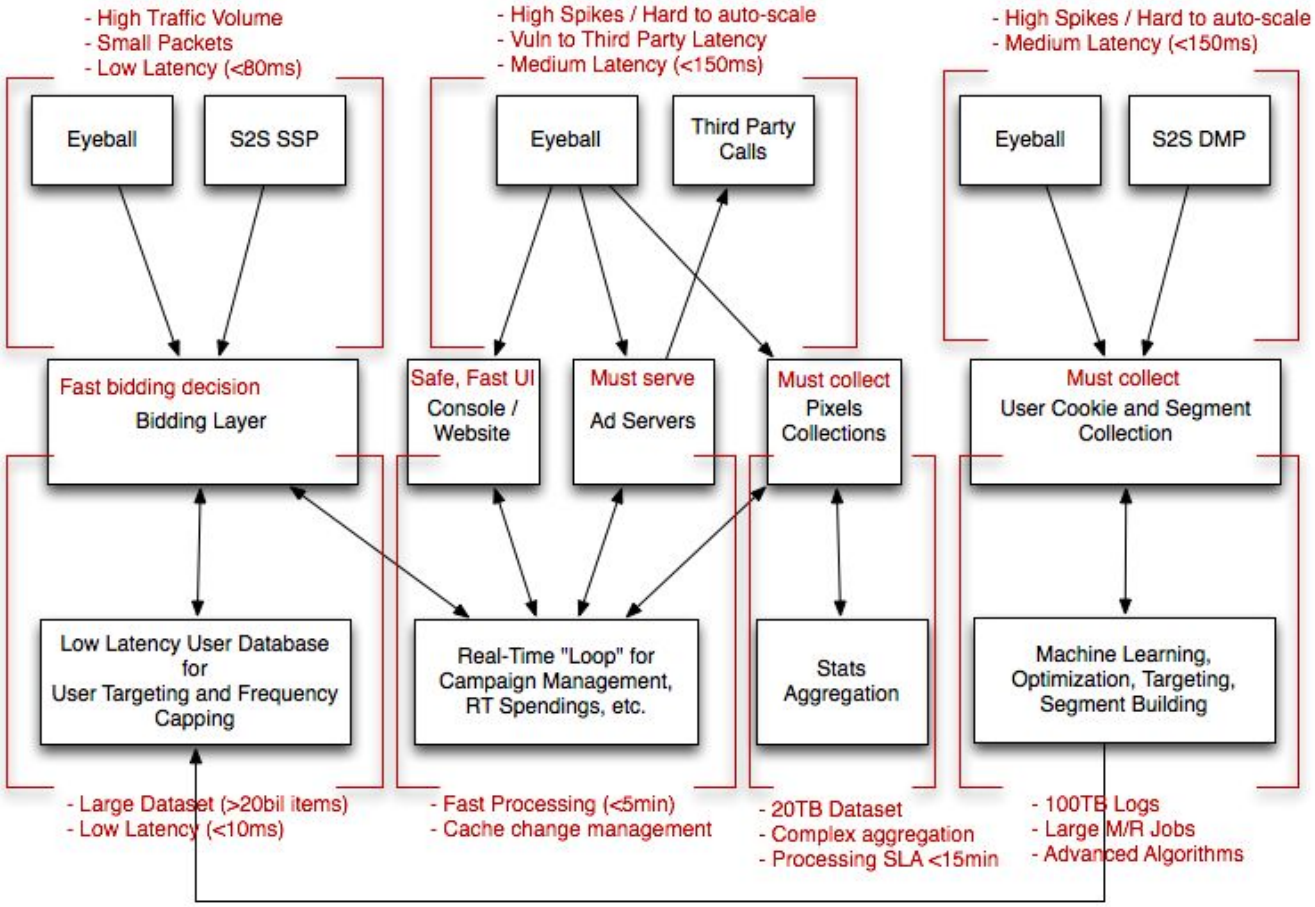
Technology Hoarders

- Java (a lot!)
- MySQL
- Couchbase
- Vertica
- Kafka
- Storm
- Zookeeper, Exhibitor
- Hadoop, HBase, Hive
- Terracotta
- ElasticSearch, Logstash, Kibana
- Varnish
- PHP, Python, Ruby, Go...
- Apache httpd
- Nagios
- Ganglia
- Graphite
- Memcached
- Puppet
- HAproxy
- OpenStack
- Git and Gerrit
- Gor
- ActiveMQ
- OpenLDAP
- Redis
- Blackbox
- Jenkins, Sonar
- Tomcat
- Jetty (embedded)
- AWS DynamoDB, EC2, S3...

High Level Technical Overview



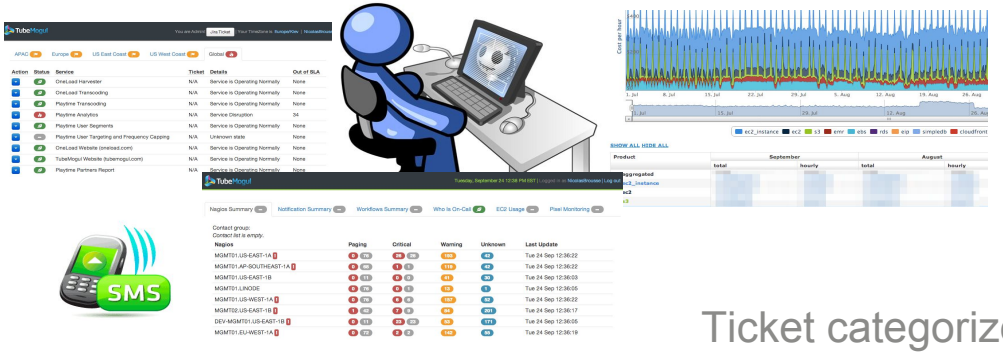
Technical Challenges



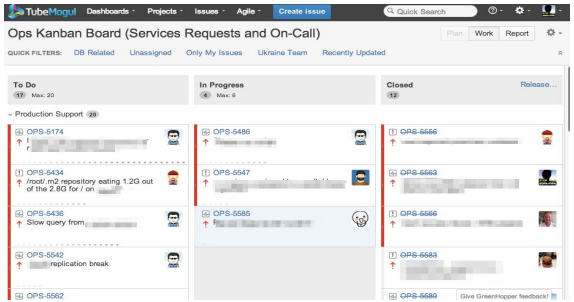
How do we manage all this?

- Tight day to day operations
- Configuration Management and Automation
- Change Management with Peer Review and CI
- Measure and Monitor a lot

OnCall Team Process



Request based on Dashboards, Monitoring, Paging or Engineers.

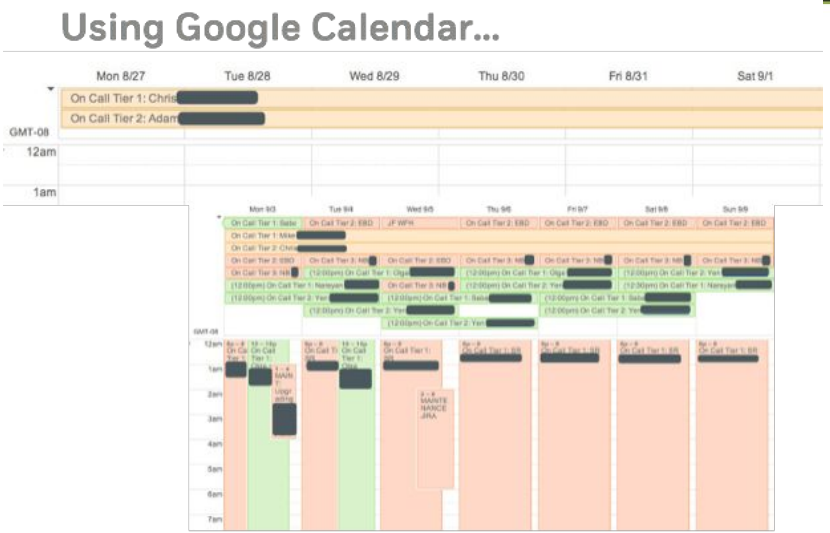


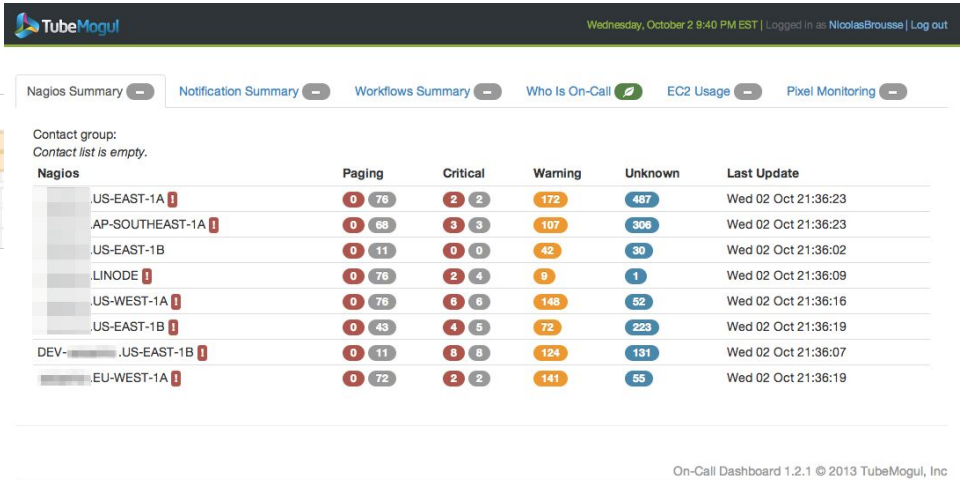
Ticket categorized in two swimlanes:

- Production Support
  - High Priority: Top to Bottom
  - On-call 24/7 (follow the sun)
  - Incident are handled 1st
  - Maintenance are handled 2nd
- Developer Support
  - Best Effort: Top to Bottom
  - Long effort request moved to Infrastructure pipeline

Alerting

- Large Nagios installation
- Introducing Sensu for scalability and as an easy Monitoring API for Developers
- Centralized OnCall Dashboard





Which Cloud Provider? Private or Public?

- TL;DR doesn't matter as long as you keep the flexibility for your dev team

- We leverage AWS for many different workload and scenarios
  - Using EC2, DynamoDB, SQS, SES, SNS, RDS, SWF, etc.
  - Workload varies from ephemeral computes to always on

- We moved part of our low latency dependent workload out of AWS to our On Premises Cloud
  - Data Center proximity to key partners
  - Performance Investigation and Tuning
  - Network Visibility
  - Business Accountability

CloudMogul with OpenStack

05/2013 → 03/2014 → 05/2015 → 08/2015

First Dev Environment

Release: Grizzly
Nodes: 12
Cores: 240
RAM: 1 TB

First Prod Environment

Release: Havana
Nodes: 40
Cores: 1136
RAM: 8 TB

First Production Switch

Release: Icehouse

Full Production

Release: Icehouse
Nodes: 144
Cores: 4464
RAM: 22 TB

Challenge of Low Latency Globally

- Geo based DNS isn't based on network performance
- Proximity to user is key
  - Reduce Latency of standard TCP Handshake
  - Reduce Latency of SSL Handshake
- Mobile Networks...
- Require a global footprint
- Large footprint means unlikely to benefit from TLS session resumption

How to ensure pixel delivery at 50ms globally on the 50th percentile while keeping a small server footprint?

Leverage CDN for Fast Pixel Collection at the Edge

fastly

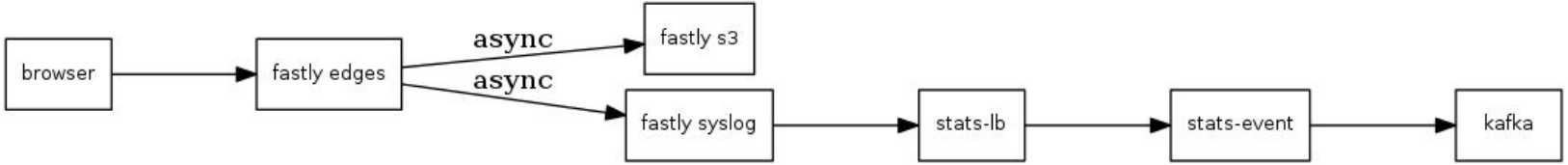
Standard pixel

Browser HTTP call conv.tubemogul.com ->
AWS US East:
Stats LB HTTP ->
Stats Event HTTP ->
Push to Kafka

Fast pixel

Browser HTTP call convf.tubemogul.com ->
Fastly CDN: Edge generate cookie or reuse cookie and return blank pixel to browser ->
Async syslog call (contain base64 string of HTTP request with headers) from Fastly to AWS US East:
Stats LB Syslog ->
Stats Event Syslog ->
Reuse Stats Event HTTP Handler ->
Push to Kafka, etc.

- Fast.ly also push all logs to S3 with data as we send via syslog. The s3 bucket can be used for troubleshooting.
- Fast.ly behavior is configured via a VLC that can be found in Gerrit.



Further Improvement

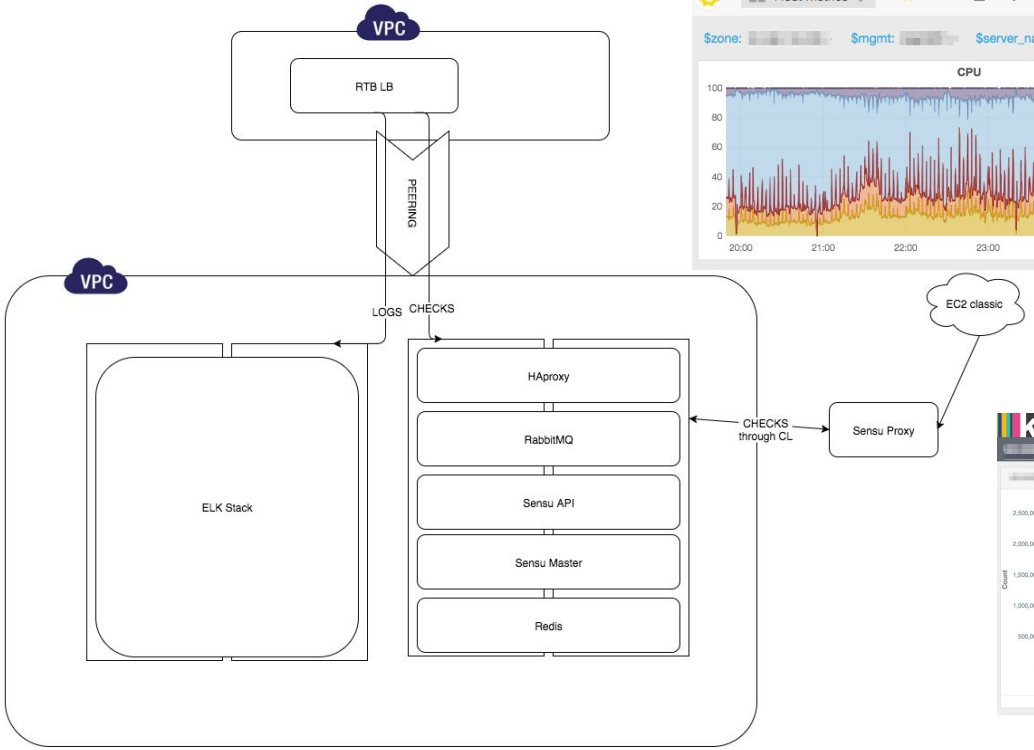
- Leverage CDN capabilities even further
  - First layer of protection against DDoS
  - Fast.ly VCL is very powerful
- Evaluate routing solution based on RUM (Cedexis)
- Evaluate smarter DNS routing (NS1)

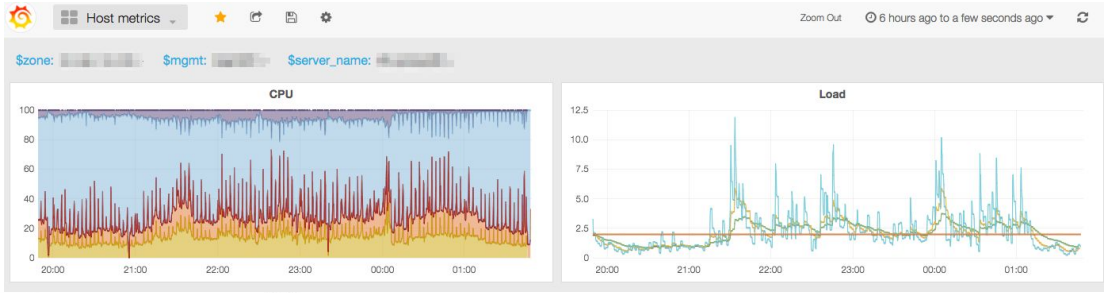
Load Balancing with HAproxy

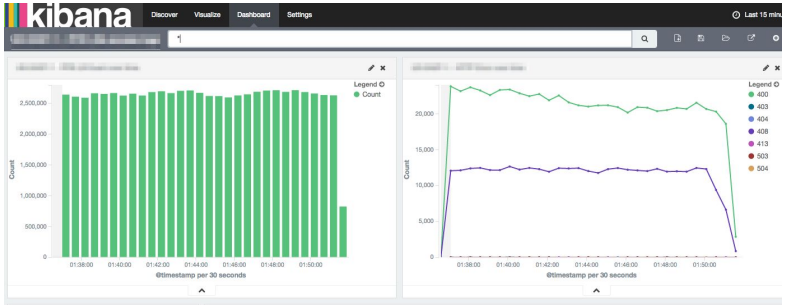
- Round Robin DNS is great
  - Until your DNS entries are too large and client start using DNS thru TCP
- In US West, we went from 31 EC2 instances (c3.2xlarge) to two SuperMicro servers
  - 32 Cores E5-2667 v3 @ 3.20GHz and 128 GB RAM
  - Use baremetal and leveraging VLAN to access OpenStack Tenant
- Managing SSL session is the most consuming in our workload (CPU and RAM)
  - A TLS connection can use up to 64Kb RAM
- CPU Pinning for network interrupts (4 Core), HAproxy (28 Core)
  - Disable irqbalance
  - Various sysctl config tuning (TCP, VM)
- One frontend for HTTP and HTTPS
- Crossdomain.xml files are served directly by HAproxy (no call to backend)
- All logs sent directly in json to ELK
- Home made process (HAvOC) to generate config and scaling of backend

Graphing and Logging As A Service

- Ganglia / Graphite / Grafana / ELK



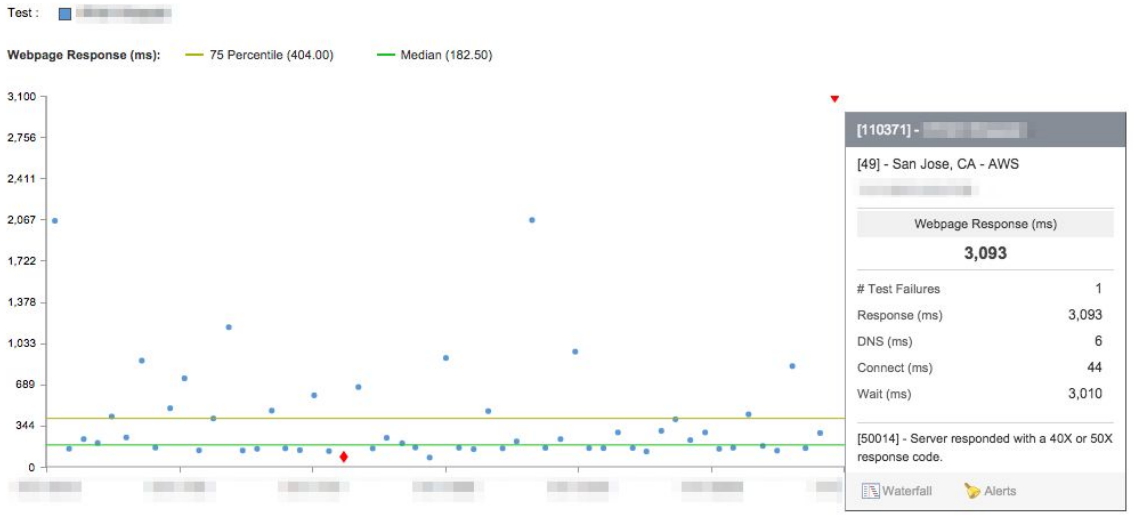




Network Visibility: Catchpoint

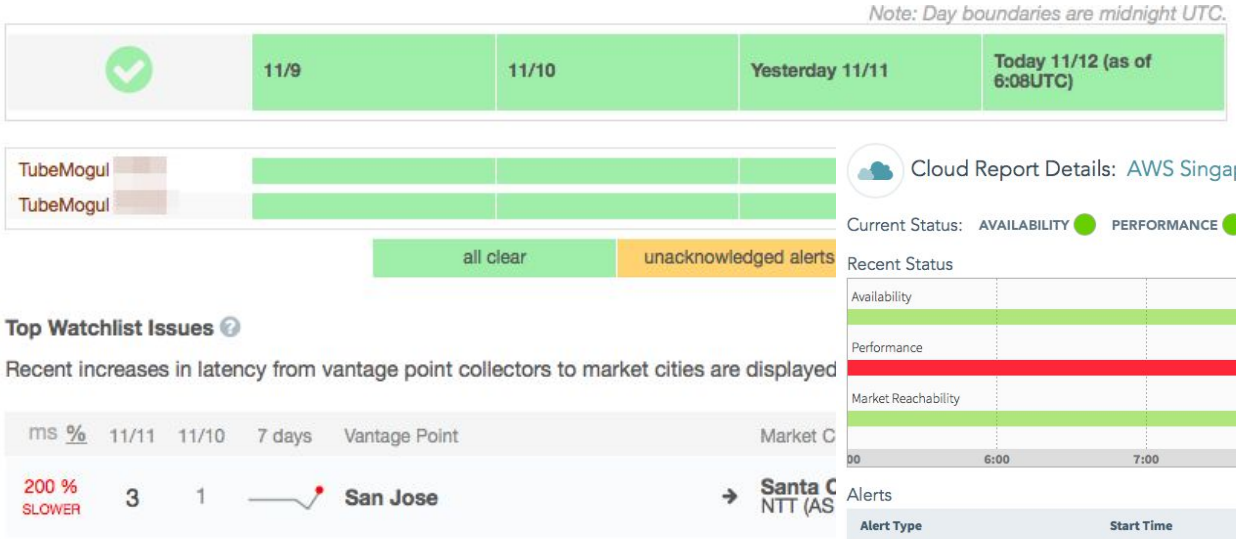
Script *:

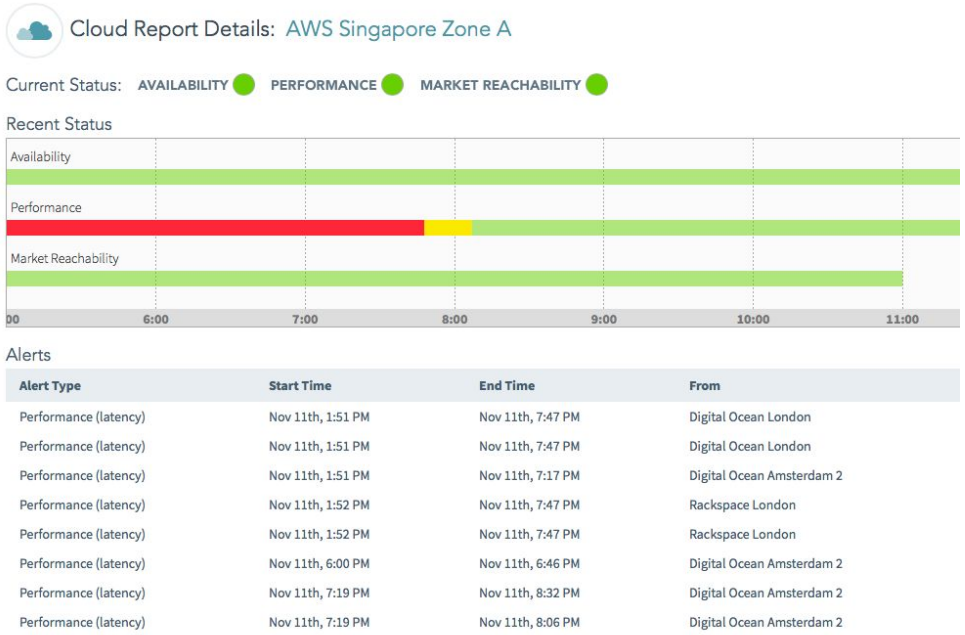
```
1  // Step - 1
2  open("http://[redacted]/crossdomain.xml")
3  setStepName("1-[redacted] crossdomain.xml")
4  assertHttpResponseCode("200")
5
6  // Step - 2
7  open("https://[redacted]")
8  setStepName("2-RTB bid request")
9  assertHttpResponseCode("200")
10
11 // Step - 3
12 open("https://[redacted]")
13 setStepName("3-AS wrapper request")
14 assertHttpResponseCode("200")
15
16 // Step - 4
17 open("https://[redacted])
18 setStepName("4-CDN swf")
19 assertHttpResponseCode("200")
20
21 // Step - 5
22 open("http://[redacted]crossdomain.xml")
23 setStepName("5-AS crossdomain.xml")
24 assertHttpResponseCode("200")
25
26 // Step - 6
27 open("https://[redacted]")
28 setStepName("6-ad server placement.js")
29 assertHttpResponseCode("200")
30
```



Monitor from multiple location globally, complex test, trace routes, alerts, etc.

Network Visibility: Dyn Internet Intelligence





Alerts on bgp route changes, prefix changes, latency variation, internet disruptions, etc.

Five Years Of Puppet!

- 2008 - 2010: Use SVN, Bash scripts and custom templates.
- 2010: Managing about **250 instances**. Start looking at Puppet.
- 2011: Puppet 0.25 then 2.7 by EOY on **400 servers** with **2 contributors**.
- 2012: **800 servers** managed by Puppet. **4 contributors**.
- 2013: **1,000 servers** managed by Puppet. **6 contributors**.
- 2014: **1,500 servers** managed by Puppet. Introduced Continuous Delivery Workflow. **9 contributors**. Start 3.7 migration.
- 2015: **2,000 servers** managed by Puppet. **13 contributors**.

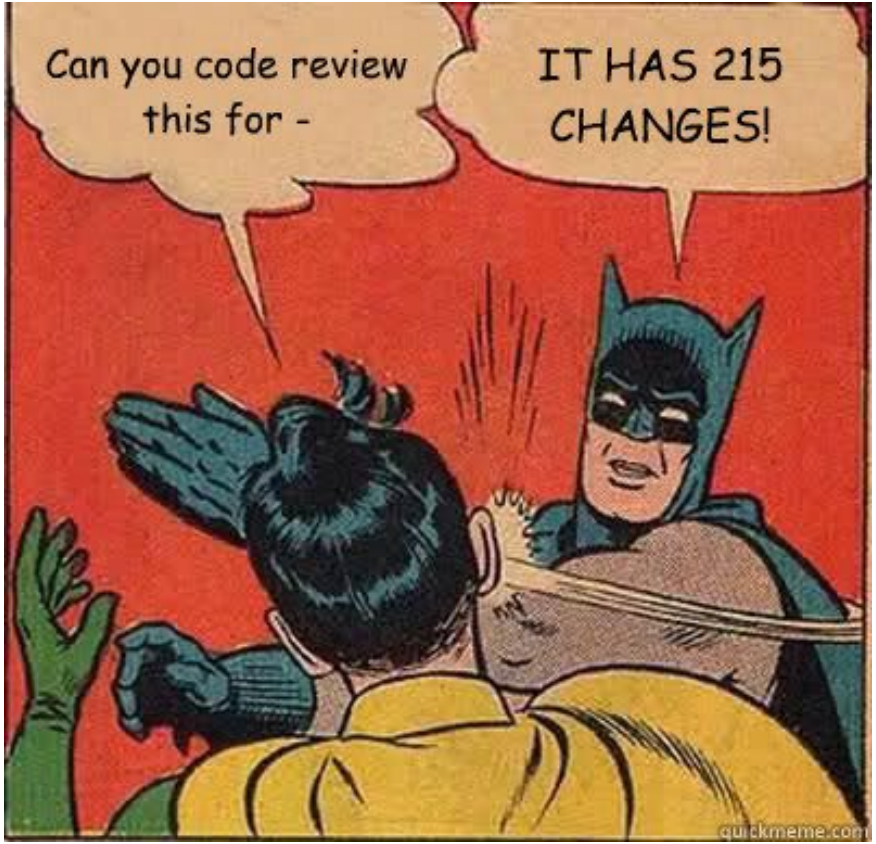
Puppet Stats

- **2000** nodes
- **225** unique nodes definition
- **1** puppetmaster
- **112** Puppet modules

Where and how do we use Puppet ?

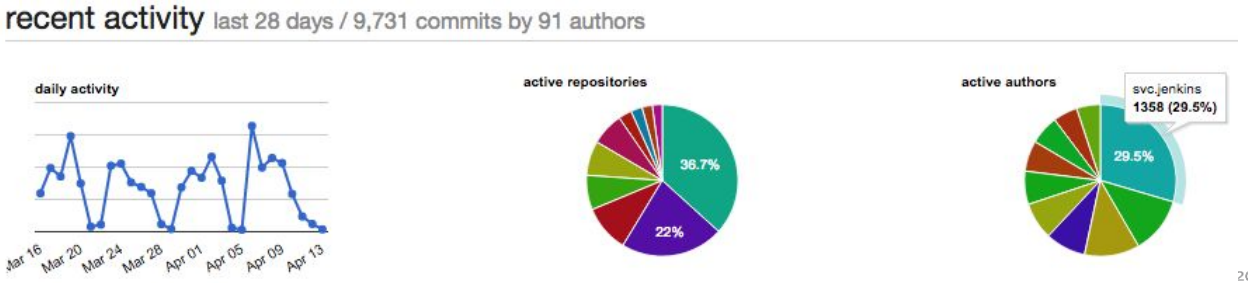
- Virtual and Physical Servers Configuration : **Master mode**
- Building AWS AMI with Packer **: Master mode**
- Local development environment with Vagrant : **Master mode**
- OpenStack deployment : **Masterless mode**

Infrastructure As Code: Code Review?



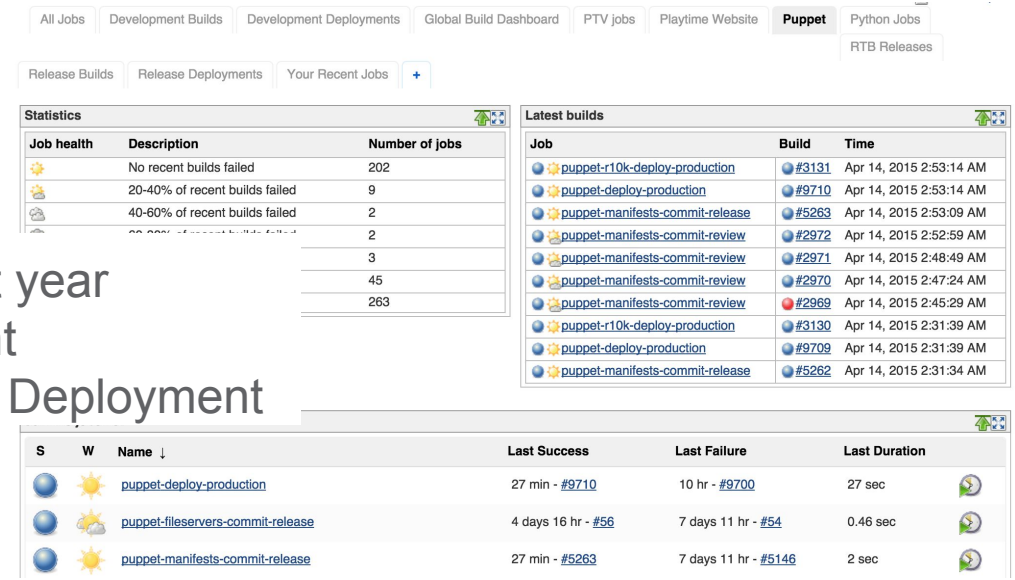
A Powerful Gerrit Integration

- Gerrit, an **industry standard** : Eclipse, Google, Chromium, OpenStack, WikiMedia, LibreOffice, Spotify, GlusterFS, etc...
- Fine Grained Permissions Rules
- Plugged to LDAP
- Code Review **per commit**
- Stream Events
- Use GitBlit
- Integrated with Jenkins and Jira
- Managing about **600 Git repositories**



Continuous Delivery with Jenkins

- **1** job per module
- **1** job for the manifests and hiera data
- **1** job for the Puppet fileserver
- **1** job to deploy



Global Jenkins stats for the past year

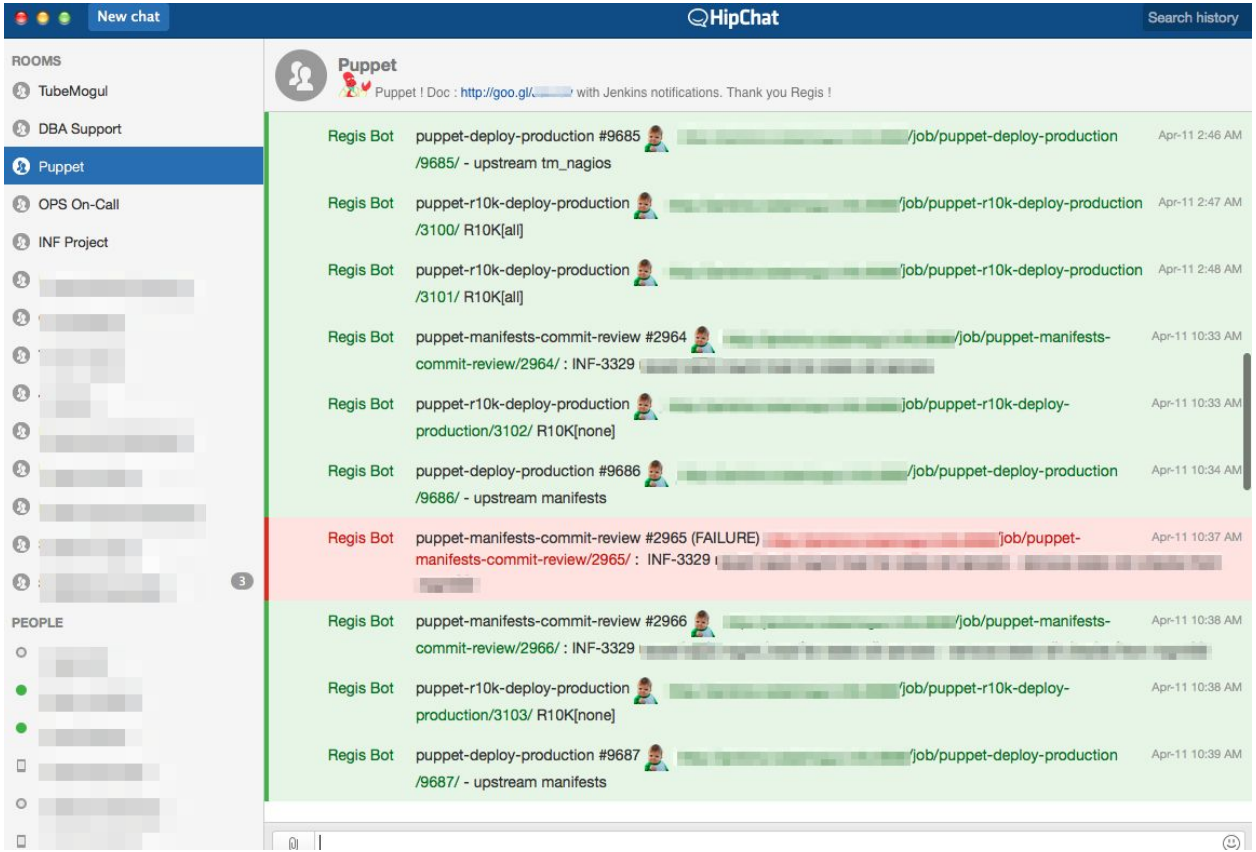
- **~10,000** Puppet deployment
- Over **8,500** Production App Deployment

Jenkins job DSL : code your Jenkins jobs

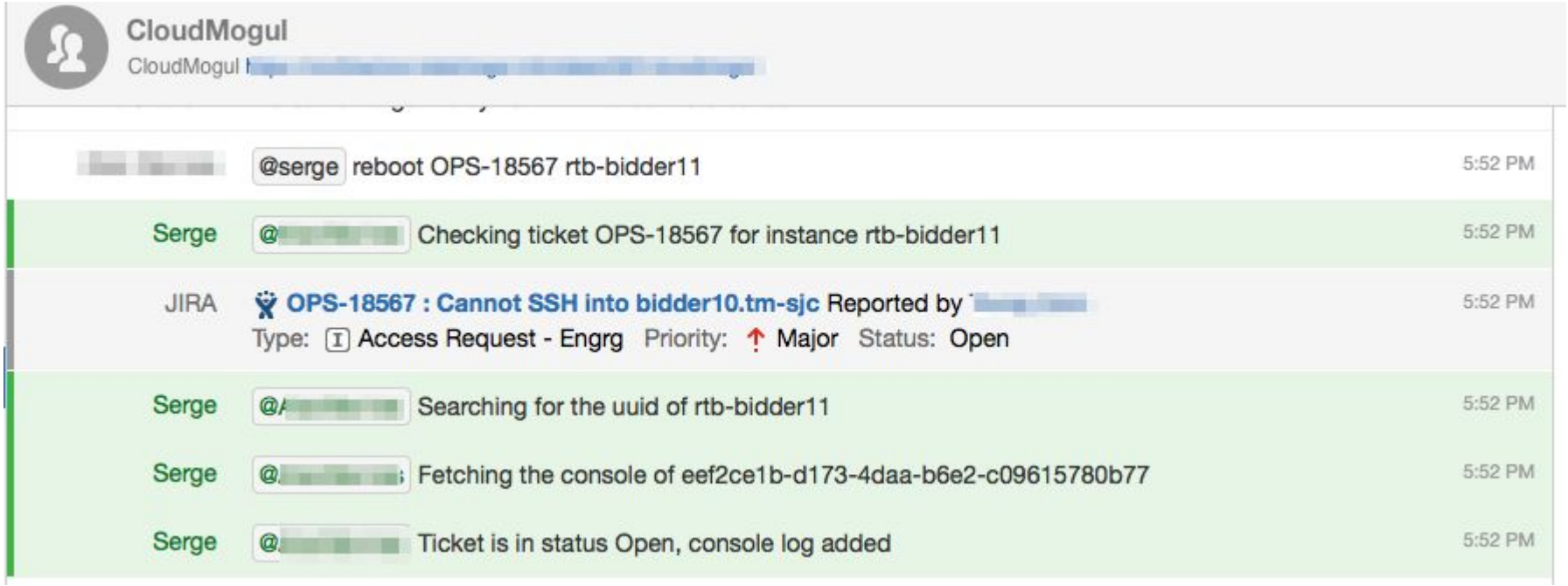
Plugin : github.com/jenkinsci/job-dsl-plugin

- **Automate** the jobs creation
- Ensure a **standard** across all the jobs
- **Versioned** the configuration
- Apply changes to all your jobs **without pain**
- **Test** your configuration changes

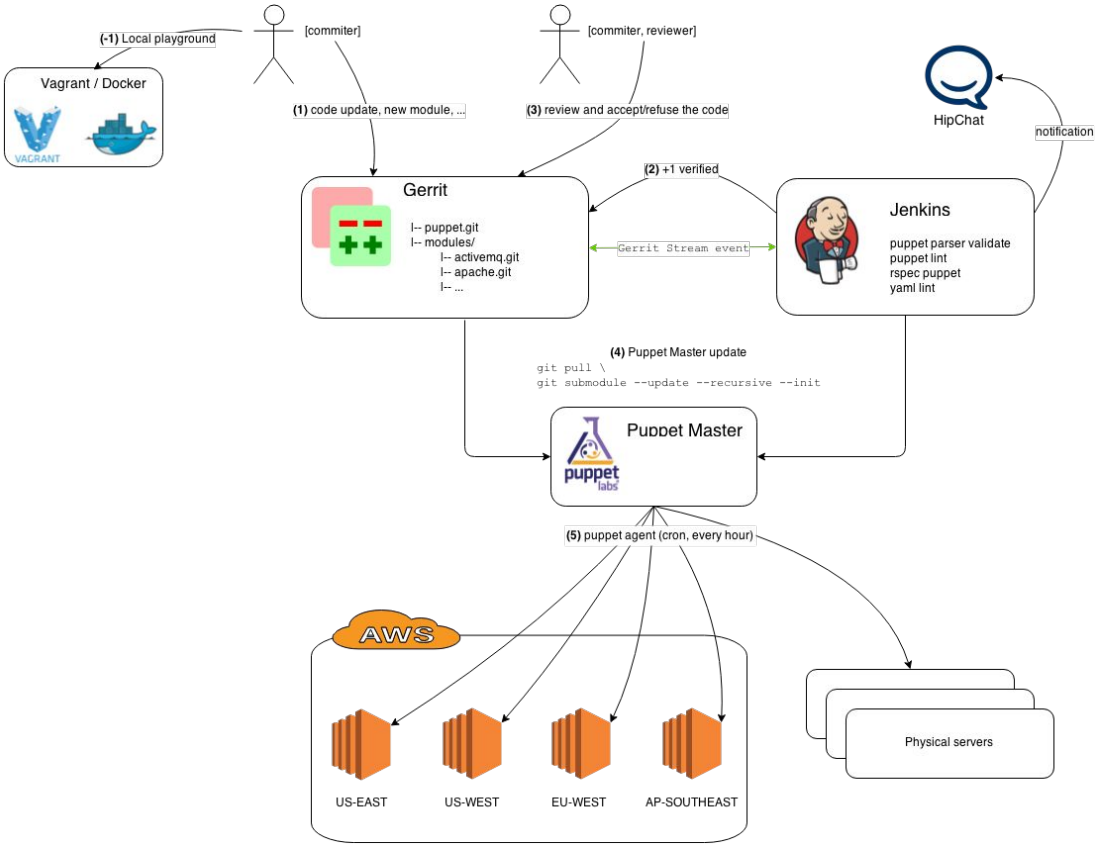
Team Awareness: HipChat Integration with Hubot



Team Awareness: HipChat Integration with more bots!



The Workflow



OPS @ TubeMogul

All This Wouldn't Be Possible Without a Strong Team.
Thank You.

SRE

Aleksey Mykhailov
Oleg Galitskiy
Brandon Rochon
Stan Rudenko
Julien Fabre
Joseph Herlant

SE

Alan Barnes
Aleksander Stepanov
Matt Cupples
Yurii Rochniak
Yurii Varvynets
Manasi Limbachiya

Cloud Engineer

Mykola Mogylenko
Pierre Gohon
Pierre Grandin

DBA

Alina Alexeeva

THANK YOU

www.tubemogul.com | @tubemogul

Nicolas Brousse @orieg