



<https://github.com/lithiumtech/presentation-sfcloudops-20150312>

About Matthew Bogner

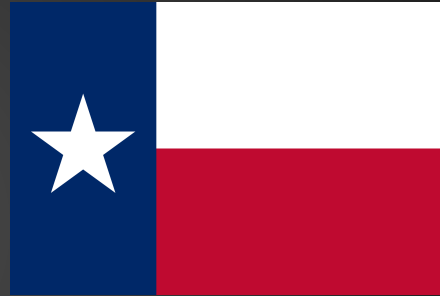
Principal Software Engineer at Lithium.

I love fault-tolerant engineering and geek out on telemetry. Data is cool.

Nobody knows it
My son Luke is the brains behind me
We have many late night coding sessions ;)

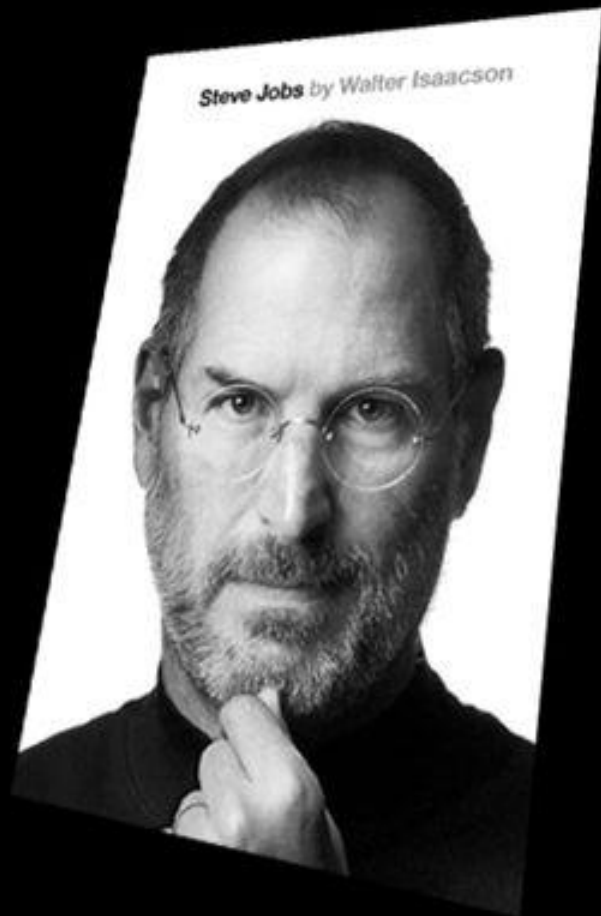


Parental Advisory



“People who know what
they’re talking about
don’t need PowerPoint.”

— Steve Jobs
From Walter Isaacson’s
book *Steve Jobs*



Team (I didn't hero this shit myself)

Paul Allen



Alan Lippert



What the app does, in 10s

Social customer service (& some other stuff)

If you “express yourself” on twitter
someone from brand answers quickly

App takes care of routing and threading
conversations to support agents

Back in the day



Then we got bought

complex migration to colo

but... blessing in disguise

enforced consistency (we shook out the
“startup-effect”)

Then we grew



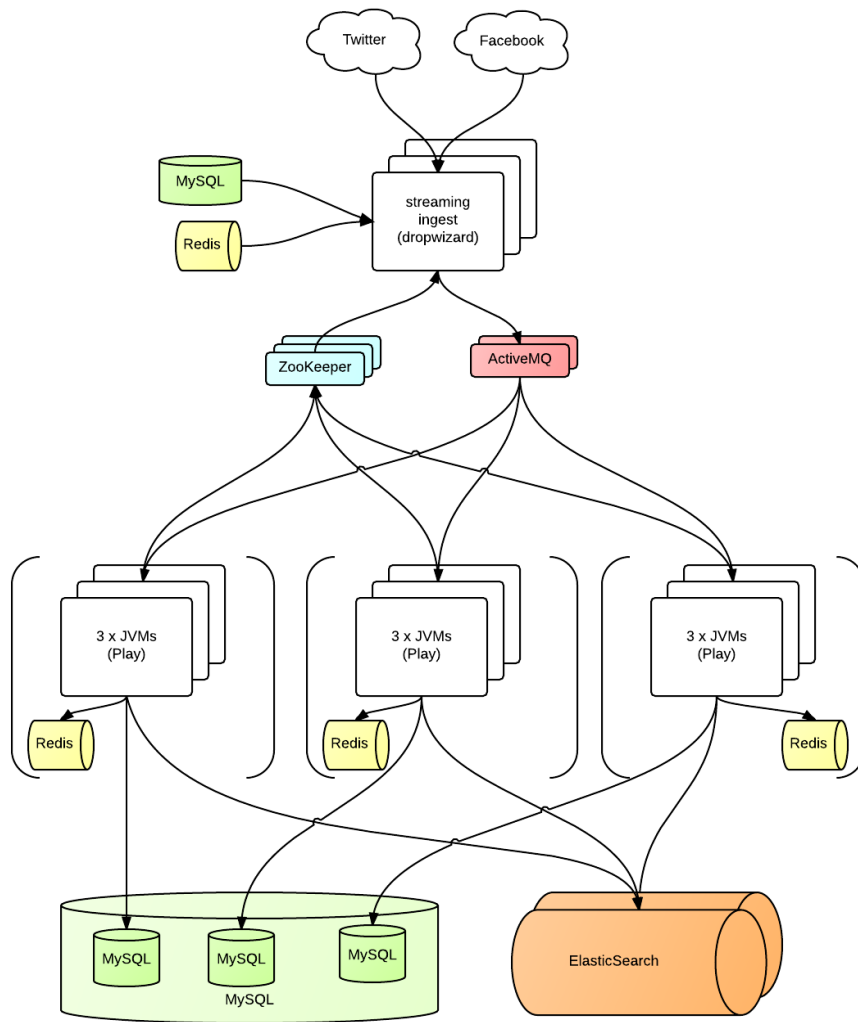
Then we needed flexibility

near real-time
analytics and
trending

and expected
multiples of
current traffic



Pieces of the application



Cloudformation, ASGs and VPCs


This presentation assumes you are an advanced AWS user and that you understand Cloudformation, Auto-Scaling Groups and VPCs

- VPCs
 - Software defined networking. Think souped up Cisco vlans & iptables
- Cloudformation
 - Templates that define your infrastructure instead of integrating to eleventy billion zillion separate APIs
- ASGs
 - common notation in this preso “(x, y)” means an ASG with min X and max Y and no triggering


Let's tackle each piece

- VPCs
 - Cloudformation
 - Config management
 - Code (yum) repo
 - Build system
 - DNS
-

- MySQL
- ElasticSearch
- Redis
- ActiveMQ
- Zookeeper
- java app(s)

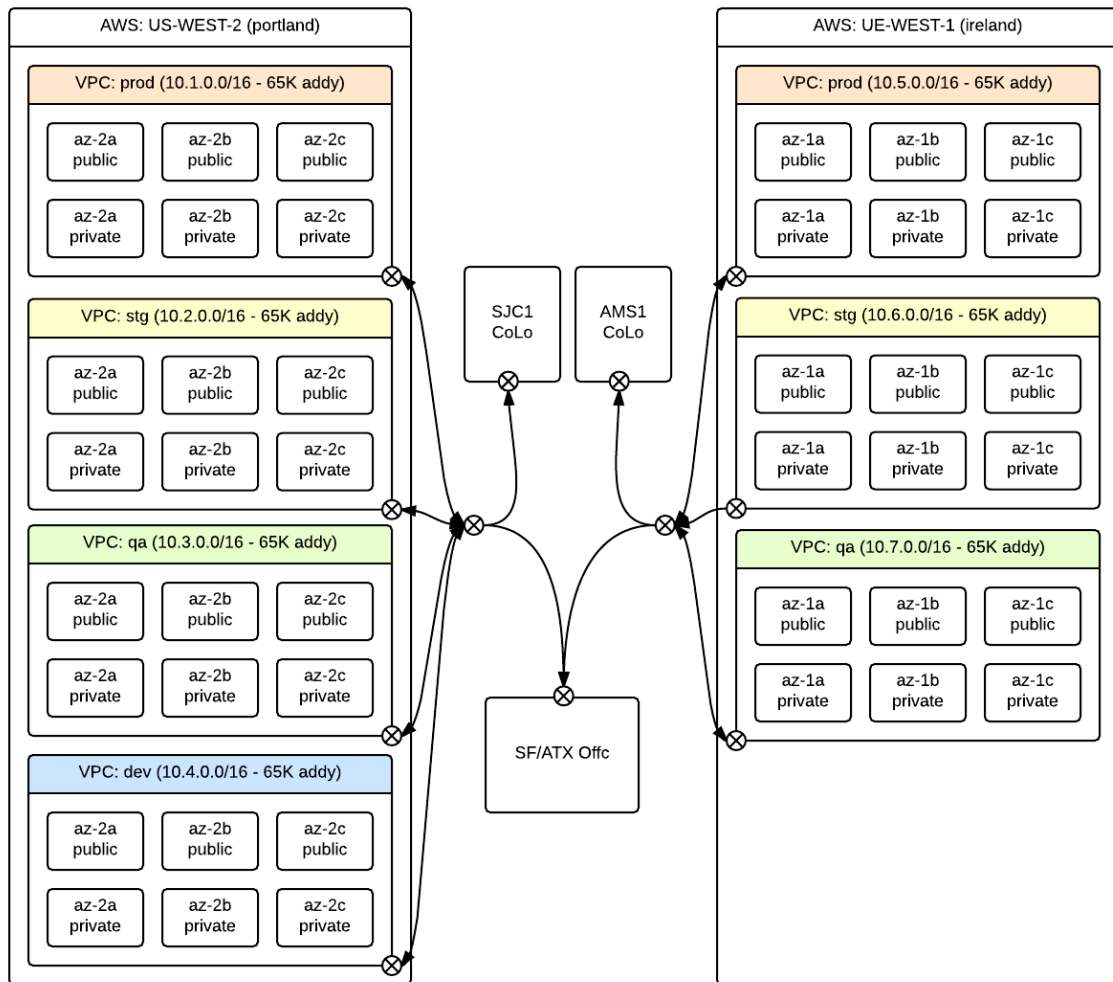


oft'
forgotten
pre-reqs



All that your
manager actually
understands you
are doing (sad
but true)

Set up the VPCs



A note on AWS region selection

Quorum based software like ZooKeeper, ElasticSearch, HBase, Kafka, Cassandra, etc. is dangerous with only 2 AZs.

Lose 1 AZ and you have 50/50 shot of surviving

Regions with 2 AZs:

- **us-west-1** (san fran)
- sa-east-1 (sao paulo)
- eu-west-2 (frankfurt)
- ap-southeast-2 (sydney)
- ap-southeast-1 (singapore)

Regions with > 2 AZs:

- us-east-1 (virginia)
- us-west-2 (portland)
- eu-west-1 (dublin)
- others...

cloudformation

- Don't mix manual f***ery with cloudformation.
 - This actually is blessing in disguise. Keeps ppl from doing dumb shit b/c they know it won't play nice with CFN.
 - Updating an existing cloudformation stack is very predictable as long as nobody is manually screwing with your stuff.
 - Most things you do to an ASG within a CFN template require you to manually terminate and relaunch instances before the change takes place (this is a good thing)
- Separate your long-lived things from your ephemeral things into sep tmpls
 - ELBs, S3 buckets, Route53 entries in one template
 - ASGs, et. Al. in another template

Config mgmt

They're a friggin' dime a dozen these days. I knew puppet from previous life. The team at Lithium wanted to use Chef. In all honesty, didn't really care which CM solution. They all work.

In hind-sight, I like Chef better than Puppet. Just personal opinion. It is more straightforward.

“Regular” Chef or Chef-Solo ?

- We started w/ a chef master
- Had 3 or 4 teams all overwriting each others stuff. Was uber-frustrating to be honest.
- Moved to chef-solo for this app stack
- Secondary benefit was eliminated single point of failure with chef master (what'ya do when that guy dies?)
 - Don't need chef search anyway

<https://github.com/lithiumtech/presentation-sfcloudops-20150312/blob/master/example-bootstrap-chef-solo.sh>

We need a private code (yum) repo

We need a place to put our proprietary bits.

I mean, tarballs on FTP is so 1990.

Actually, did FTP exist in 1990?

Maybe gopher:// instead.

I digress...

yum-s3-repo-manager

Open Source!

<https://github.com/lithiumtech/yum-s3-repo-manager>

<https://github.com/lithiumtech/presentation-sfcloudops-20150312/blob/master/example-bootstrap-s3-yumrepo.sh>

yum-s3-repo-manager

Place an RPM in the “inbox” s3 folder, and the monitoring process (checks every 60s) puts anything it finds into the repo, recomputes the yum repo metadata and then uploads back to s3.

Boxes talk to s3 as the repo instead of the repo mgr or a specific server. S3 has 100-ish% uptime. This has been nice on multiple occasions for us.

Build System - Jenkins

- We used to use Atlassian Bamboo (\$\$, closed source, blah)
- major hurdle with jenkins was making it sufficiently HA to power our infrastructure.
CAVEAT - this solution as it is described here binds jenkins to a single AZ
 - create-persistent-ebsvolume.sh
 - <https://github.com/lithiumtech/presentation-sfcloudops-20150312/blob/master/example-jenkins-persistent-ebsvolume.sh>
 - launch cloudformation stack (on bootstrap of instance attach to persistent EBS volume using tag discovery)
 - <https://github.com/lithiumtech/presentation-sfcloudops-20150312/blob/master/example-jenkins-filesystem-fun.sh>
- Use [ec2-plugin](#) to launch instances as needed and tear them down when not in use.
 - Guilty admission, we use c3.xlarge for a lot of our builds
- Use [packer](#) to pre-bake jenkins slave AMIs (mysql, elasticsearch, redis, bunch of other stuff already on it)

DNS - managing bind sucks

Do we **really** need it? (IMO, life without DNS sucks). This conversation was a challenge here at Lithium.


Long story short - Route53 for internal stuff, and existing provider to overlay to externally-facing ELBs. (interesting fact, Route53 has added private DNS for requests originating within the VPC since we did this)

<https://github.com/lithiumtech/presentation-sfcloudops-20150312/blob/master/example-route53-registration.sh>


Now we have tackled the pre-req's

- VPCs
 - Cloudformation
 - Config management
 - Code (yum) repo
 - Build system
 - DNS
-

- MySQL
- ElasticSearch
- Redis
- ActiveMQ
- Zookeeper
- java app(s)



off'
forgotten
pre-reqs



All that your
manager actually
understands you
are doing (sad
but true)

MySQL

- No RDS - we have smart DBAs and complex MySQL infrastructure and requirements
- Want to be able to tolerate instance and AZ failure in AWS without manual intervention and rapid failover
- Use [MHA](#) by Yoshinori from Facebook and [MHA-Helper](#) by our very own Ovais Tariq
- This was by far the most complex thing I've personally EVER automated
 - so much so, that I'm going to talk about it for next few slides

MySQL Topology

2 ASGs

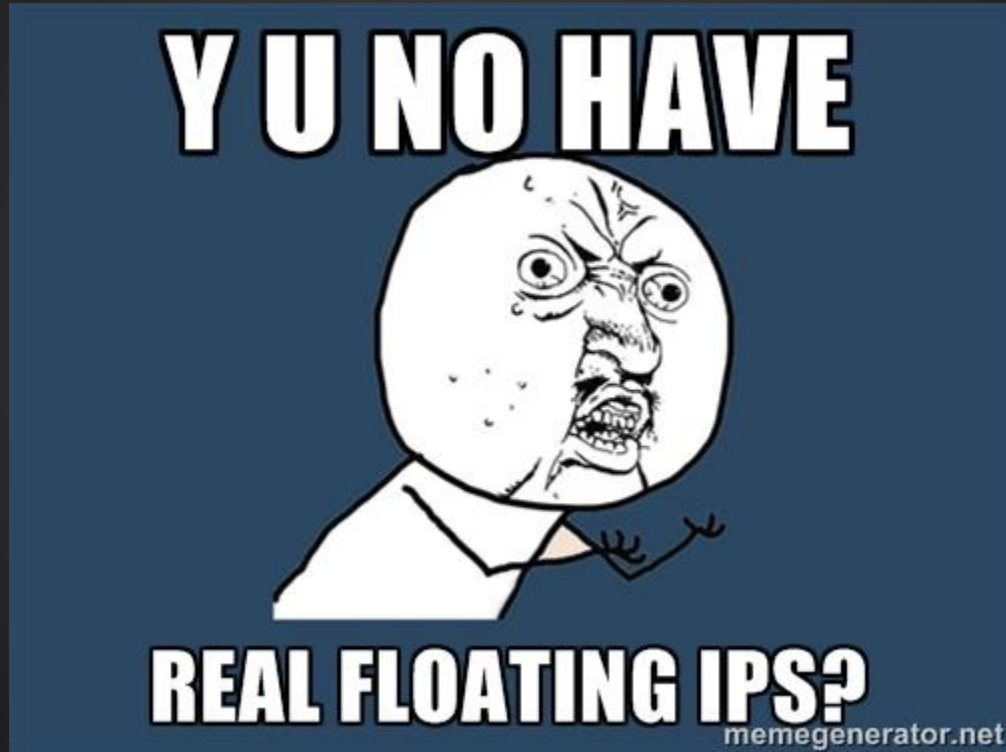
- master-eligible ASG (3,3)
 - 1 master
 - 1 read slave
 - 1 backup slave
 - -- if we made this (5,5) we'd end up with 3 read slaves
- mha manager ASG (1,1)

MHA

- Upon creation of stack, selection of master and MHA manager done to prevent them being in the same AZ
- Any failure after that is based on most up-to-date slave (logic courtesy MHA)
- MHA requires virtual IPs to facilitate application failover

Virtual IPs in AWS for MySQL

Hold my
beer and
watch
this...



Floating IPs in AWS - problem

- Need cross-AZ IP address
- DNS failover doesn't have crisp boundary across all hosts and takes too long.
Applications can cache the IP mapping too
- Private IP ENI is bound to a specific AZ and public elastic IP is ... public... and don't want my DB to have public IP

Floating IPs in AWS - solution

- Use VPC route tables
- Put route in route-table to route 192.168.x.y/32 to specific ENI
- Upon failover, update existing routes to point to new ENI of the new master
- Update and traffic switch controlled at network layer and is pretty much instantaneous

<https://github.com/lithiumtech/presentation-sfcloudops-20150312/blob/master/example-ipswitch.sh>

MySQL end result

Failover in about 11s from detection of “down”
Newly created slaves will auto-bootstrap themselves from last backup and start slaving from master without manual intervention (soon).

(slow clap)

elasticsearch

- whoops! old ass version (0.19.10) of elasticsearch (circa 2010). time to break out the IDE since client in existing java app won't talk to new cluster version via thrift
 - write to multiple clusters simultaneously
 - only read from one
 - use this tactic to bulk reindex all content offline
- use AWS plugin and tagging to facilitate discovery
- (3,3) cross-AZ ASG
- elasticsearch is (compared to everything else) easy to run in AWS
- biggest tip is to really make sure open file descriptor limits are raised to something higher than default.

elasticsearch - cont'd

Ended up using ES's thrift client to interact with the old cluster, and then use Jest to interact with new cluster since can't have two thrift clients on classpath at same time.

<https://github.com/searchbox-io/Jest/tree/master/jest>

Contribution.... (idle cxn reaper)

<https://github.com/searchbox-io/Jest/pull/149>

Zookeeper

- used Netflix's [exhibitor](#) to make bootstrapping cluster and node IDs much easier
 - We didn't even have to modify it
 - Handles backups and log rotation and stuff
 - “just works”
 - OMG - don't store persistent data in ZK
- (3,3) cross-AZ ASG
- TIP: put zookeeper entirely on ephemeral SSD disk. ZK is very sensitive to disk IO latency.

Redis

- For the record, multi-AZ ElastiCache Redis *with automated master failover* didn't exist when we started.
- Long story short, just use ElastiCache Redis.
- I don't recommend doing what I'm about to describe as it is causing us some grief currently.

Redis - part 2

- python side-car process that connects to ZooKeeper with [Kazoo](#) library
- Side-car competes for leader election with other nodes in redis ASG.
- Elected leader becomes the master and the non-leader side-cars slave from the master and push backups to S3 hourly
- Problem: the sidecar keeps disconnecting from zookeeper prematurely and we have un-necessarily frequent master failover

ActiveMQ → RabbitMQ → ActiveMQ → SQS

- Ugh... another black eye
- ActiveMQ 5.7 currently in use in colo
- ActiveMQ 5.10 added support for zookeeper integration to elect a master and have seamless failover
 - unfortunately activemq 5.10 doesn't even start out of the box b/c of classpath dep problems - wtf?!?!?!11
 - Even when you work around that problem, failover seems to replay every message in ActiveMQ's logs to the new master - even messages that have already been delivered and acked by a consumer

ActiveMQ → RabbitMQ → ActiveMQ → SQS

- So... RabbitMQ looked cool
- Nobody here knows Erlang (those logs are weird)
- RabbitMQ clustering worked great in QA, Stage and perf testing
- Put it in prod... (fireworks)
 - split-brain
 - lack of quorum
 - any hiccup in cross-AZ connectivity and it loses it's shit

ActiveMQ → RabbitMQ → ActiveMQ → SQS

- In fit of rage at 11pm, I switched everything back to ActiveMQ
- And yes, we even chose the “pause_minority” value for “cluster_partition_handling”
 - !@#\$%^&*
 - <http://www.rabbitmq.com/partitions.html>
- Now we are on active path to using SQS
 - Let AWS deal with it



ONLY??!?!?

java apps

- This was the easy part
- Our backend and front-end java apps were already horizontally scalable
- Just kind of tip of the iceberg and a ton of in-house experts, so we sailed through the automation of this.
- Discovery - combination of ZooKeeper and EC2 tag discovery with each chef run

Security - Encryption-at-Rest

- Local ephemeral disk
 - YES! You CAN encrypt the local ephemeral disk (as long as it is not also the root volume)
 - <https://github.com/lithiumtech/presentation-sfcloudops-20150312/blob/master/example-ephemeral-disk-encryption.sh>
- EBS storage
 - avoid temptation to try BlockDeviceMappings in CFN b/c it doesn't support specifying encrypted volume option (I asked multiple times) -- at least not as of 1/1/2015
 - Create volume using AWS CLI with "--encrypted" flag during bootstrap and attach, mount & format immediately before first chef run (or do it with chef, whatever).
 - <https://github.com/lithiumtech/presentation-sfcloudops-20150312/blob/master/example-attach-encrypted-ebsvol.sh>

putting it all together

- Everyone needs wrapper scripts, right?
 - shitty `deploy.sh` scripts
 - competing with shitty `deploy.rb` scripts
 - competing with non-shitty `deploy.py` script
- `.py` won
 - this makes me happy inside - I can't stand ruby
- These `py` scripts wrapper calls to AWS CLIs and provide uber light-weight orchestration with Fabric

Fabric

<http://www.fabfile.org/>



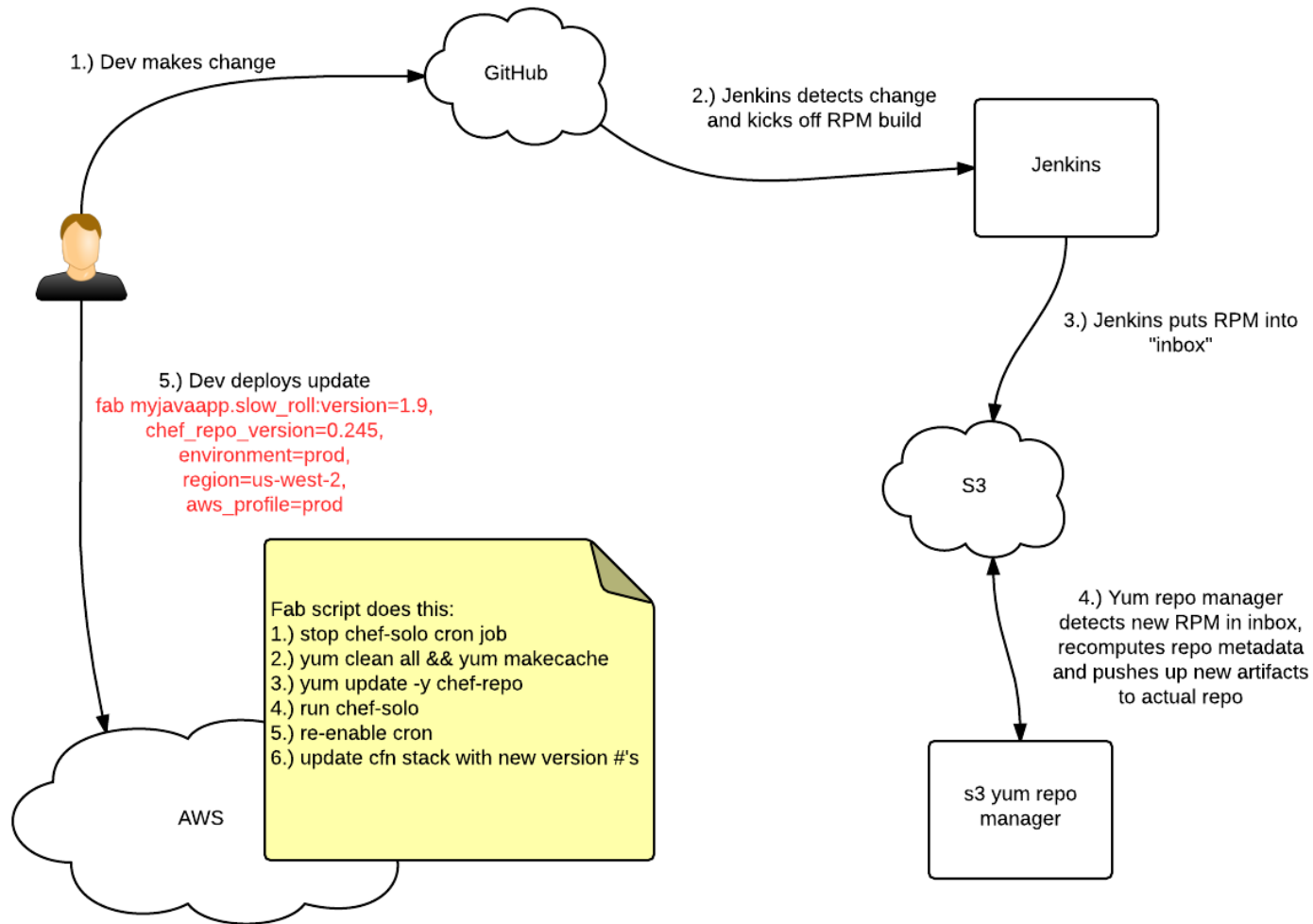
We use fabric for performing administrative tasks:

thread/heap dumps

restarts

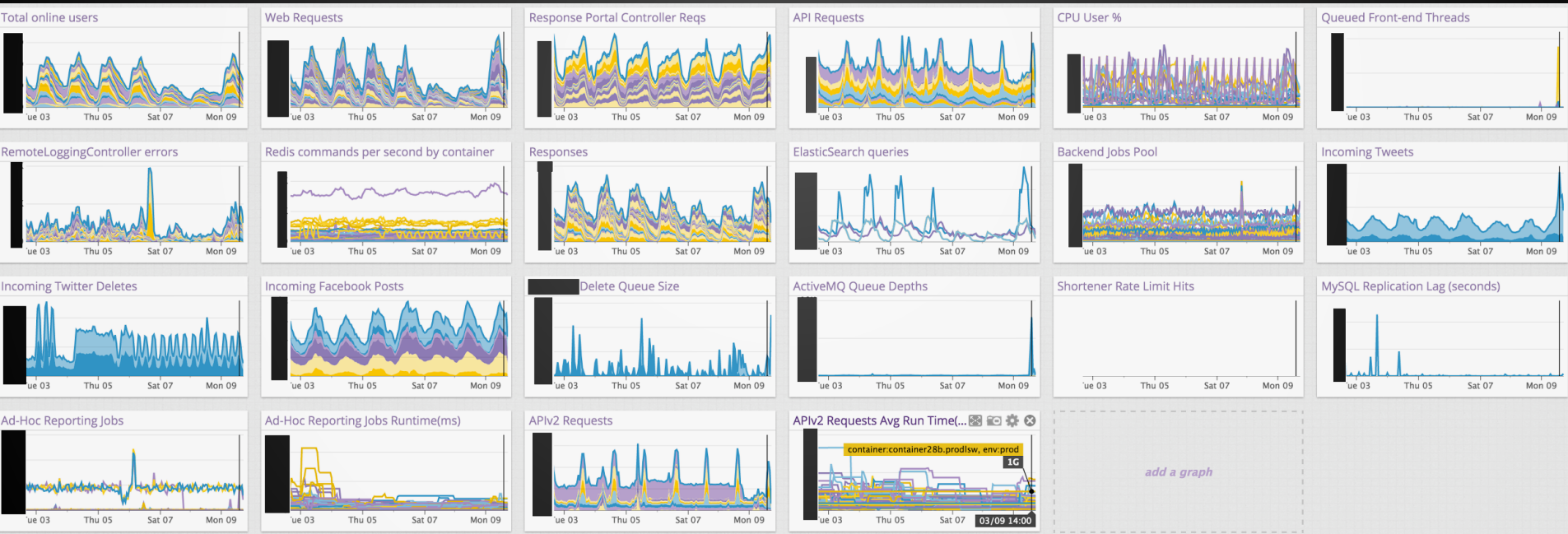
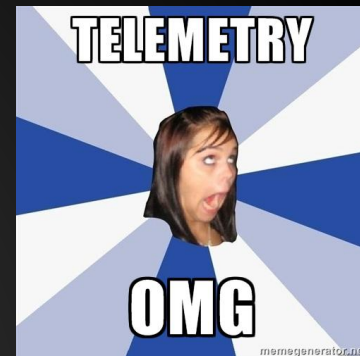
hotfix deploy orchestration, etc.

E2E Flow



Datadog and Pagerduty

telemetry and alerting



Sumologic

Centralized logging

Installed and configured by chef

Questions?

We're hiring (my manager made me do this)

<http://www.lithium.com/company/careers/job-openings>

matthew.bogner@lithium.com

<https://github.com/matthewbogner>