Cloud Operations at Netflix:
Optimizing Innovation Speed
While Supporting Availability.

Flowcon November 2013

Roy Rapoport
@royrapoport rsr@netflix.com
www.linkedin.com/in/royrapoport

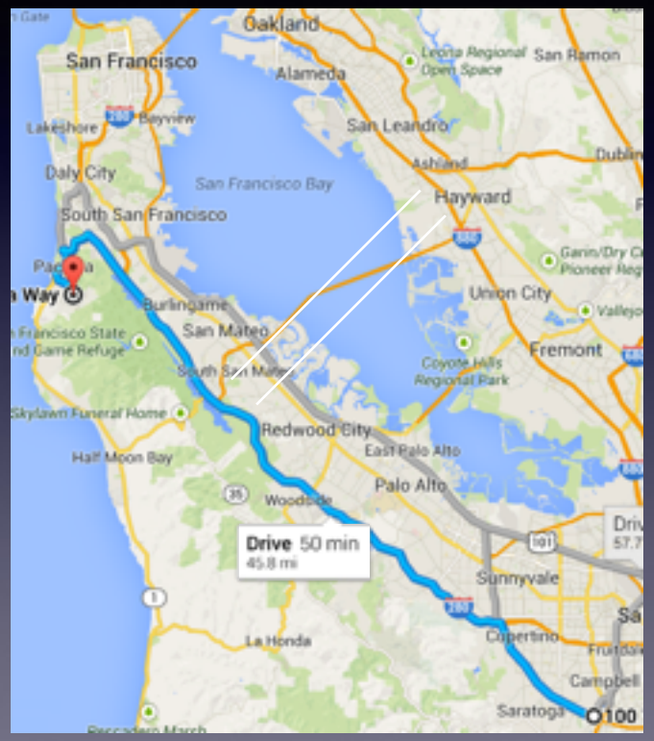
Presenter's notes (what you see here) are meant to complement slide content to allow this slide deck to be read independently of the original presentation at FlowCon (November 1st, 2013). That said, you're likely going to get more value out of watching the presentation on YouTube.

A Word About Me

- About 20 years in technology
- Systems engineering, networking, software development, QA, release management
- Time at Netflix: 1586 days (4y:4m:3d)
- Before at Netflix: Service Delivery in the IT/Ops, troubleshooter, Builder of Python Things[tm]
- Current role: Cloud Monitoring
  - We build platforms
  - Sometimes we make them easy to use

Jack of all trades (with corresponding mastery of none); big Ops background. One of the handfull or two of people who've moved from IT/Ops to Engineering at Netflix.

My Commute



This is my commute — about 45 miles each way (90 miles every day)

What Is a Thing For?

- **Hyundai Elantra 2011**
- **1.8L, 148 HP**
- **32 MPG**

http://www.flickr.com/photos/oliverchua/ (CC)

Not a picture of my car (my car is dirtier), but I have the same model and I do in fact get about 32 MPG in combined driving

What Is a Thing For?

- **Maserati Quattroporte 2009**
- **4.2L, 400 HP**
- **12 MPG**

http://www.flickr.com/photos/xrrr/

A much more powerful car, of course

What Is a Thing For?

http://www.flickr.com/photos/xrrr/

http://www.flickr.com/photos/oliverchua/ (CC)

- **Maserati Quattroporte 2009**
- **4.2L, 400 HP**
- **12 MPG**

- **Hyundai Elantra 2011**
- **1.8L, 148 HP**
- **32 MPG**

The reasons why you might pick one car would not be the reasons why you'd pick the other — I have an acquaintance who actually owns a Maserati, and I've heard him complain about its fuel efficiency. This is wrong — the Maserati is not built for fuel efficiency.

So to figure out whether something satisfies its purpose, you need to understand its purpose. What is that thing for?

What's Important?

1. **Speed of innovation**
2. **Availability**
3. **Cost**

At Netflix, our first priority is speed of innovation. Availability is important to us, but not our first priority. Cost is a distant third priority.

Design For What's Important

Freedom and Responsibility
Hire Smart Experienced People
Set them Loose
Watch Magic Happen

The way we optimize for speed of innovation is to create a culture focusing on freedom and responsibility — we drop smart, experienced people into this culture, provide them with ample context, and let them make the best decisions they can, staying out of their way when they implement.
(We sometimes say that the job of a manager at Netflix is recruit and retain world-class engineers and provide them with context. That's it.)

Operations: Let's Not

Deploy Engineering's Code
- **Separation of Duties: Not a Thing**

Manage by Runbook
- **Externalizes Cost**
- **Makes You Stupid**

Stop People From Making Mistakes
- **Creates Conflict**
- **Works Around the Problem**
- **Stops People From Making the Right Call**
- **Lowers Judgment Bar**

http://www.flickr.com/photos/dw/

Separation of Duties is not actually specified as a SOX requirement; it's mostly done by people because other people do it.

Managing by runbooks allows your developers to deploy terrible code by having other people bear the cost of managing/operating their applications.

Hence cost externalization.

If you put smart people in dumb organizations, you get dumb organizations with more dumb people, not smart organizations.

Operations: Let's

Help Educate Developers
To Make Better Decisions
Help Recover Faster
TTD
TTR

http://www.flickr.com/photos/graeme_pow/

In order for Operations to actually help support our first priority — speed of innovation — what they need to do is help educate developers to make better decisions (more in a moment), and help us recover faster (more so than helping us avoid failure / outages) — the stuff that can be measured by Time To Detect (TTD) and Time To Resolve (TTR) metrics.

Operations: Let's

Best practices and Availability Patterns

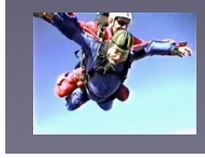
http://www.flickr.com/photos/livn2do/

CORE (Cloud Operations and Reliability Engineering) played a pivotal role in our active/active effort to support US customers from both us-east-1 and us-west-2 datacenters by working with developer groups to figure out what guidelines we needed to have in place for failing between regions, and developing some of the automation around failover.

They also work with developer groups to help drive Latency Monkey and Chaos Monkey adoption, as well as other resiliency patterns.

Operations: Let's

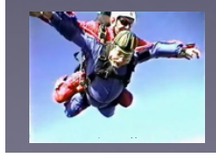
Build Better Alerts

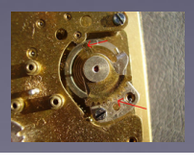


We have an extremely powerful alerting system with the ability to define fine-grained, complex, alerting conditions. While much of that power is available to the Netflix layperson via GUIs, ultimate power requires great understanding of our alerting syntax. CORE engineers are probably in the top 10% of Netflix engineers in their knowledge of our alerting capabilities and their ability to extract value out of the alerting system.

Operations: Let's

sps,nf.cluster,(nccp-modern*|nccp-legacy*),:re,nccprt,
(,NCCPLicense,com_netflix_streaming_nccp_request_license,),:in,:and,stat,SuccessfulRequests,:eq,:and,device.rollup,ps3,:eq,:and,:sum,:set,entering_trough,sps,:get,1h,:offset,
0.95,:mul,sps,:get,:gt,:set,smoothed,sps,:get,
10,0.1,0.02,:des,:set,low_volume,smoothed,:get,-0.005,:mul,
0.1,:add,:set,mid_volume,smoothed,:get,-0.00125,:mul,0.1,:add,:set,base,0.06,:set,min_pct,
1,smoothed,:get,20,:lt,low_volume,:get,:mul,smoothed,:get,
80,:lt,mid_volume,:get,:mul,:add,entering_trough,:get,0.05,:mul,:add,base,:get,:add,:sub,
10,0.1,0.02,:des,:set,sps,:get,min_pct,:get,smoothed,:get,:mul,:lt,5,:rolling-count,2,:ge,$
(device.rollup),:legend



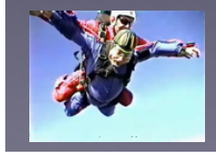


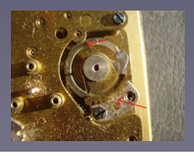
This is an example of one alert configuration for graphing; our CORE people are some of the best SMEs we have for our in-house alerting platform.

Operations: Let's

Tool Builders

Chronos



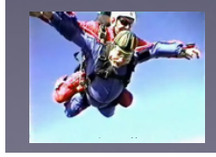


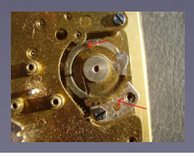
Chronos displays a timeline of changes in production, submitted automatically by the tools that implement these changes. See http://vimeo.com/68183623 for my presentation on this tool (and why you should build or buy something like it).

Operations: Let's

Tool Builders

Central
Alert
Gateway



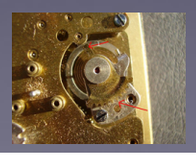


The Central Alert Gateway (CAG) has now been moved over to Monitoring Engineering, but for the first two years of its life it was written and maintained by CORE. CAG accepts events via REST and, depending on event characteristics, forwards them to PagerDuty (for major severity issues), SES (for minor severity issues), or suppresses them. We wrote it because while PagerDuty had a great ability to deduplicate event API calls (based on a given key), it had no ability to support lower-severity events for email-only notification, so we had to write our own. CAG has supported an event volume upward of 400K events per day.

Operations: Let's

Tool Builders

Chaos Gorilla

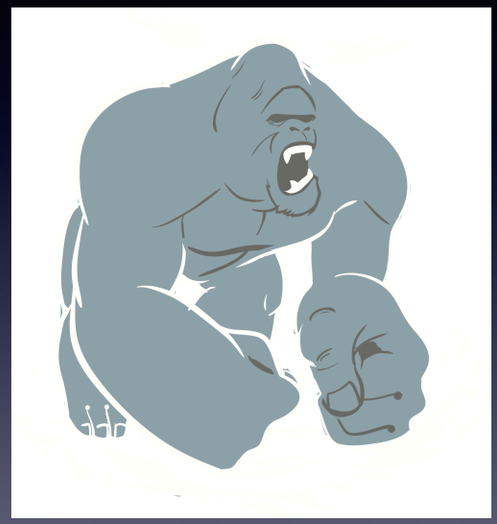



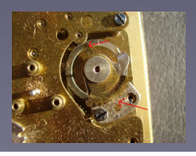
Chaos Gorilla allows us to simulate Amazon Availability Zone failures; we run a Chaos Gorilla exercise at least once a quarter.

Operations: Let's

Tool Builders

Chaos Kong



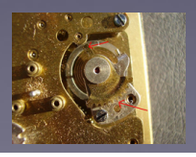




Now that we're running actively in two regions for serving most of the Americas, we've written Chaos Kong to allow us simulate a regional failure.

Operations: Let's

Firefighters

Low-level coordination

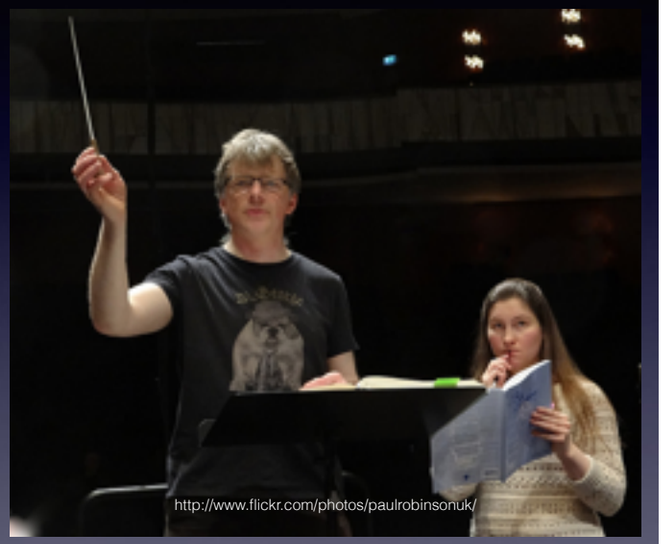





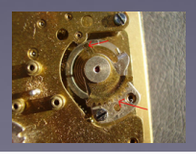
Typically, a more junior member of CORE does this sort of stuff — launching the conference call, potentially paging people, taking notes.
It's important to note, BTW, that anyone at Netflix can always page anyone on-call — it was one of the first value-add apps we built on top of our PagerDuty schedules.

Operations: Let's

Firefighters



Engineering Crisis Manager



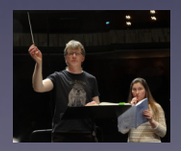

A second CORE person, typically a more seasoned Site Reliability Engineer (SRE) plays an Engineering Crisis Manager (ECM) role. This role is responsible not for doing any hands-on work, but for helping engineers coordinate efforts and making sure our efforts are aimed at recovering service, rather than at discovering root cause. The ECM partners with the person doing low-level coordination — it's a two-person team responsible for dealing with the crisis, at two different levels.

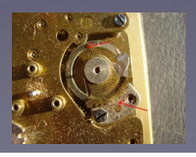
We used to have ECM duty rotate among all managers/directors in Engineering, but that resulted in a duty cycle that was on the order of about 18 weeks, and it turned out that having someone only be on-call every 18 weeks resulted in them being extremely rusty when it was their turn. We try to aim for a duty cycle of no more than about six weeks now (and, given that the ECM is involved only in major severity events, tend to see ECM duty be relatively light on an average week)

Operations: Let's

Firefighters

Incident Reviews
(Easier Than You'd Expect)

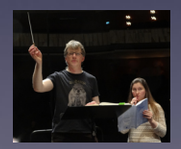


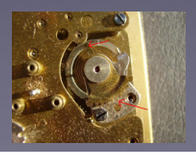


We do a pretty decent job keeping our Incident Reviews (IRs) blameless and factual. Typically, the challenge CORE people who run IRs have is that we'll have a developer come into the room and spend a few minutes describing everything they might have done wrong and every possible mistake they may have made, and conclude with "well, that's why we had that problem," whereas we tend to believe strongly that human error is never an acceptable root cause in our business (John Allspaw, BTW, said it before I did — I was inspired by his Velocity/2011 presentation on post mortems), and need to shift the conversation from the developer to the technology/tools failure.

So What Does It Take?

- Grace Under Pressure
- Great Technical Skills
- A Passion for Making Things Better
- A Thirst for Persuasion






tl;dr

- Stay Out of the Way
- Help Others Stay Out of Their Way
- Improve Visibility
- Recover Faster