Migrating to Microservices

Adrian Cockcroft @adrianco

QCon London – 6th March 2014

What I learned from my time at Netflix

- Speed wins in the marketplace
- Remove friction from product development
- High trust, low process
- Freedom and responsibility culture
- Don’t do your own undifferentiated heavy lifting
- Simple patterns automated by tooling
- Microservices for speed and availability

adrian cockcroft @adrianco 10 Apr

Baffling-late-adopters as a Service

Retweeted by Andrew Clay Shafer

Expand

Typical reactions to my Netflix talks…

BV

Demands on IT Increased 1000x

Compete or lose in the market!

Colonel Boyd USAF, on Combat

"Get inside your adversaries' OODA loop to disorient them"

Observe

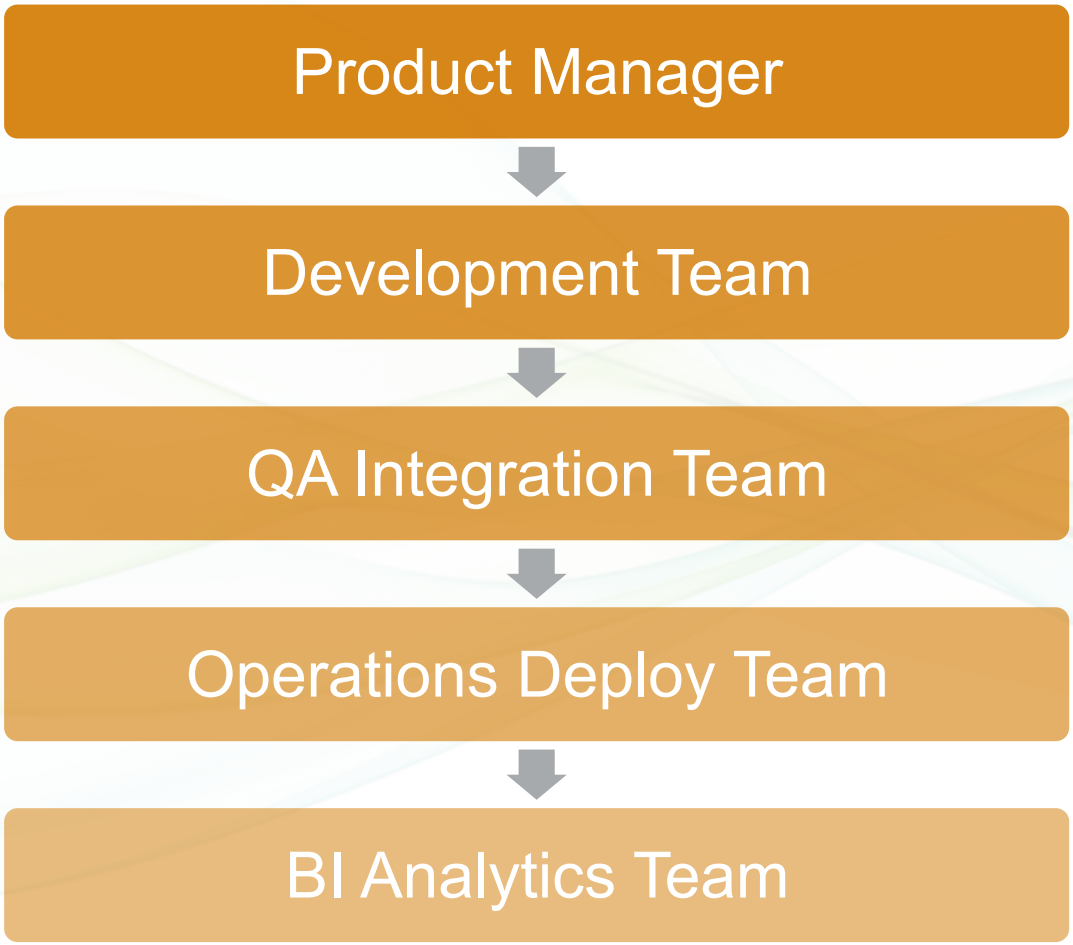
Orient

Decide

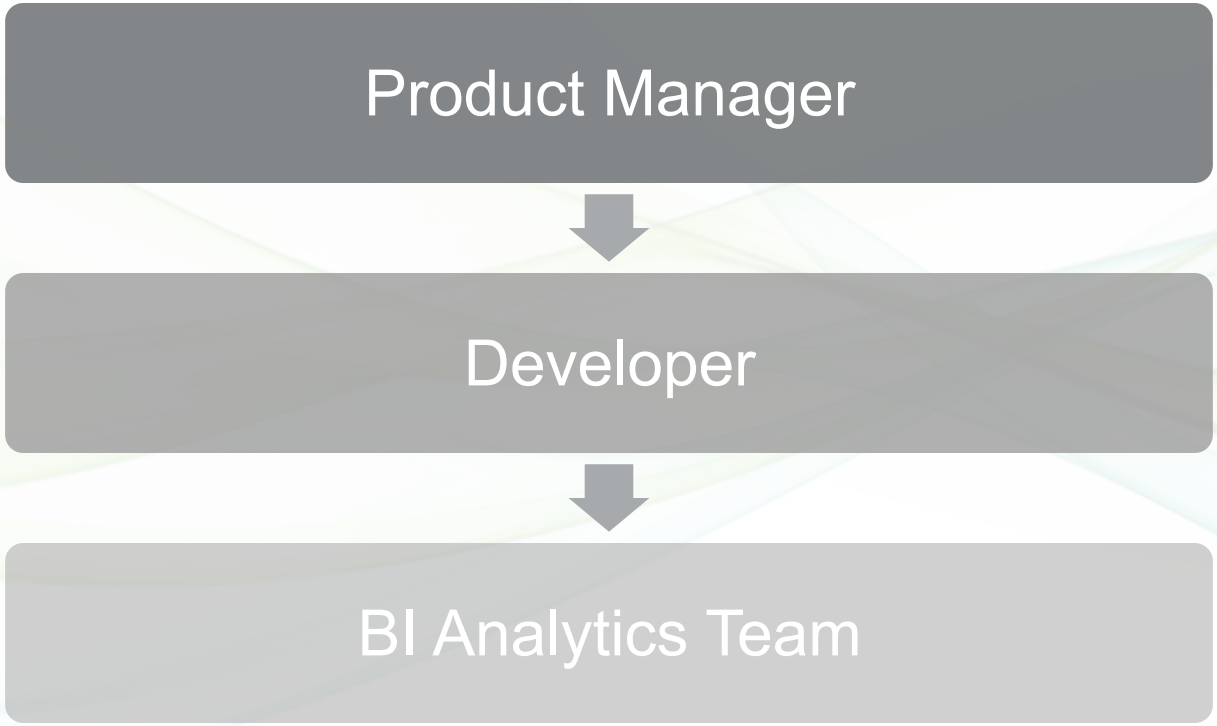
Act

How fast can you act?

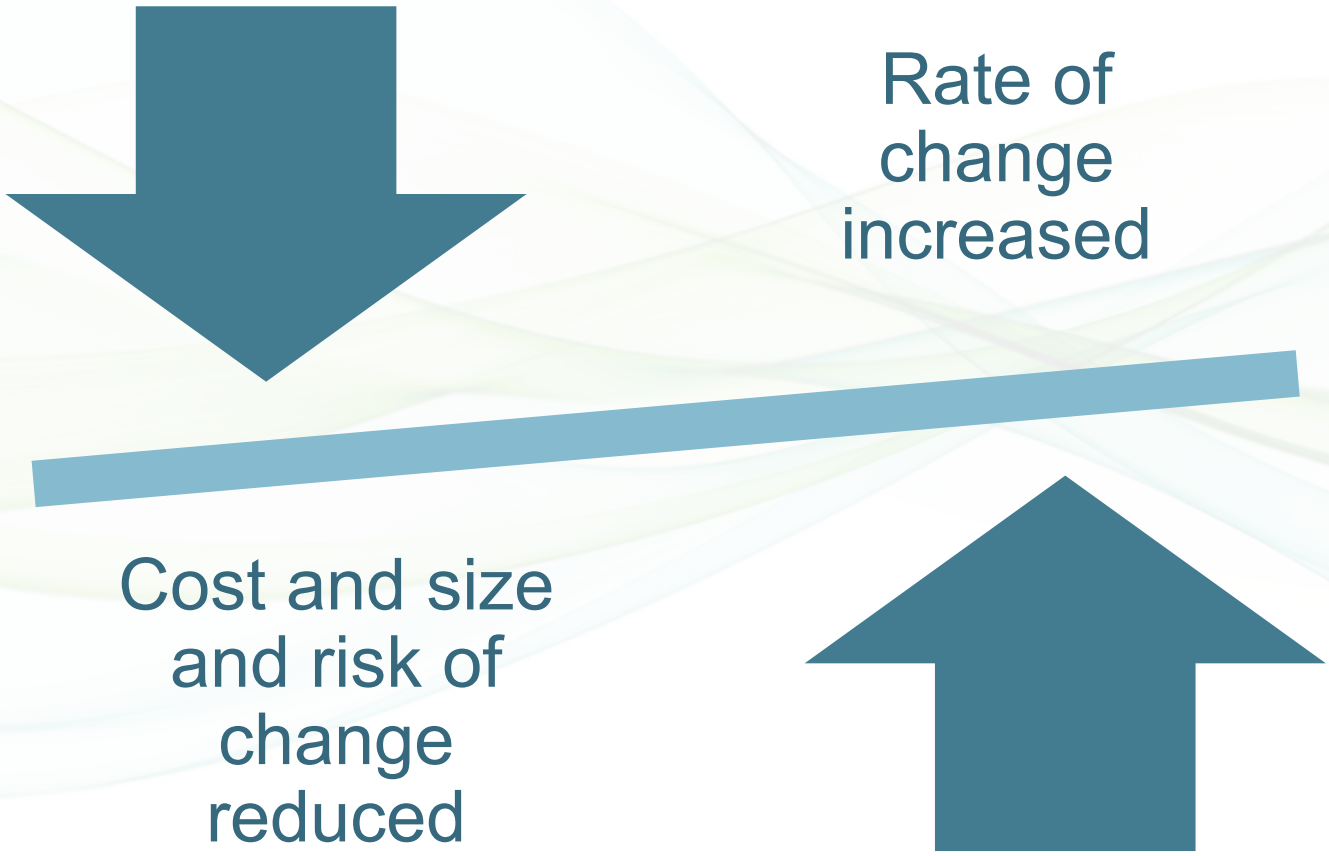
Process Hand-Off Steps for Product Development on IaaS

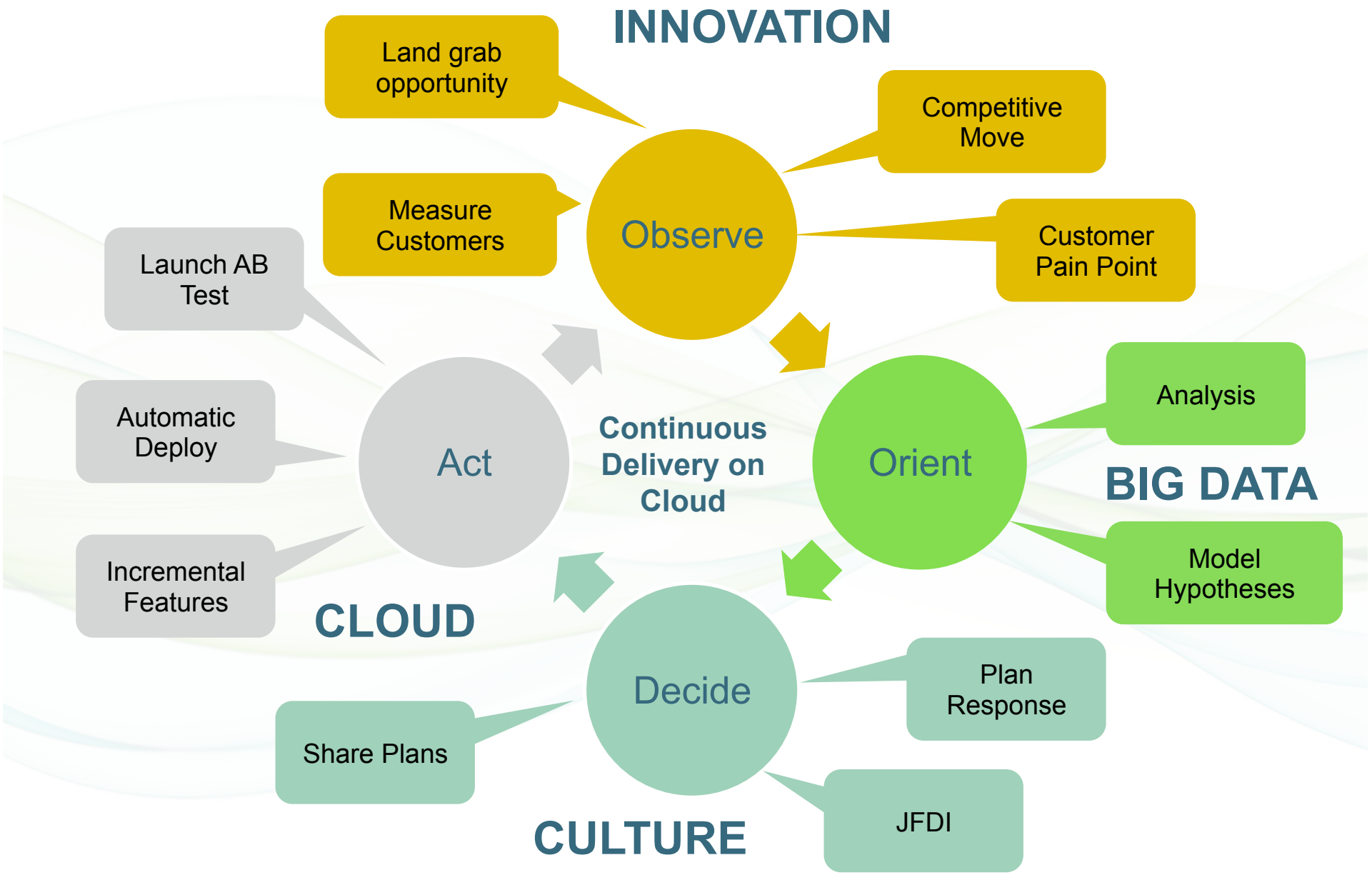


Process Hand-Off Steps for Feature Development on PaaS



What Happened?




INNOVATION
Land grab opportunity
Competitive Move
Measure Customers
Observe
Customer Pain Point
Launch AB Test
Automatic Deploy
Act
Continuous Delivery on Cloud
Orient
Analysis
BIG DATA
Incremental Features
Model Hypotheses
CLOUD
Decide
Plan Response
Share Plans
JFDI
CULTURE

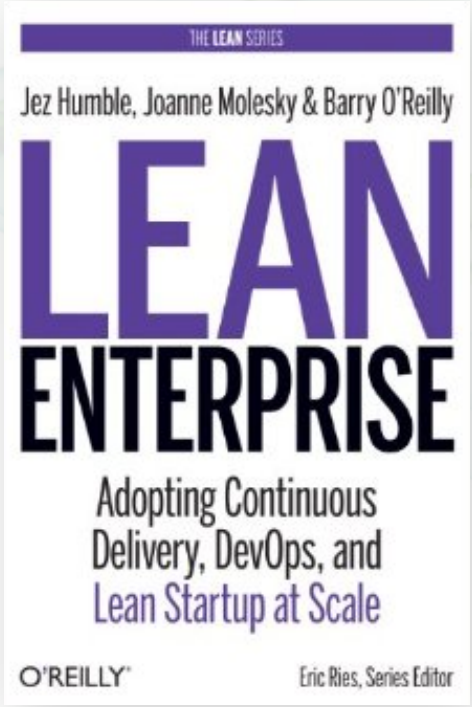
OK, how do I get there?

"This is the IT swamp draining manual for anyone who is neck deep in alligators."

Adrian Cockcroft, Cloud Architect at Netflix

Continuous Deployment for Speed

- There is no time for handoffs between teams
- IT is a cloud API providing DevOps automation
- "Run what you wrote" – root access *and* Pagerduty
- High trust culture for fast local action
- Freedom and responsibility for developers
- Lean Enterprise – coming May 2014



BV

Open Source Ecosystems

- The most advanced, scalable and stable code is OSS
- No procurement cycle, fix and extend it yourself
- Github is your company's online resume
- Extensible platforms create ecosystems
- Give up control to get ubiquity – Apache license
- Don't miss Simon Wardley's Cloud Expo and QCon talks!

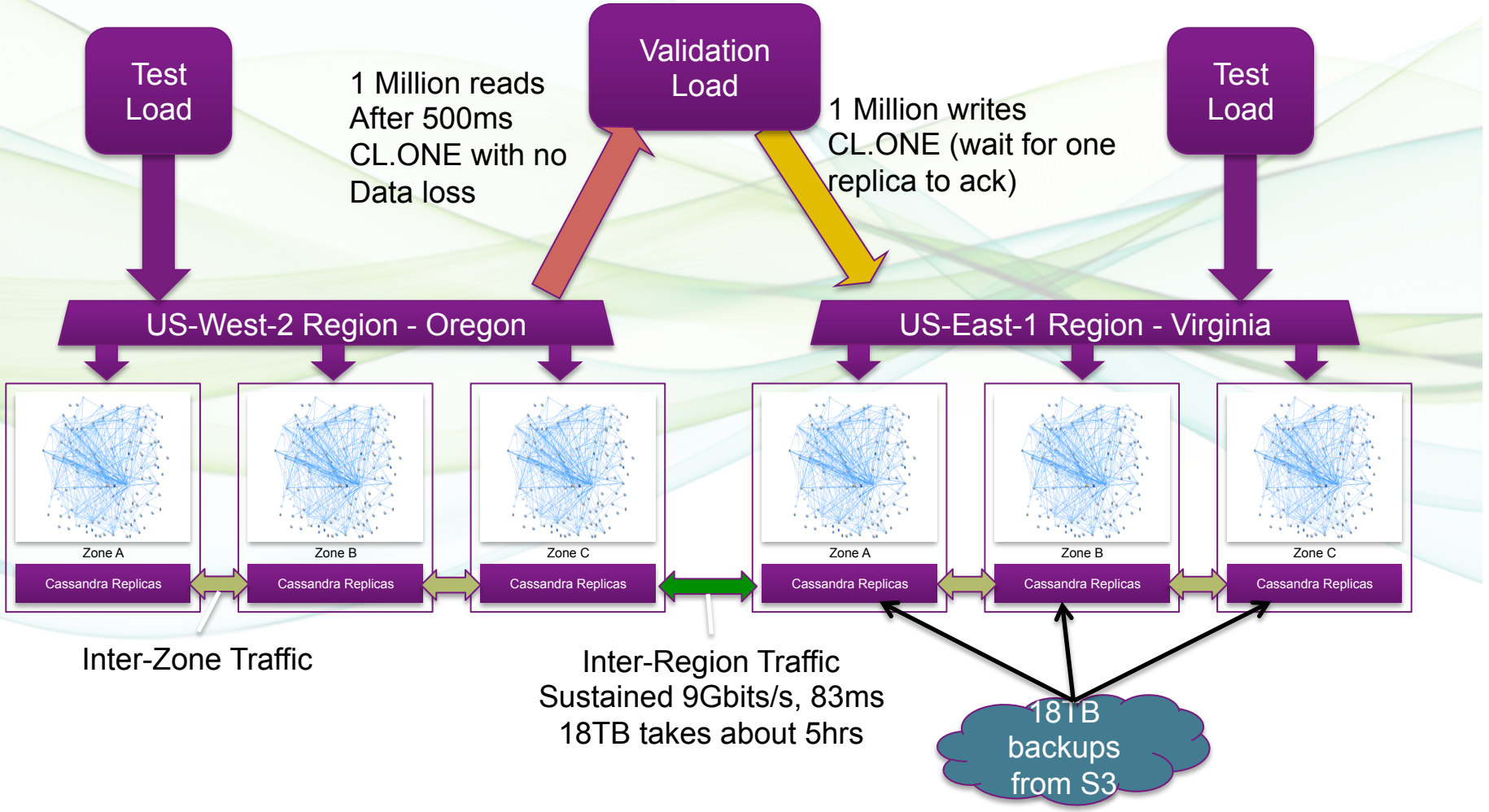
Innovate, Leverage and Commoditize

Cloud Native for High Availability

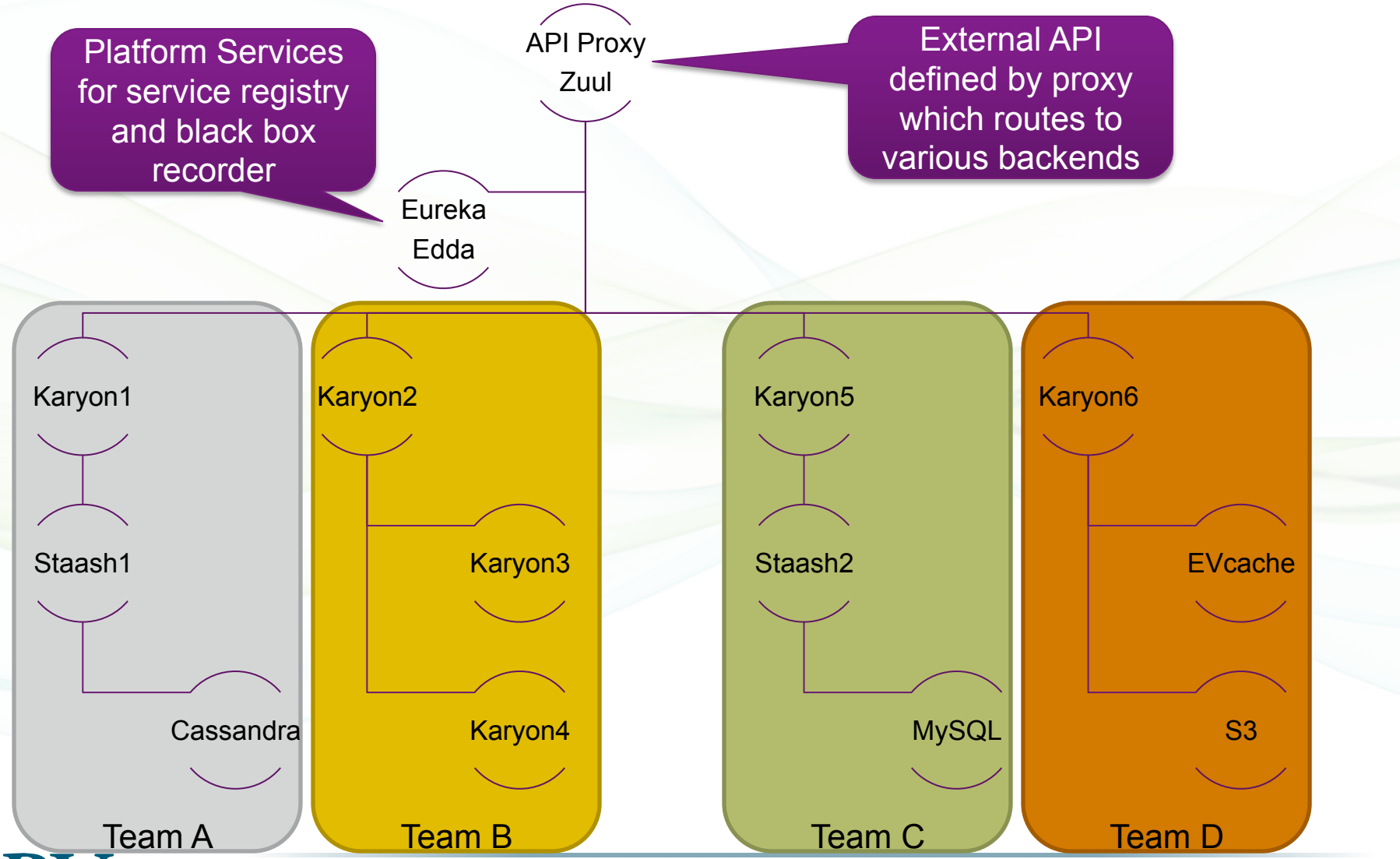
- Business logic isolation in stateless micro-services
- Immutable code with instant rollback
- Auto-scaled capacity and deployment updates
- Distributed across availability zones and regions
- De-normalized single function NoSQL data stores
- NetflixOSS at netflix.github.com and techblog.netflix.com
- Details from Netflix team at Qcon London March 4-8 2014

Cloud Native Benchmarking

Write intensive test of cross region replication capacity
16 x hi1.4xlarge SSD nodes per zone = 96 total
192 TB of SSD in six locations up and running Cassandra in 20 minutes



NetflixOSS Style Microservices Deployment

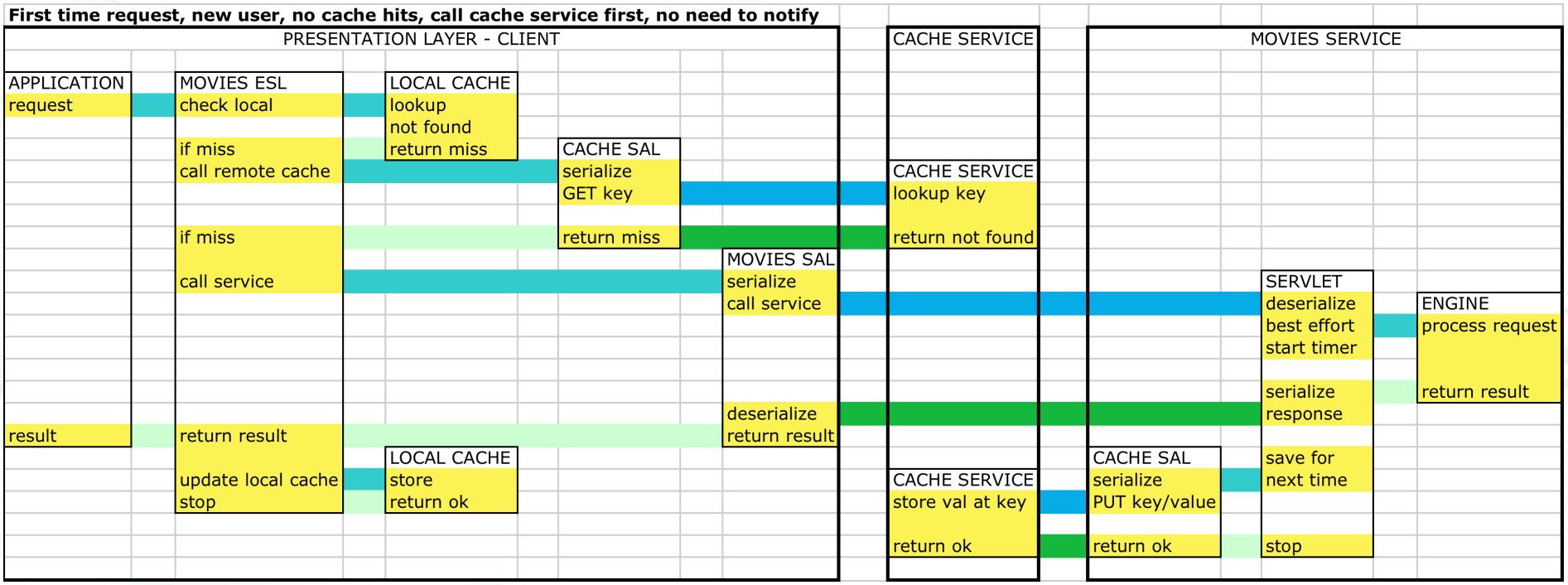


Separate Concerns Using Micro-services

- Inverse Conway's Law – teams own service groups
- One "verb" per single function micro-service
- Size doesn't matter
- One developer independently produces a micro-service
- Each micro-service is it's own build, avoids trunk conflicts
- Stateless business logic
- Stateful cached data access layer

Micro-service Interaction Swimlane Diagram

Two Karyon based services keeping state in an EVcache



Microservices Development Architecture

- Versioning
  - Leave multiple old microservice versions running
  - Fast introduction vs. slow retirement asymmetry
- Client libraries
  - Even if you start with a protocol, a client side driver is the end-state
  - Best strategy is to own your own client libraries from the start
- Multithreading and Non-blocking Calls
  - Reactive model RxJava using Observable to hide threading
  - Try migration from Tomcat to Netty to get non-blocking I/O speedup
- Enterprise Service Bus / Messaging
  - Message buses are CP with big problems getting to AP
  - Use for send and forget over high latency links

Microservice APIs

- API Patterns
  - RPC, REST, Self-describing overhead, public vs. in-house
  - XPATH, jsonpath adds some flexibility but not as useful in-house
- Scripted API Endpoints - Dynamic Client RPC Pattern
  - See Daniel Jacobson's talks at slideshare.net/netflix
  - March 3rd 2014 techblog.netflix.com post by Sangeeta Narayanan
- Service discovery
  - Build time Ivy, Gradle and Artifactory
  - Run time Zookeeper for CP, Eureka for AP
- Understanding existing code boundaries
  - Structure 101 – buy a bigger printer and wallpaper a room

Microservice Datastores

- Book: Refactoring Databases
  - SchemaSpy to examine schema structure
  - Denormalization into one datasource per table or materialized view
- CAP – Consistent or Available when Partitioned
  - Look at Jepsen models for common systems aphyr.com/tags/jepsen
  - AP as default for distributed system unless downtime is explicitly OK
- Circuit Breakers – See Fluxcapacitor.com for code examples
  - NetflixOSS Hystrix, Turbine, Latency Monkey, Ribbon/Karyon
  - Also look at Finagle/Zipkin from Twitter and Metrics, Graphite
  - Speed of development vs. scale driving resilience
- Microservice lifecycle
  - Mature slow changing, new fast changing
  - Number increase over time, services increase in size then split

BV

Micro-services Bring-Up Strategy Simplest and Soonest

Strategies for impatient product managers

- Carrot

  “This new feature you want will be ready faster as a microservice”

- Stick

  “This new feature you want will only be implemented in the new microservice based system”

- Shiny Object

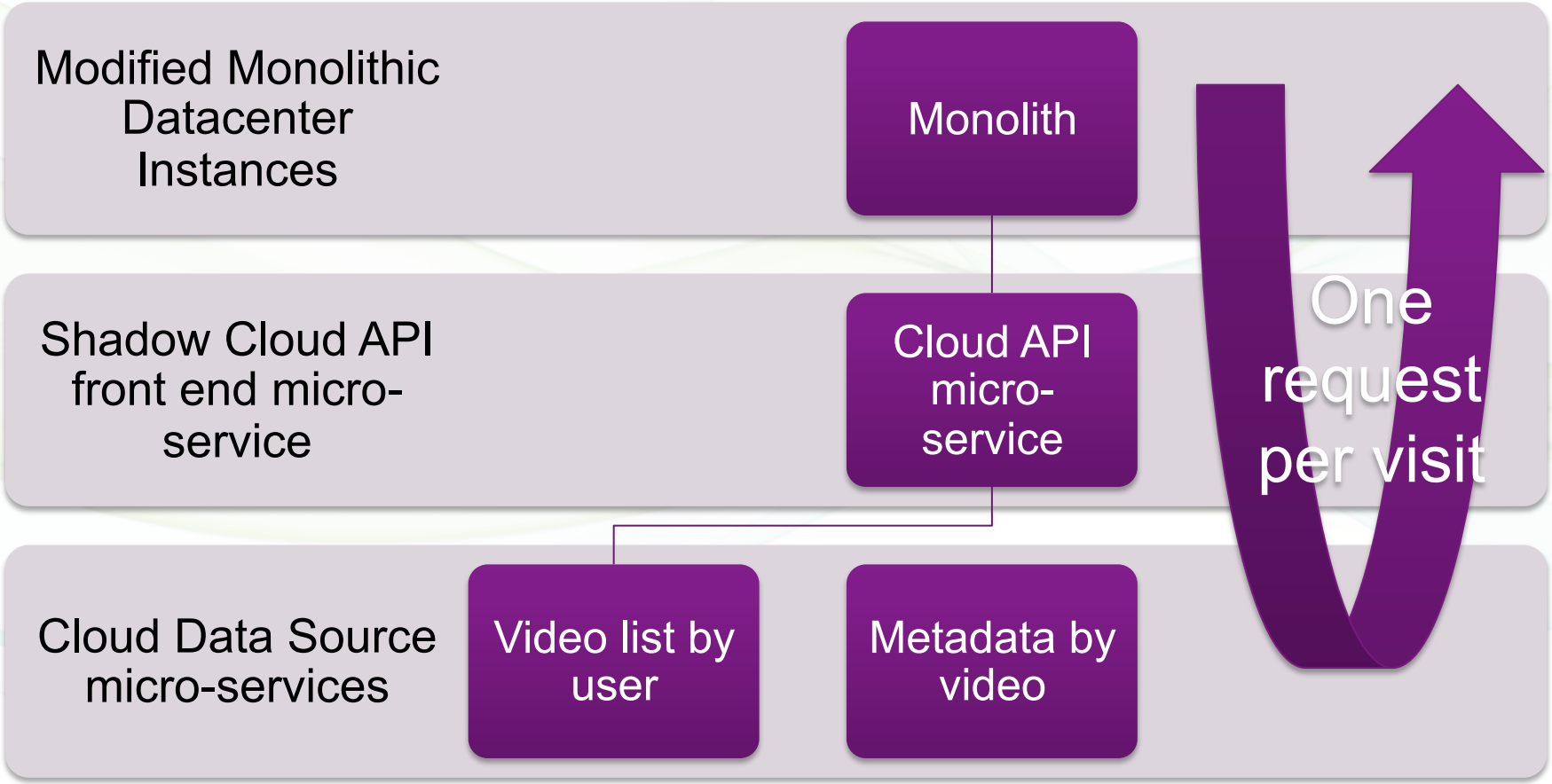
  “Why don’t you concentrate on some other part of the system while we get the transition done?”

Shadow Traffic Backend Redirection

- First attempt to send traffic to cloud based microservice
  - Real traffic stream to validate cloud back end
  - Uncovered lots of process and tools issues
  - Uncovered Service latency issues
- Modified Monolithic Datacenter Code Path
  - Returns Genre/movie list for a customer
  - Asynchronously duplicates request to cloud
  - Start with send-and-forget mode, ignore response
- Dynamic Consistent Traffic Percentage
  - If (customerid % 100 < threshold) shadow_call()

BV

Shadow Redirect Pattern



Metadata Shim Micro-service

- Metadata server isolates new platform from old codebase
  - Isolate/unblock cloud team from metadata team schedule
  - Monolithic code only supports obsolete movie object
- VMS subsets the metadata
  - Only load data used by cloud micro-services
  - Fast bulk loads for VMS clients speed startup times
- VMS pre-processes the metadata
  - Explore next generation metadata cache architecture
  - Distribute metadata to micro-services using S3 or memcached

Microservices Deployment

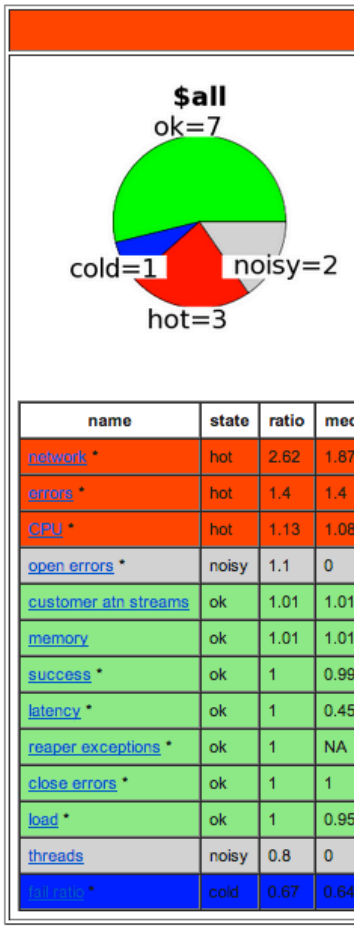
- Deployment tooling
  - Vagrant for small services on machine testing
  - Cloud based Asgard tag/developer routing
  - Dependencies described with CFengine promises or Puppet
  - Coordinated deployments with Fabric or CloudFormation
- Production updates and Immutability
  - Monolithic breaks everything at once
  - Microservice add a new microservice, no impact, route test traffic to it
  - Version aware routing, eventual retirement
- Systems vs. Goals and Learning from Failure
  - Conways law, Speed/Agility, A|B test based improvements
  - Scott Adams “How to fail at almost everything and still win big”

Automatic Canary Red/Black Deployment

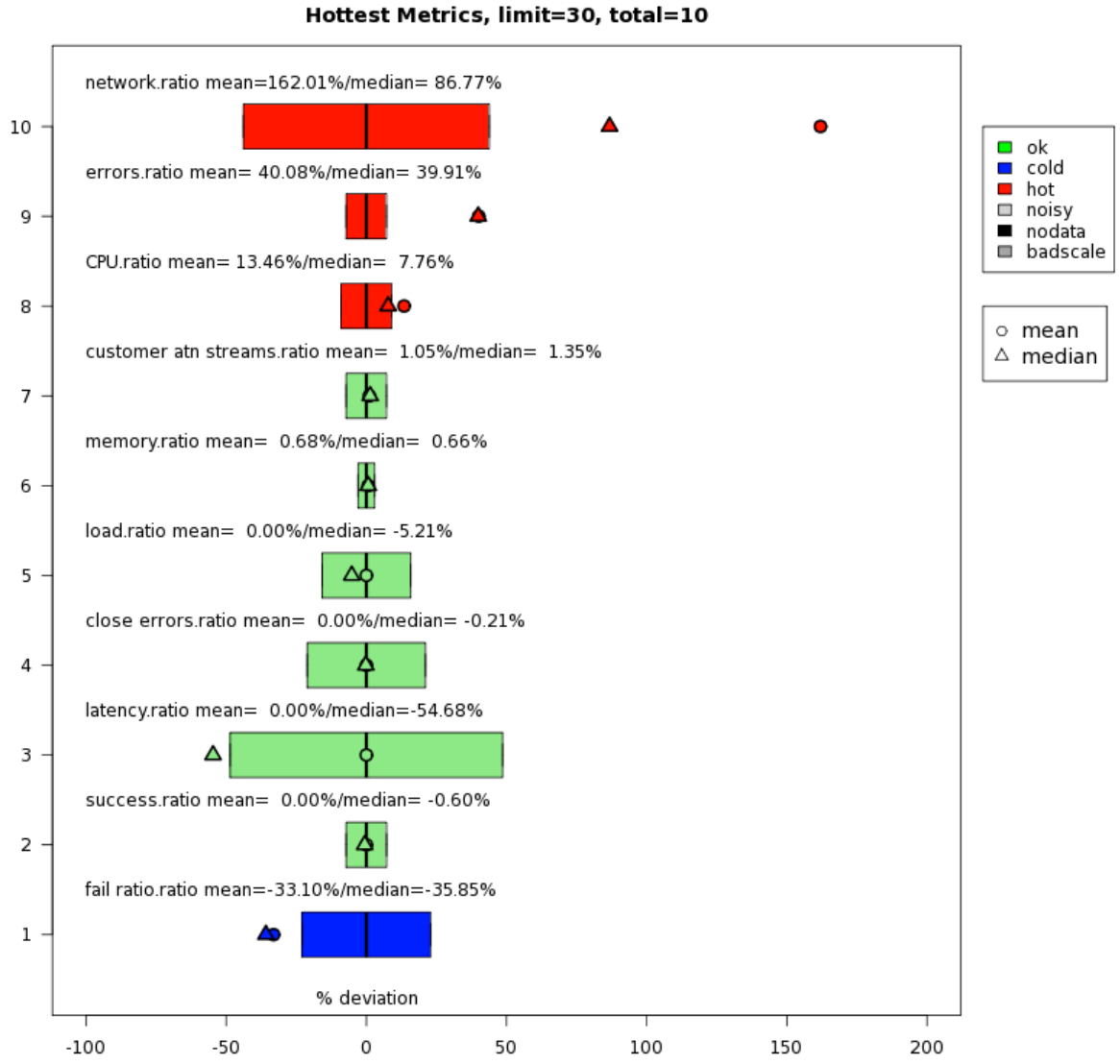
In use at Netflix for tens of large fleet microservices in active development

- Developer checks in code then get email notifications of progress
- Jenkins build launches AMI in test account and starts tests
- If tests pass launch canary signature analysis in production
  - Start one new instance of the old code per zone
  - Start one new instance of the new code per zone
  - Ramp up traffic and analyze metrics on all six
- If canary signature looks good replace current production
  - Scale canary build up to full capacity
  - Send all the traffic to the new code
  - Wait until after peak traffic time then remove old code instances
- Security team tools notice the new build via Edda query
  - Automatic penetration test scan

Netflix Bad Canary Signature

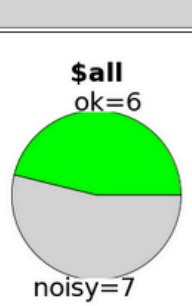


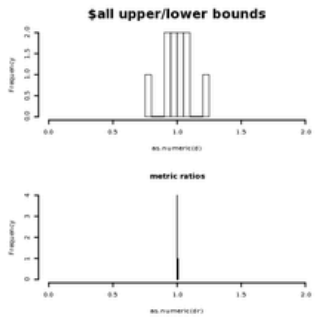
name	state	ratio	mec
network *	hot	2.62	1.87
errors *	hot	1.4	1.4
CPU *	hot	1.13	1.08
open errors *	noisy	1.1	0
customer atn streams	ok	1.01	1.01
memory	ok	1.01	1.01
success *	ok	1	0.99
latency *	ok	1	0.45
reaper exceptions *	ok	1	NA
close errors *	ok	1	1
load *	ok	1	0.95
threads	noisy	0.8	0
fail ratio *	cold	0.67	0.64

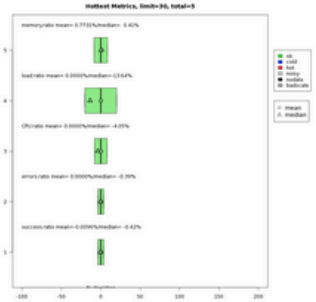


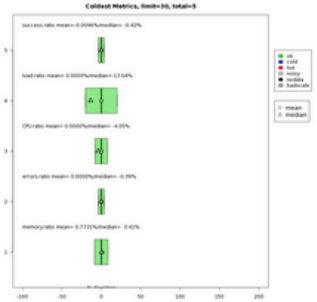
Netflix Happy Canary Signature

$all: state=noisy (98%)





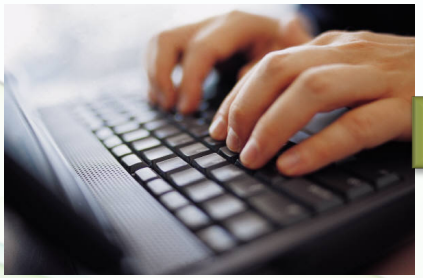




name	state	ratio	median	lower	upper
close errors *	noisy	1.19	0	0	6.13
failure *	noisy	1.18	0	0	6.14
fail ratio *	noisy	1.15	0	0	7.05
memory	ok	1.01	1	0.91	1.09
errors *	ok	1	1	0.96	1.04
latency *	noisy	1	0.59	0.31	1.69
customer atn streams *	ok	1	NA	1	1
CPU *	ok	1	0.96	0.92	1.08
load *	ok	1	0.86	0.8	1.2
network *	noisy	1	0.21	0	2.34
success	ok	1	1	0.96	1.04
threads	noisy	0.83	0.84	0.29	1.71
recovered	noisy	0.75	0	0	2.54

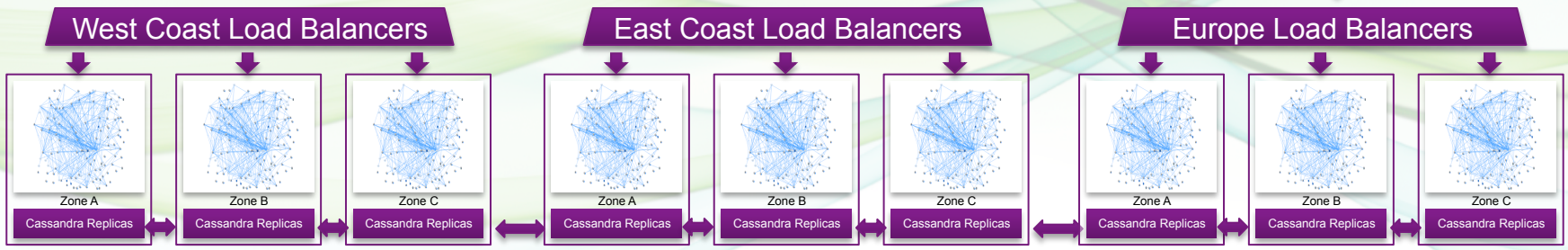
Netflix Global Deploy-to-Prod Automation

Afternoon in California
Code checked in



Night-time in Europe

If passes test suite, canary then deploy



Canary then deploy

Canary then deploy

After peak on West Coast

Next day on East Coast

After peak in Europe

Monitoring Micro-services

Visualizing the request flow

- Appdynamics
  - Instrument the JVM to capture everything including traffic flows
  - Insert tag for every http request with a header annotation guid
  - Visualize the over-all flow or the business transaction flow
- Boundary.com and Lyatiss CloudWeaver
  - Instrument the packet flows across the network
  - Capture the zone and region config from cloud APIs and tags
  - Correlate, aggregate and visualize the traffic flows
- Instrumented PaaS Communication Mechanisms
  - CloudFoundry and Apcera route all traffic through NATS
  - NetflixOSS ribbon client and karyon server http annotation guid
  - Scales beyond capabilities of centralized vendor based tools

Scaling Continuous Delivery Models

Monolithic – Etsy, Facebook

- Etsy – 8 devs per train
- Everyone runs the monolith
- Queue for the next train
- Coordination chat session
- Need to learn deploy process
- Copy code to existing servers
- Few concurrent versions
- 50 monolithic updates/day
- Roll-forward only
- “Done” is released to prod

Microservices – Netflix, Gilt

- Everyone has their own build
- Dev runs their own microservice
- No waiting, no meetings
- API call to update prod timeline
- Automated hands-off deploy
- Immutable code on new servers
- Unlimited concurrent versions
- 100s of independent updates
- Roll-back in seconds
- “Done” is retired from prod

BV

Separation of Concerns

Bounded Contexts

Summary

- Speed wins in the marketplace
- Remove friction from product development
- High trust, low process
- Freedom and responsibility culture
- Don’t do your own undifferentiated heavy lifting
- Simple patterns automated by tooling
- Microservices for speed and availability

Any Questions? Upcoming Presentations by @adrianco

- Battery Ventures http://www.battery.com
- Adrian's Blog http://perfcap.blogspot.com
- Netflix Tech Blog http://techblog.netflix.com
- Netflix Slideshare http://slideshare.com/netflix

- Migrating to Microservices – Qcon London - March 6th, 2014
- Monitorama Keynote Portland OR - May 7th, 2014
- GOTO Chicago Opening Keynote May 20th, 2014
- DevOps Summit at Cloud Expo New York – June 10th, 2014
- GOTO Copenhagen/Aarhus – Denmark – Oct 25th, 2014

BV