Netflix and Open Source

April 2013

Adrian Cockcroft

@adrianco #netflixcloud @NetflixOSS

http://www.linkedin.com/in/adriancockcroft

Cloud Native

NetflixOSS – Cloud Native On-Ramp

Netflix Open Source Cloud Prize

We are Engineers

We solve hard problems

We build amazing and complex things

We fix things when they break

We strive for perfection

Perfect code

Perfect hardware

Perfectly operated

But perfection takes too long…

So we compromise

Time to market vs. Quality

Utopia remains out of reach

Where time to market wins big

Web services

Agile infrastructure - cloud

Continuous deployment

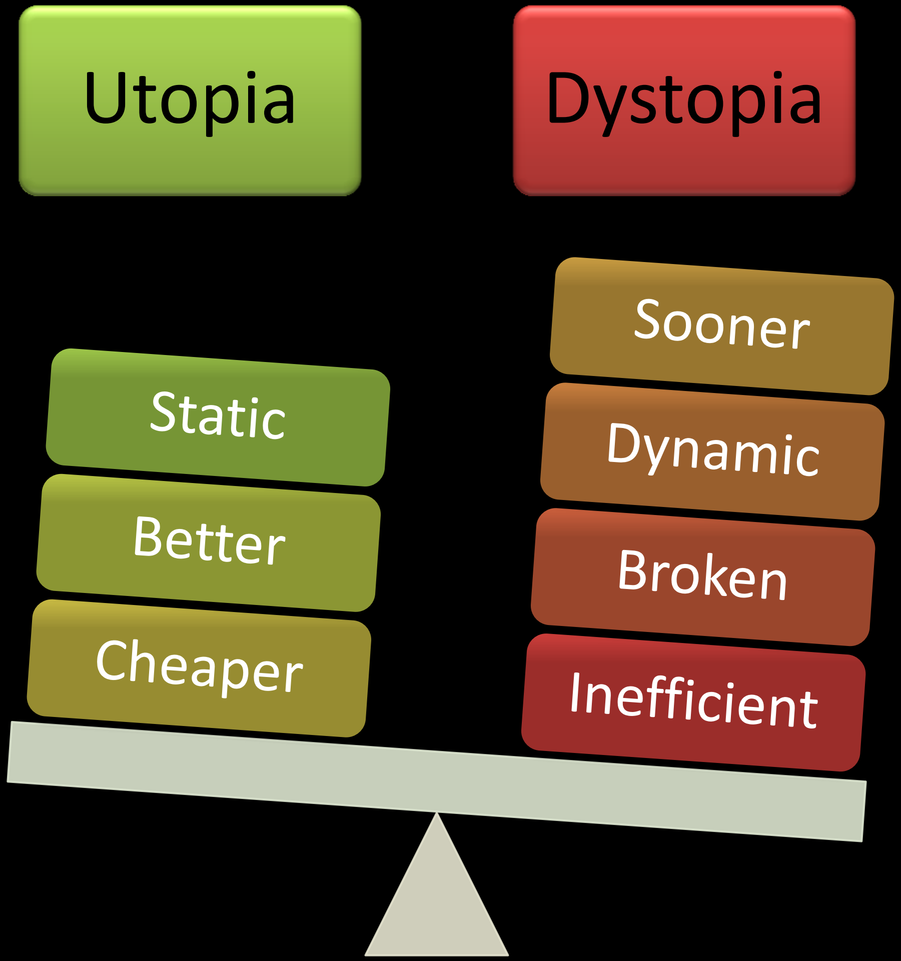
How Soon?

Code features in days instead of months

Hardware in minutes instead of weeks

Incident response in seconds instead of hours

Tipping the Balance



Utopia

Dystopia

Static

Better

Cheaper

Sooner

Dynamic

Broken

Inefficient

A new engineering challenge

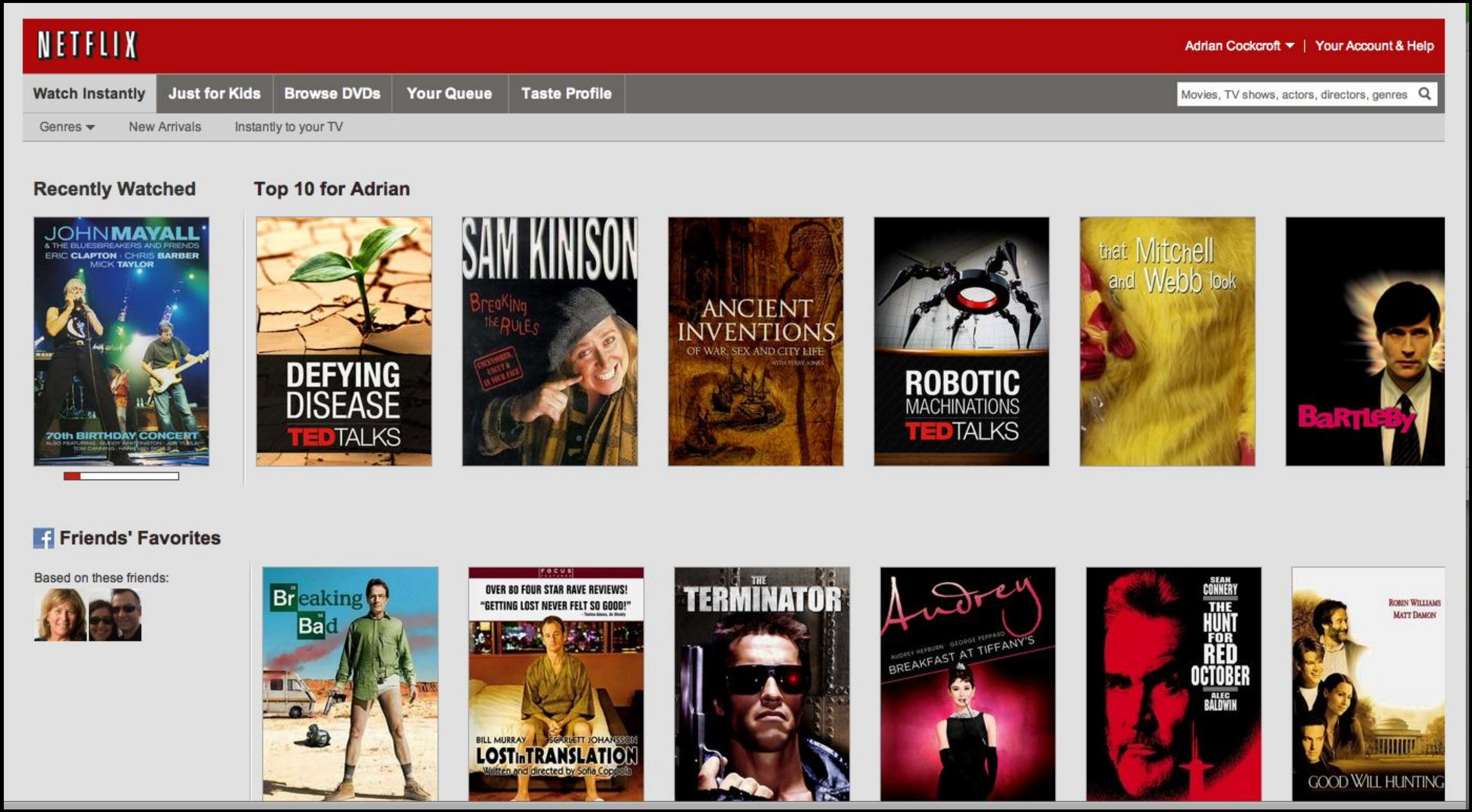
Construct a highly agile and highly available service from ephemeral and often broken components

Cloud Native

How does Netflix work?

Netflix Member Web Site Home Page

Personalization Driven – How Does It Work?



NETFLIX

Adrian Cockcroft ▾ | Your Account & Help

Watch Instantly | Just for Kids | Browse DVDs | Your Queue | Taste Profile

Movies, TV shows, actors, directors, genres

Genres ▾ New Arrivals Instantly to your TV

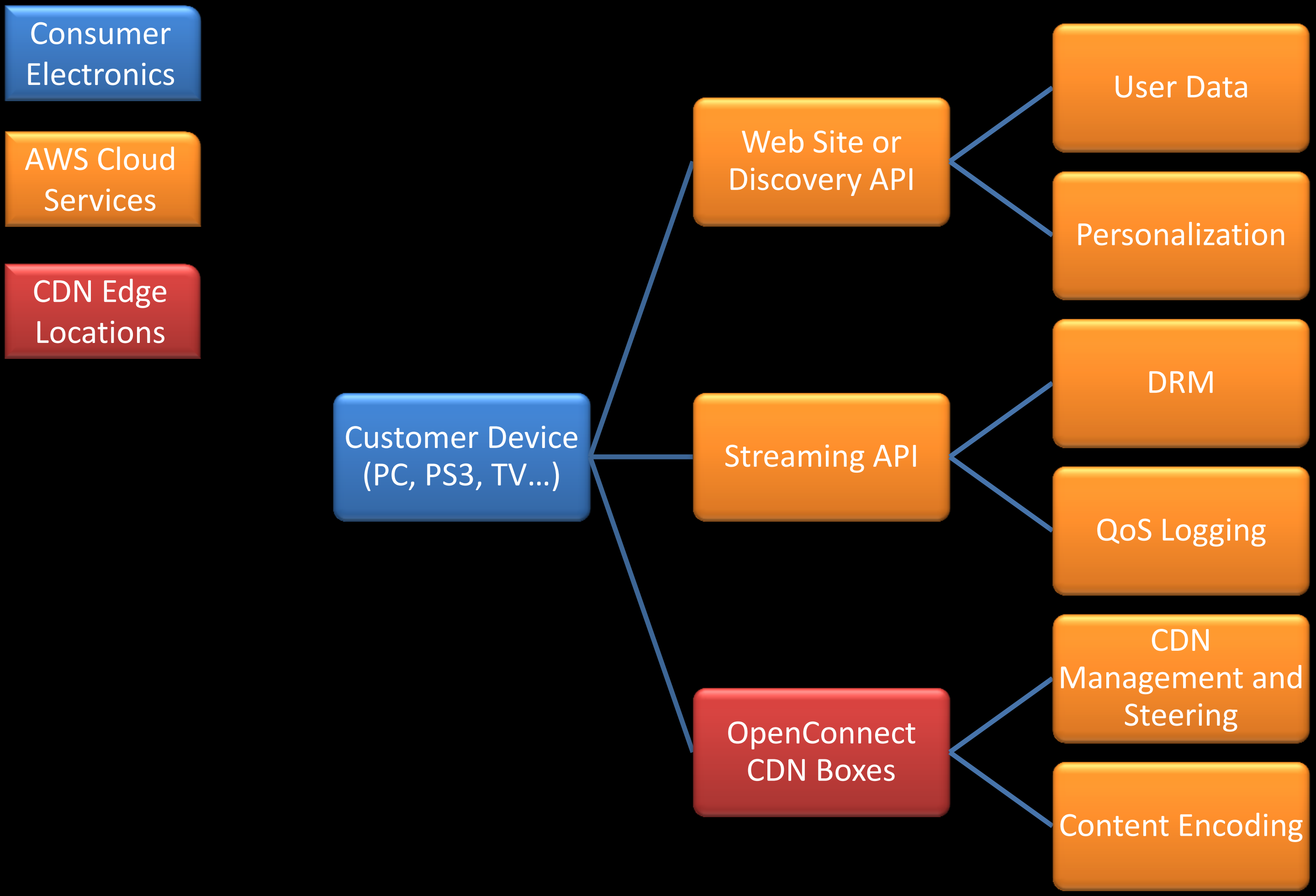
Recently Watched

Top 10 for Adrian

Friends' Favorites

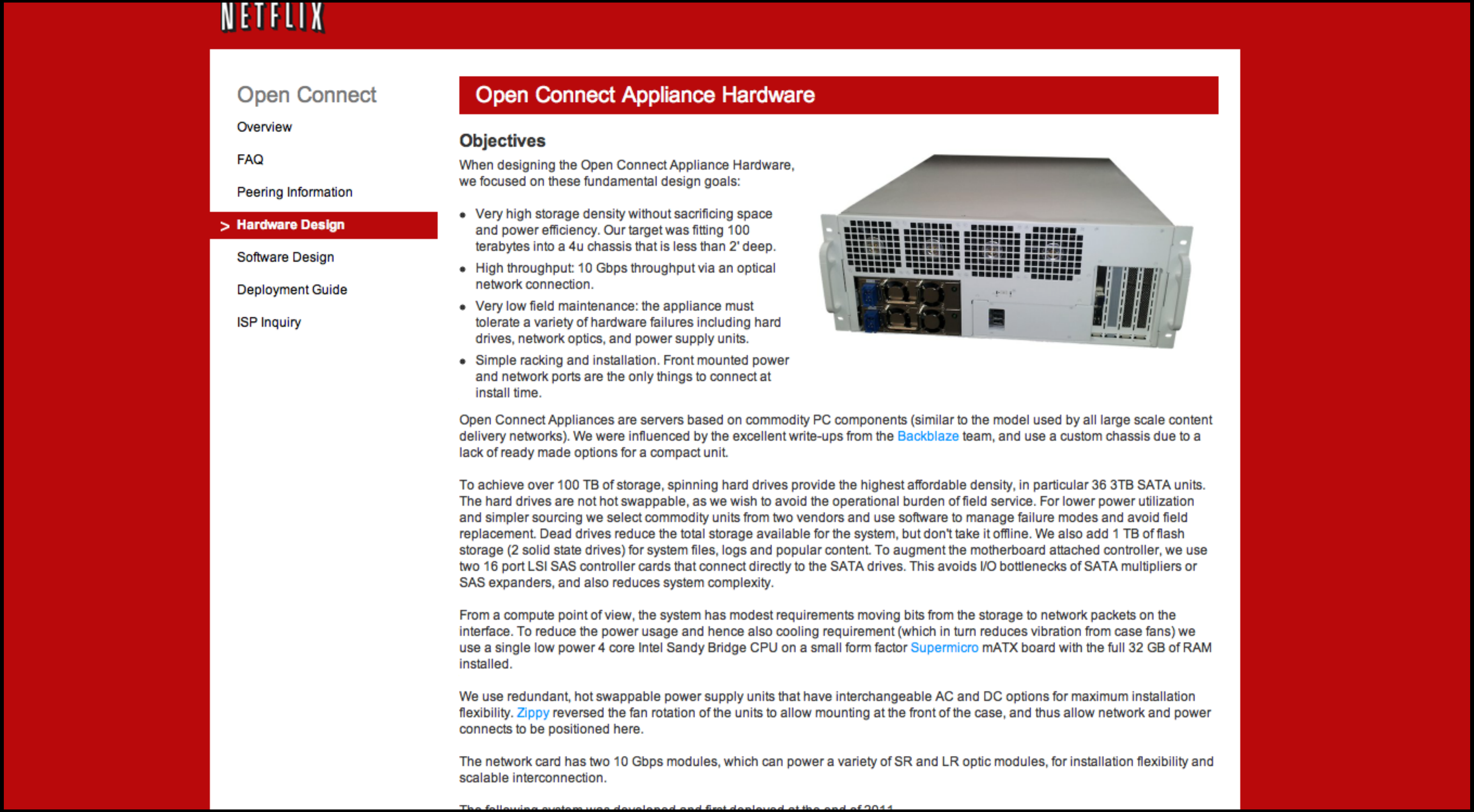
Based on these friends:

How Netflix Streaming Works



Content Delivery Service

Open Source Hardware Design + FreeBSD, bird, nginx



Open Connect Appliance Hardware

Objectives

When designing the Open Connect Appliance Hardware, we focused on these fundamental design goals:

- Very high storage density without sacrificing space and power efficiency. Our target was fitting 100 terabytes into a 4u chassis that is less than 2' deep.
- High throughput: 10 Gbps throughput via an optical network connection.
- Very low field maintenance: the appliance must tolerate a variety of hardware failures including hard drives, network optics, and power supply units.
- Simple racking and installation. Front mounted power and network ports are the only things to connect at install time.

Open Connect Appliances are servers based on commodity PC components (similar to the model used by all large scale content delivery networks). We were influenced by the excellent write-ups from the Backblaze team, and use a custom chassis due to a lack of ready made options for a compact unit.

To achieve over 100 TB of storage, spinning hard drives provide the highest affordable density, in particular 36 3TB SATA units. The hard drives are not hot swappable, as we wish to avoid the operational burden of field service. For lower power utilization and simpler sourcing we select commodity units from two vendors and use software to manage failure modes and avoid field replacement. Dead drives reduce the total storage available for the system, but don't take it offline. We also add 1 TB of flash storage (2 solid state drives) for system files, logs and popular content. To augment the motherboard attached controller, we use two 16 port LSI SAS controller cards that connect directly to the SATA drives. This avoids I/O bottlenecks of SATA multipliers or SAS expanders, and also reduces system complexity.

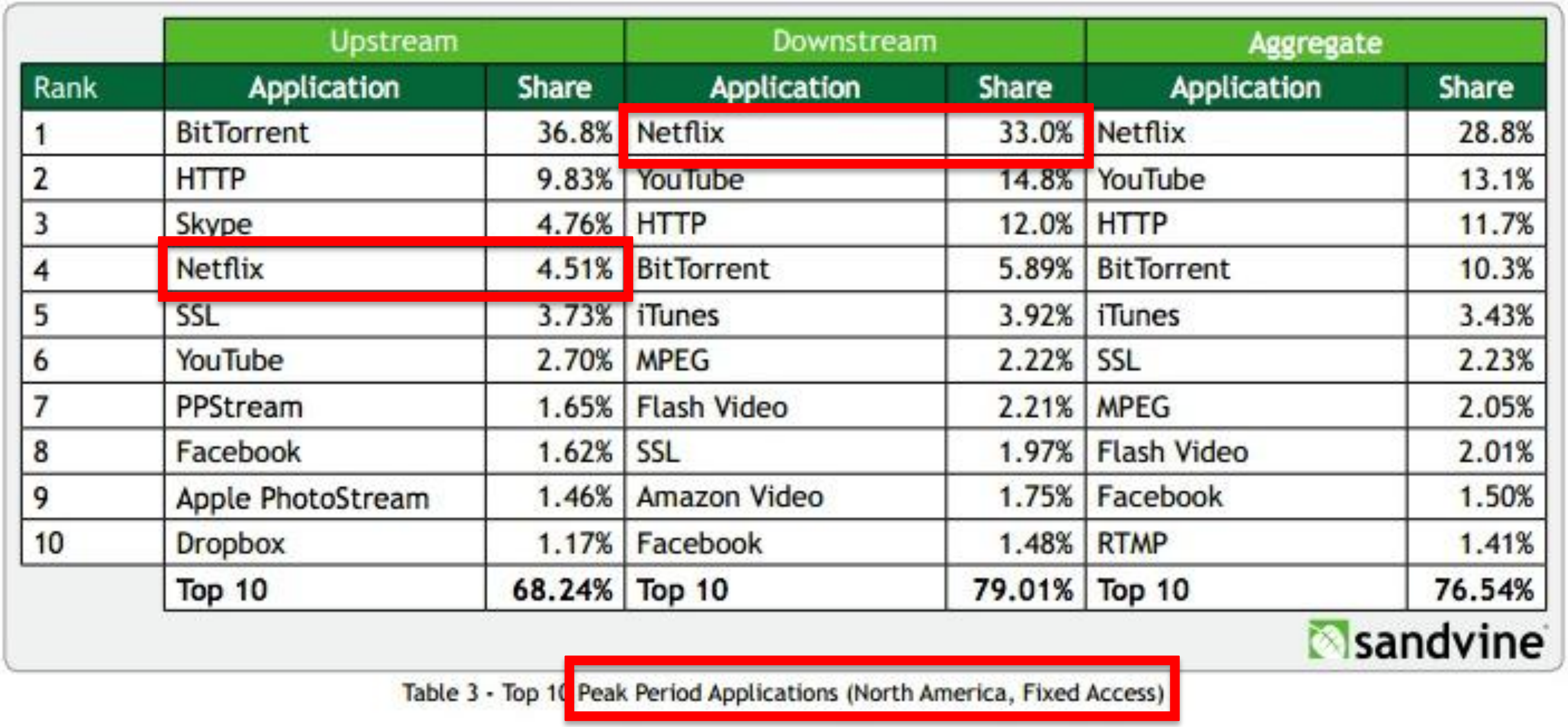
From a compute point of view, the system has modest requirements moving bits from the storage to network packets on the interface. To reduce the power usage and hence also cooling requirement (which in turn reduces vibration from case fans) we use a single low power 4 core Intel Sandy Bridge CPU on a small form factor Supermicro mATX board with the full 32 GB of RAM installed.

We use redundant, hot swappable power supply units that have interchangeable AC and DC options for maximum installation flexibility. Zippy reversed the fan rotation of the units to allow mounting at the front of the case, and thus allow network and power connects to be positioned here.

The network card has two 10 Gbps modules, which can power a variety of SR and LR optic modules, for installation flexibility and scalable interconnection.

The following system was developed and first deployed at the end of 2011.

November 2012 Traffic



	Upstream		Downstream		Aggregate	
Rank	Application	Share	Application	Share	Application	Share
1	BitTorrent	36.8%	Netflix	33.0%	Netflix	28.8%
2	HTTP	9.83%	YouTube	14.8%	YouTube	13.1%
3	Skype	4.76%	HTTP	12.0%	HTTP	11.7%
4	Netflix	4.51%	BitTorrent	5.89%	BitTorrent	10.3%
5	SSL	3.73%	iTunes	3.92%	iTunes	3.43%
6	YouTube	2.70%	MPEG	2.22%	SSL	2.23%
7	PPStream	1.65%	Flash Video	2.21%	MPEG	2.05%
8	Facebook	1.62%	SSL	1.97%	Flash Video	2.01%
9	Apple PhotoStream	1.46%	Amazon Video	1.75%	Facebook	1.50%
10	Dropbox	1.17%	Facebook	1.48%	RTMP	1.41%
	Top 10	68.24%	Top 10	79.01%	Top 10	76.54%

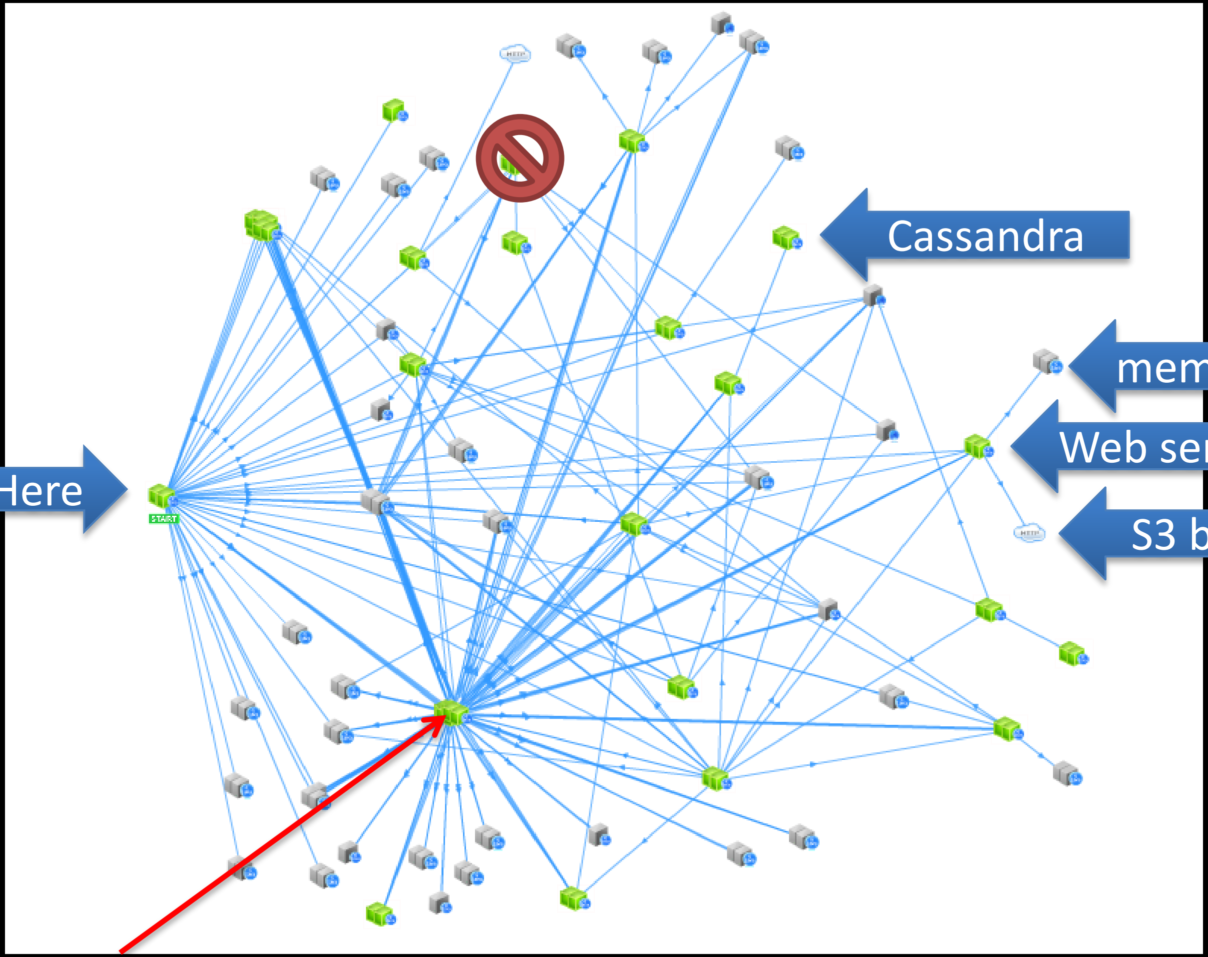
sandvine

Table 3 - Top 10 Peak Period Applications (North America, Fixed Access)

Real Web Server Dependencies Flow

(Netflix Home page business transaction as seen by AppDynamics)

Each icon is three to a few hundred instances across three AWS zones



Cassandra

memcached

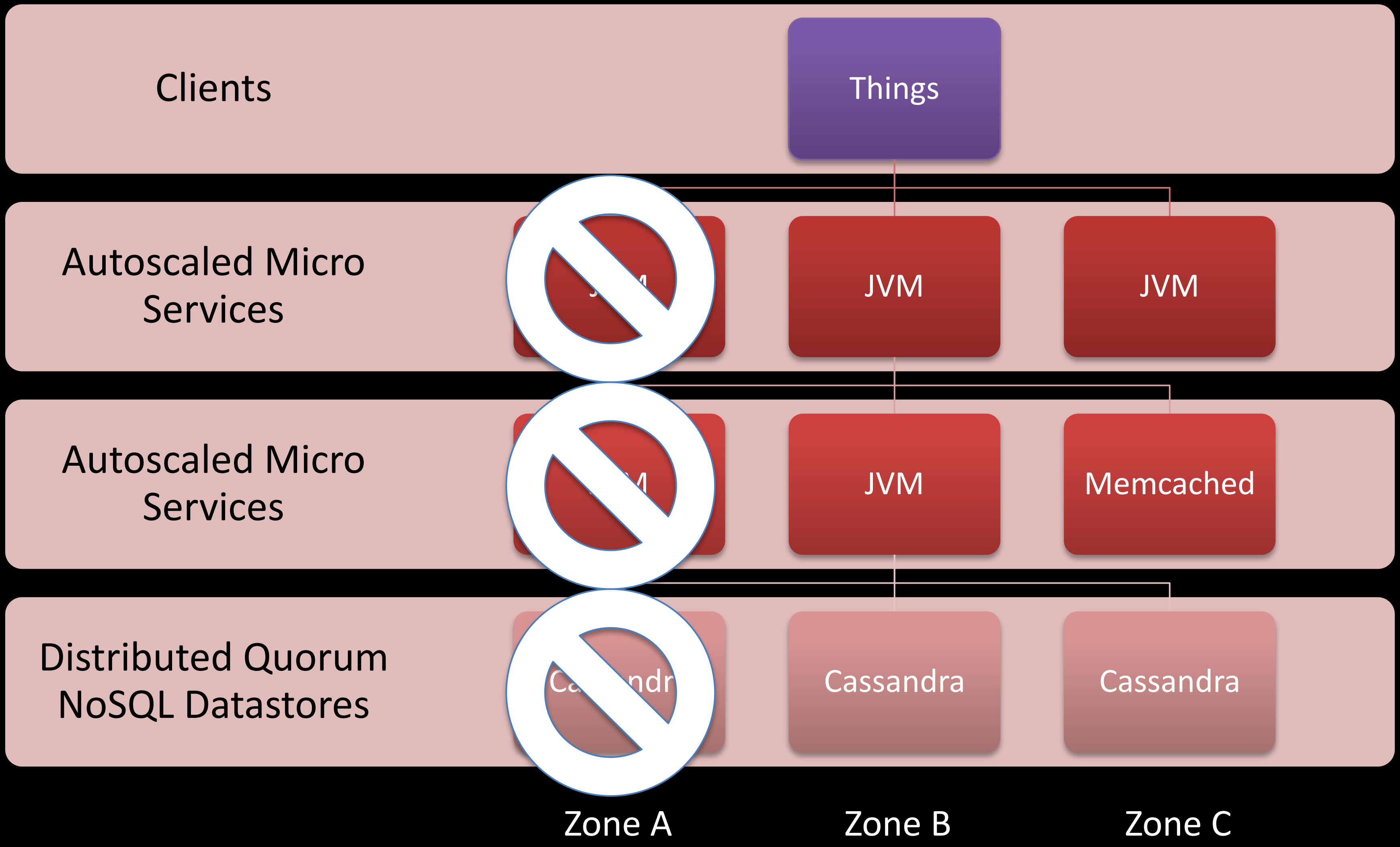
Web service

S3 bucket

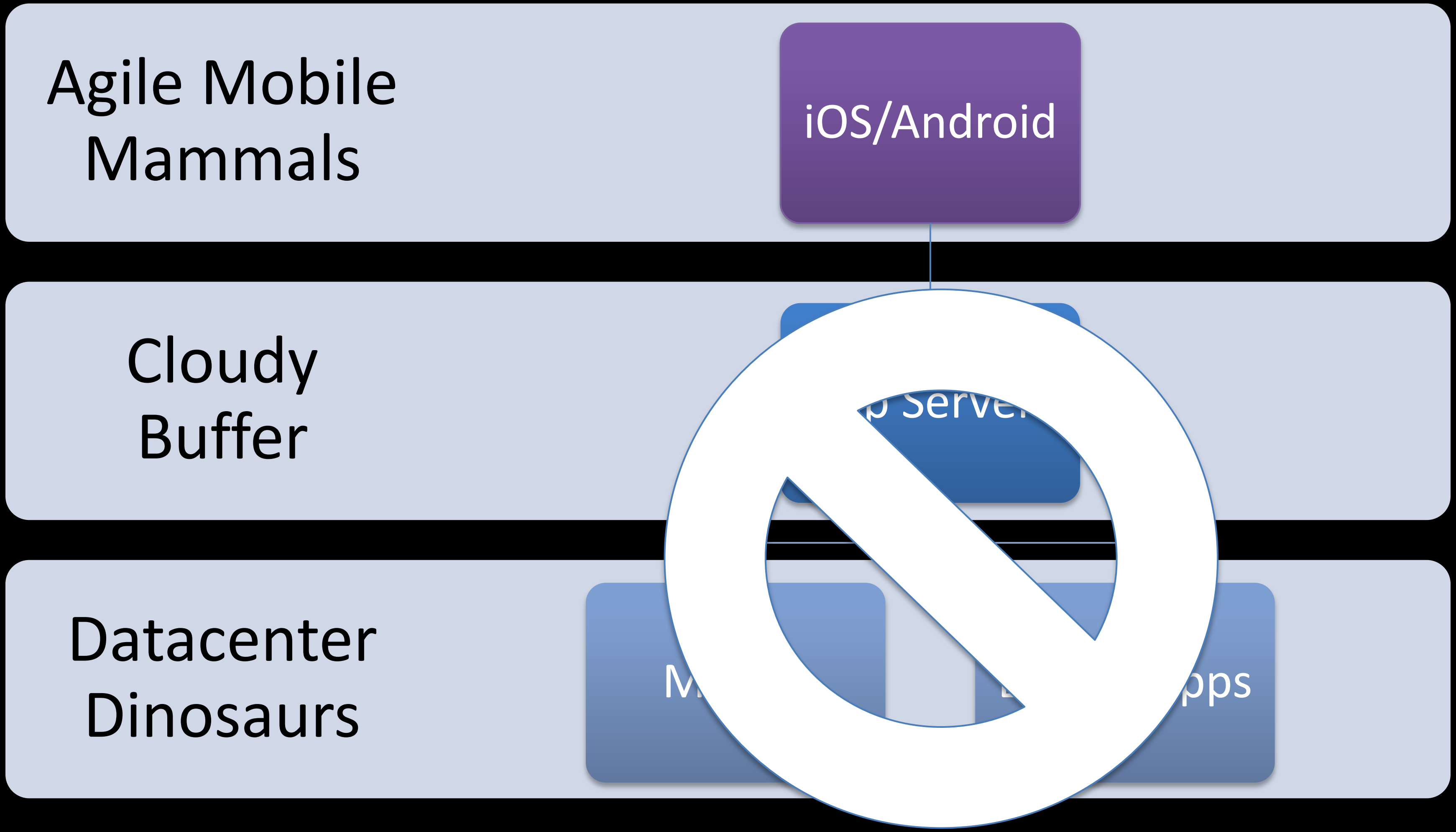
Start Here

Three Personalization movie group choosers (for US, Canada and Latam)

Cloud Native Architecture


Clients
Things
Autoscaled Micro Services
JVM
JVM
Autoscaled Micro Services
JVM
Memcached
Distributed Quorum NoSQL Datastores
Cassandra
Cassandra
Zone A
Zone B
Zone C

Non-Native Cloud Architecture



Agile Mobile Mammals

Cloudy Buffer

Datacenter Dinosaurs

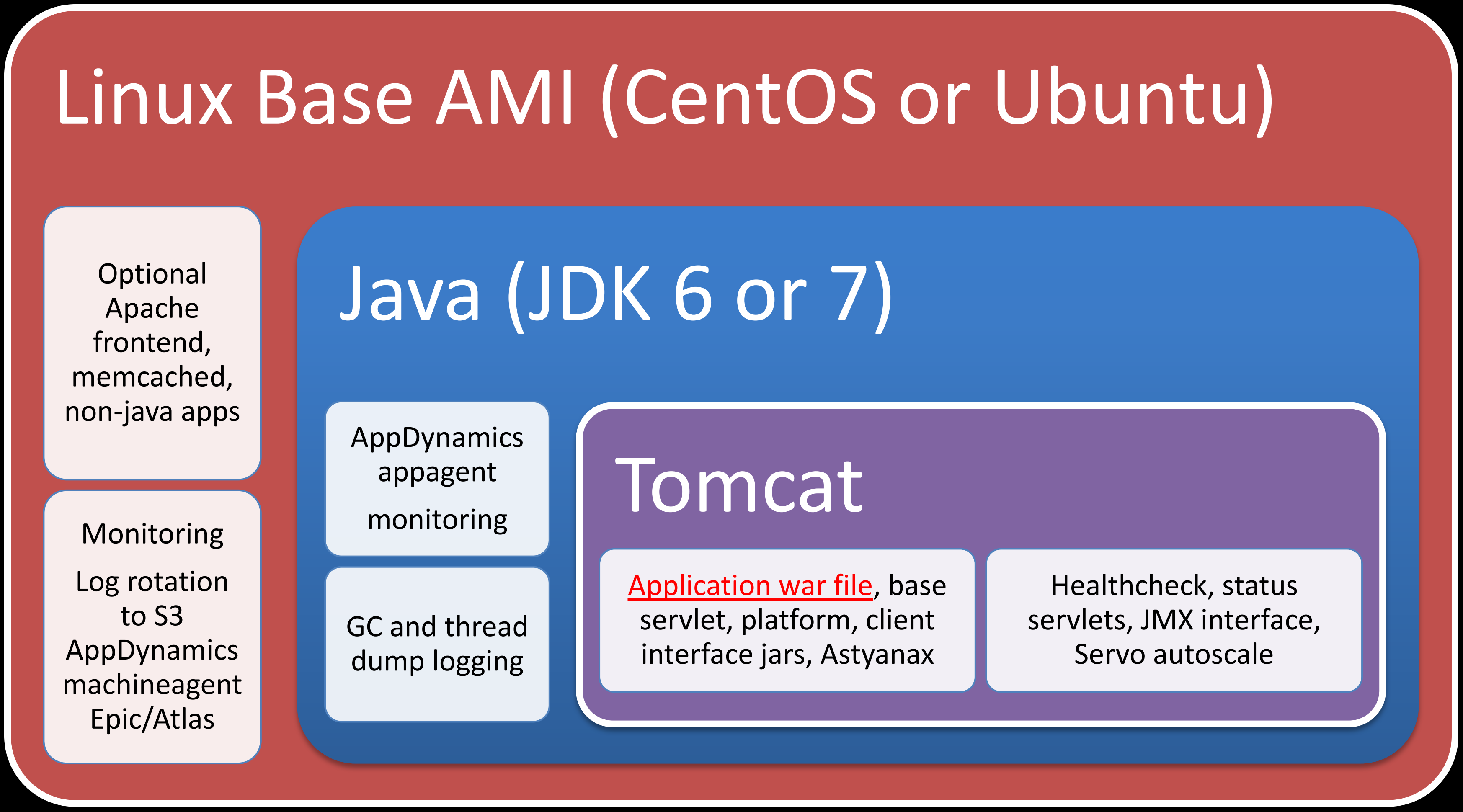
New Anti-Fragile Patterns

Micro-services

Chaos engines

Highly available systems composed
from ephemeral components

Stateless Micro-Service Architecture



Linux Base AMI (CentOS or Ubuntu)

Optional Apache frontend, memcached, non-java apps

Monitoring

Log rotation to S3

AppDynamics machineagent

Epic/Atlas

Java (JDK 6 or 7)

AppDynamics appagent

monitoring

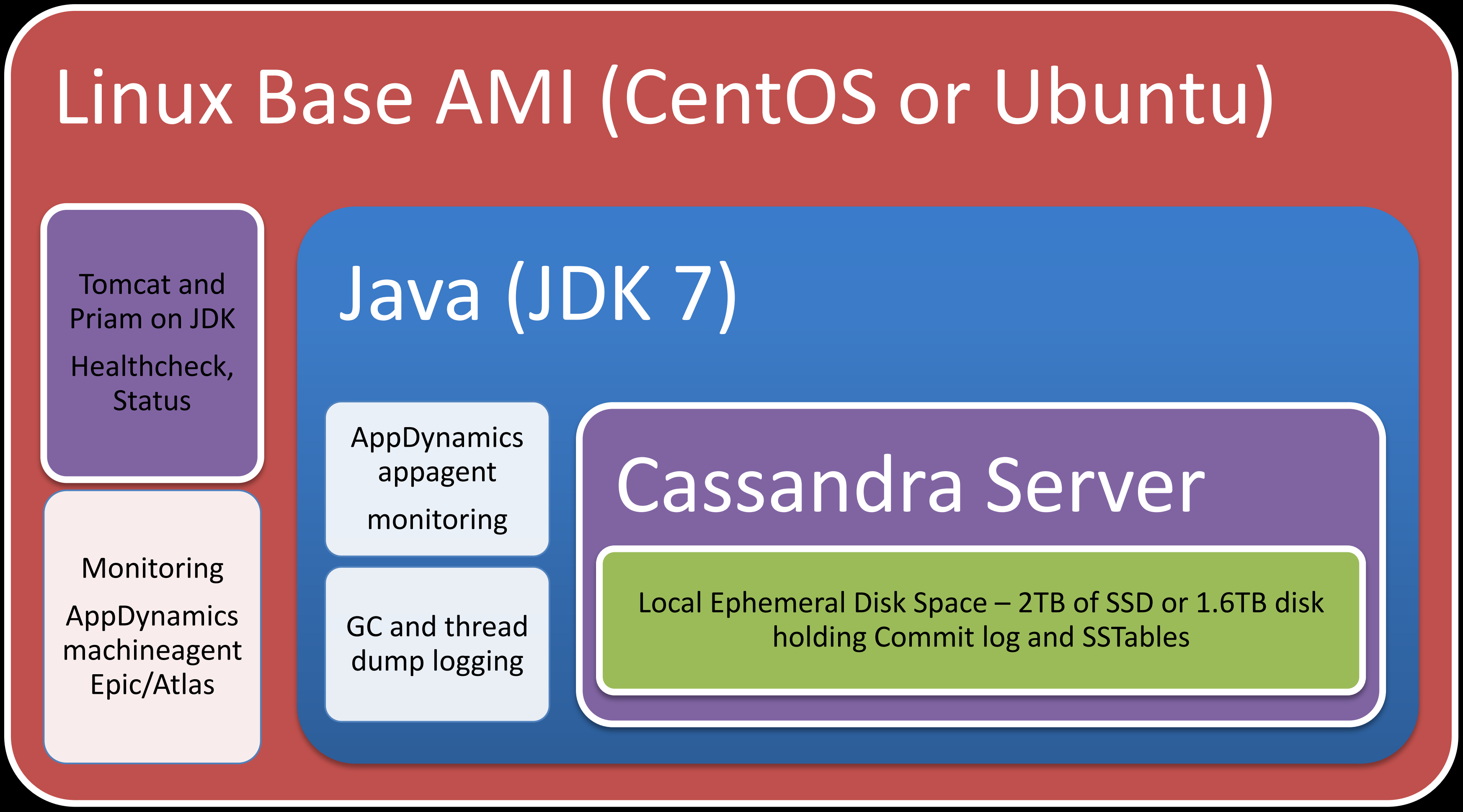
GC and thread dump logging

Tomcat

Application war file, base servlet, platform, client interface jars, Astyanax

Healthcheck, status servlets, JMX interface, Servo autoscale

Cassandra Instance Architecture



Linux Base AMI (CentOS or Ubuntu)

Tomcat and Priam on JDK

Healthcheck, Status

Monitoring

AppDynamics machineagent

Epic/Atlas

Java (JDK 7)

AppDynamics appagent

monitoring

GC and thread dump logging

Cassandra Server

Local Ephemeral Disk Space – 2TB of SSD or 1.6TB disk holding Commit log and SSTables

Cloud Native

Master copies of data are cloud resident

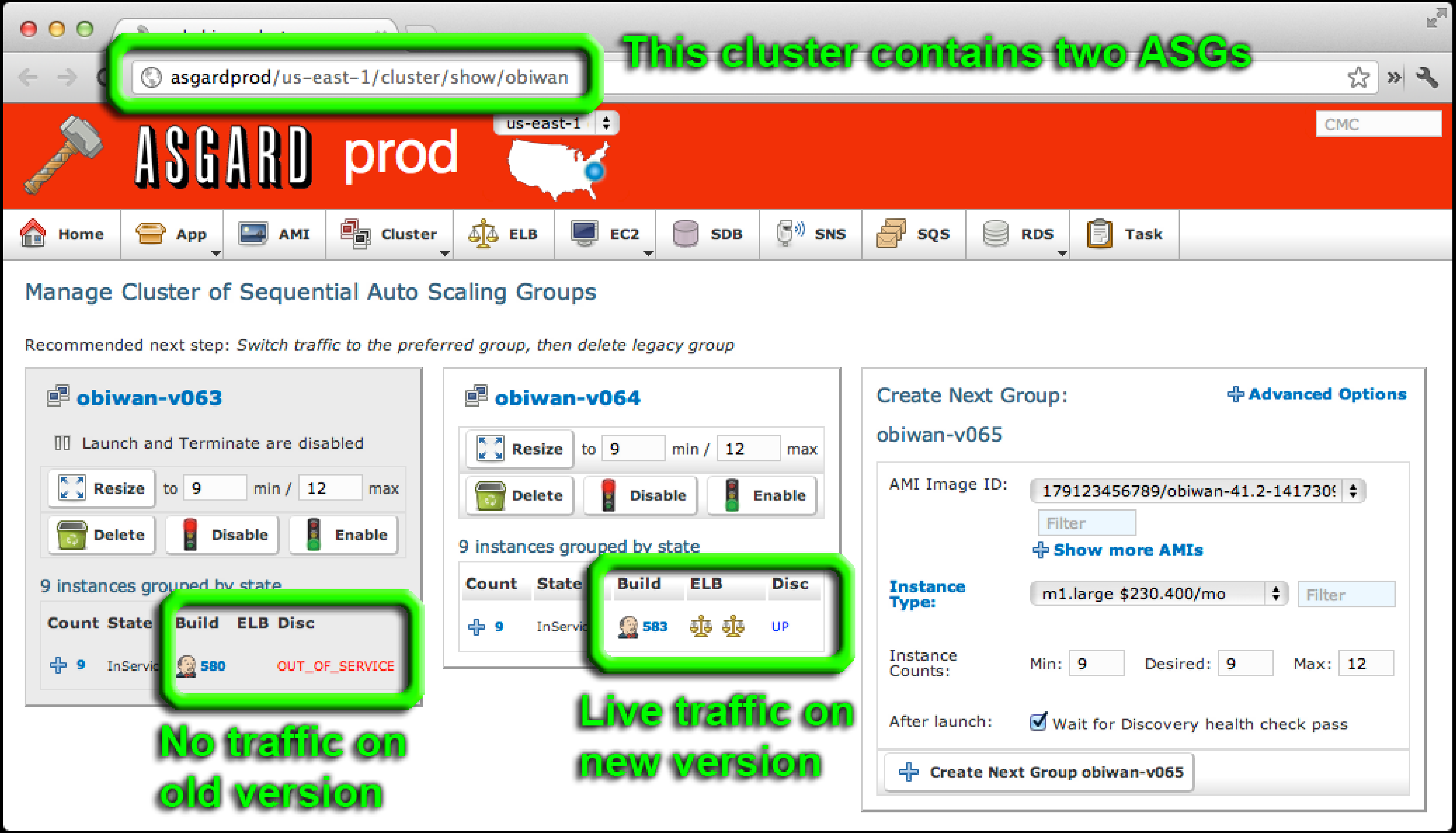
Everything is dynamically provisioned

All services are ephemeral

Dynamic Scalability

Asgard

http://techblog.netflix.com/2012/06/asgard-web-based-cloud-management-and.html



asgardprod/us-east-1/cluster/show/obiwan

This cluster contains two ASGs

Manage Cluster of Sequential Auto Scaling Groups

Recommended next step: *Switch traffic to the preferred group, then delete legacy group*

obiwan-v063

Launch and Terminate are disabled

9 instances grouped by state

Count	State	Build	ELB	Disc
9	InServi	580		OUT_OF_SERVICE

No traffic on old version

obiwan-v064

9 instances grouped by state

Count	State	Build	ELB	Disc
9	InServi	583	⚖ ⚖	UP

Live traffic on new version

Create Next Group:

obiwan-v065

AMI Image ID: 179123456789/obiwan-41.2-141730

Instance Type: m1.large $230.400/mo

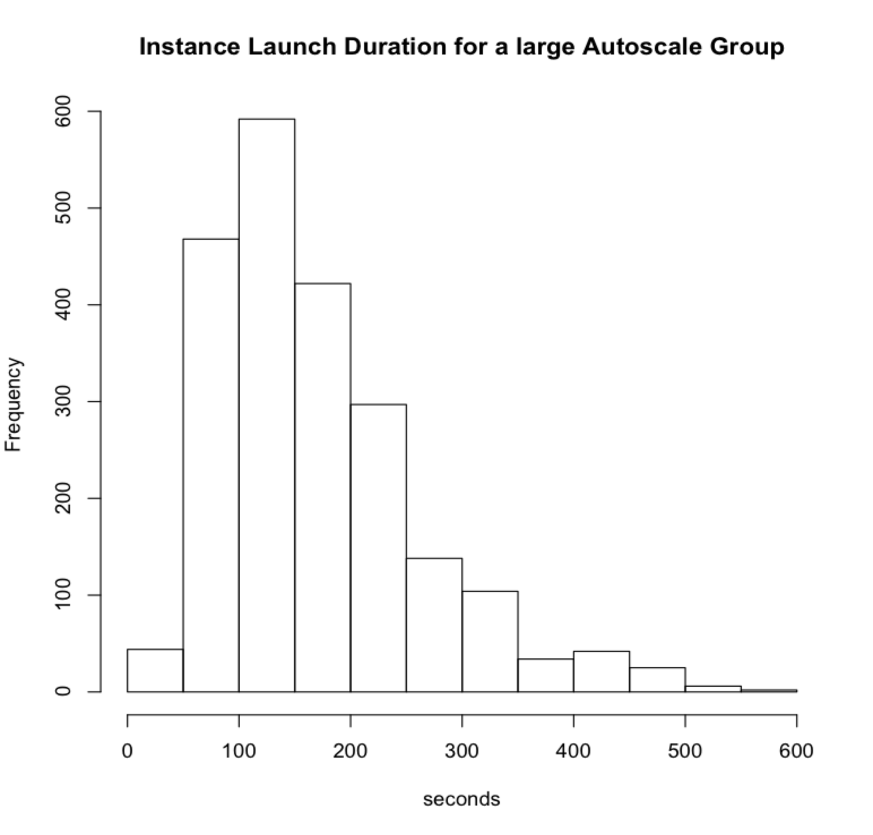
Instance Counts: Min: 9 Desired: 9 Max: 12

After launch: ☑ Wait for Discovery health check pass

Cloud Deployment Scalability

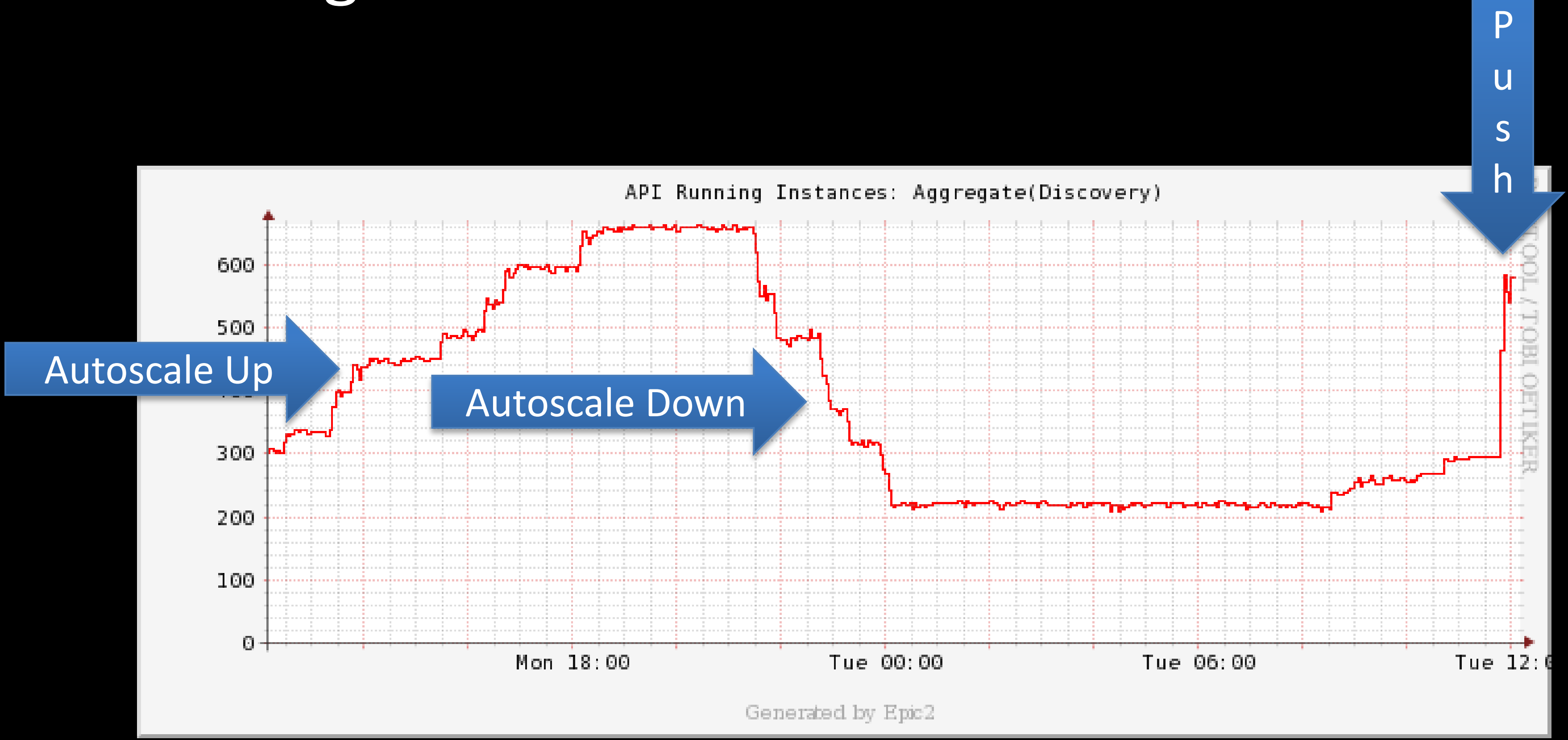
New Autoscaled AMI – zero to 500 instances from 21:38:52 - 21:46:32, 7m40s
Scaled up and down over a few days, total 2176 instance launches, m2.2xlarge (4 core 34GB)

```
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 41.0   104.2   149.0   171.8   215.8   562.0
```



Ephemeral Instances

- Largest services are autoscaled
- Average lifetime of an instance is 36 hours

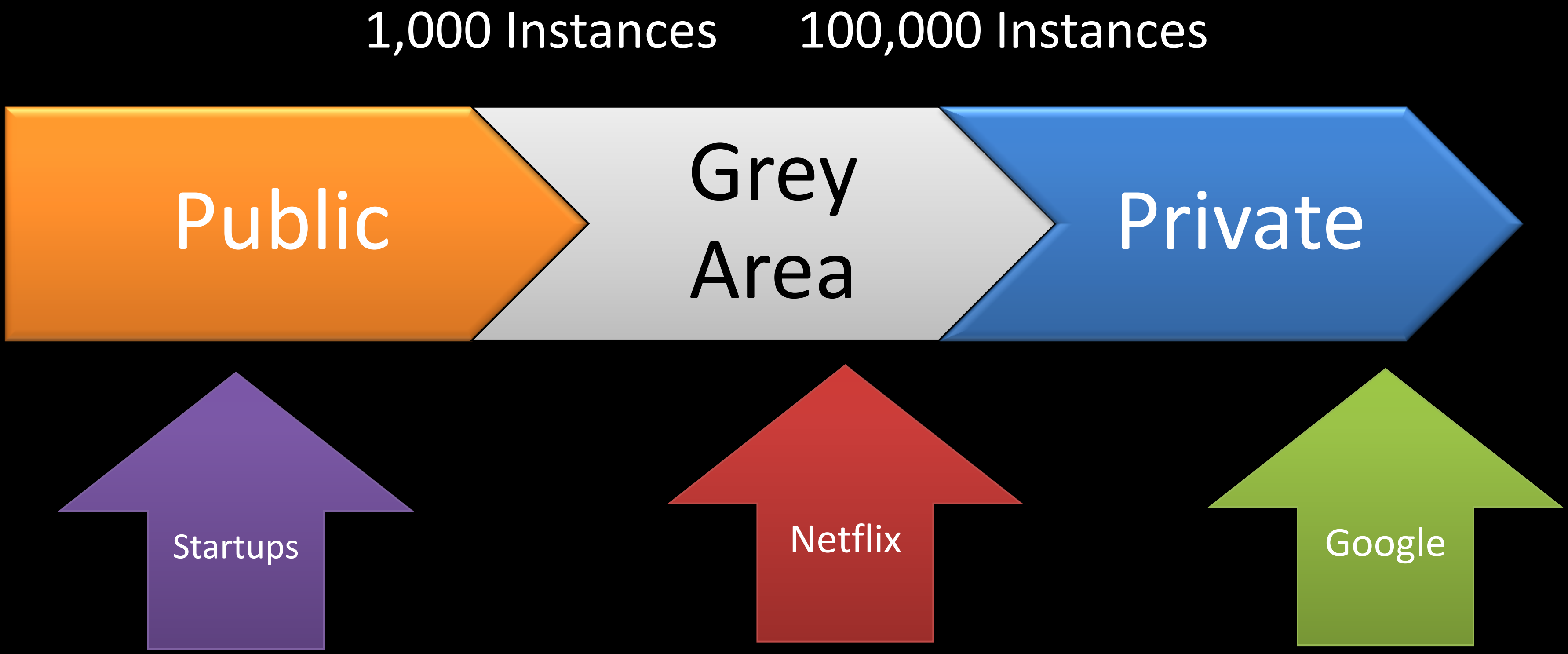


Autoscale Up

Autoscale Down

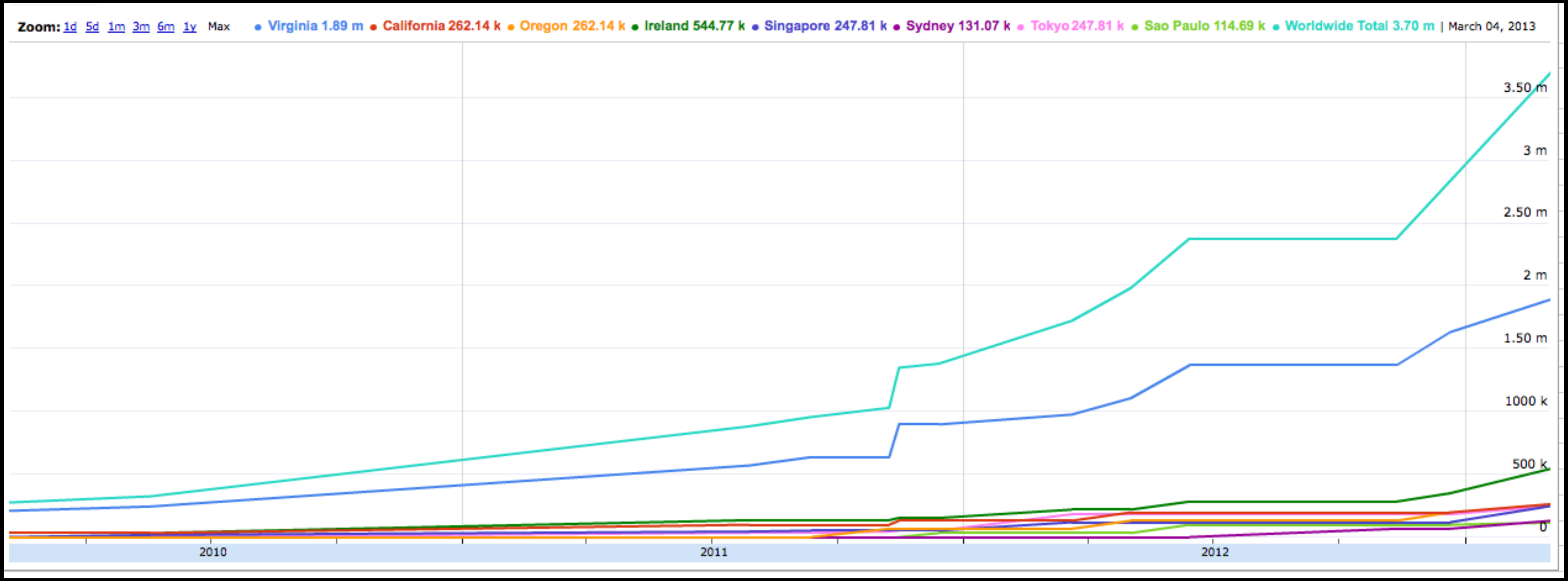
Push

Leveraging Public Scale



How big is Public?

AWS Maximum Possible Instance Count 3.7 Million
Growth >10x in Three Years, >2x Per Annum



AWS upper bound estimate based on the number of public IP Addresses
Every provisioned instance gets a public IP by default

Availability

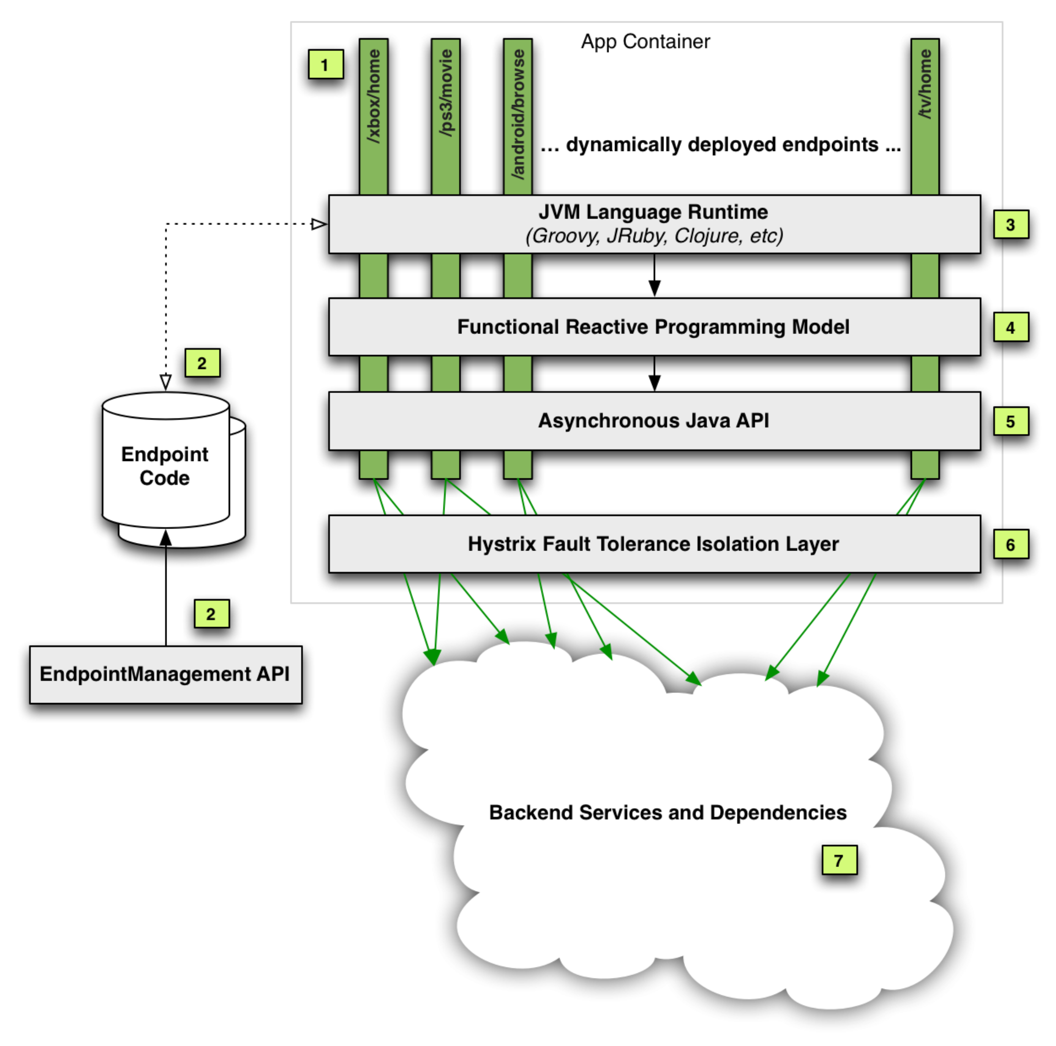
Is it running yet?

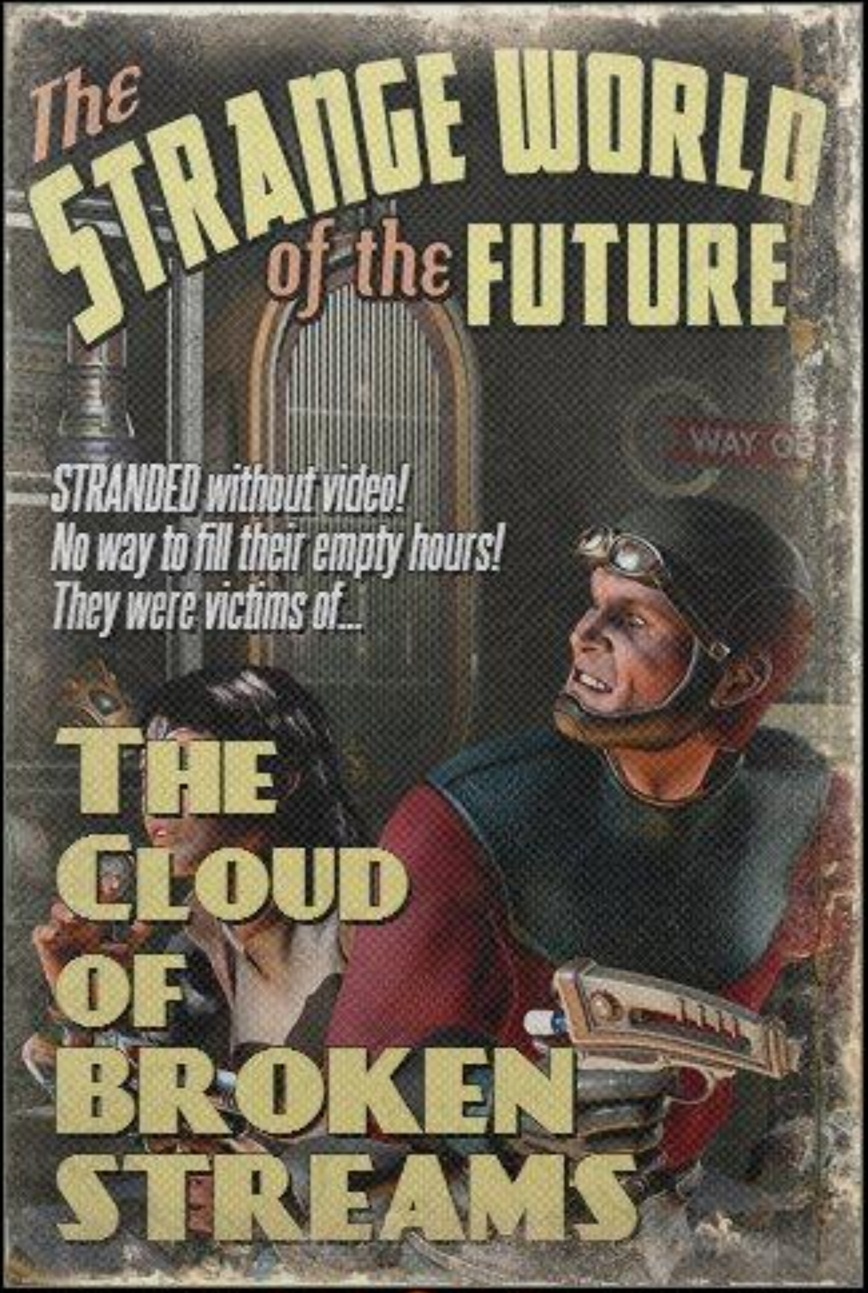
How many places is it running in?

How far apart are those places?

Antifragile API Patterns

Functional Reactive with Circuit Breakers and Bulkheads


App Container
1
/xbox/home
/ps3/movie
/android/browse
... dynamically deployed endpoints ...
/tv/home
JVM Language Runtime
(Groovy, JRuby, Clojure, etc)
3
Functional Reactive Programming Model
4
Asynchronous Java API
5
Hystrix Fault Tolerance Isolation Layer
6
2
Endpoint Code
2
EndpointManagement API
Backend Services and Dependencies
7


The STRANGE WORLD of the FUTURE
STRANDED without video!
No way to fill their empty hours!
They were victims of...
THE CLOUD OF BROKEN STREAMS
WAY O

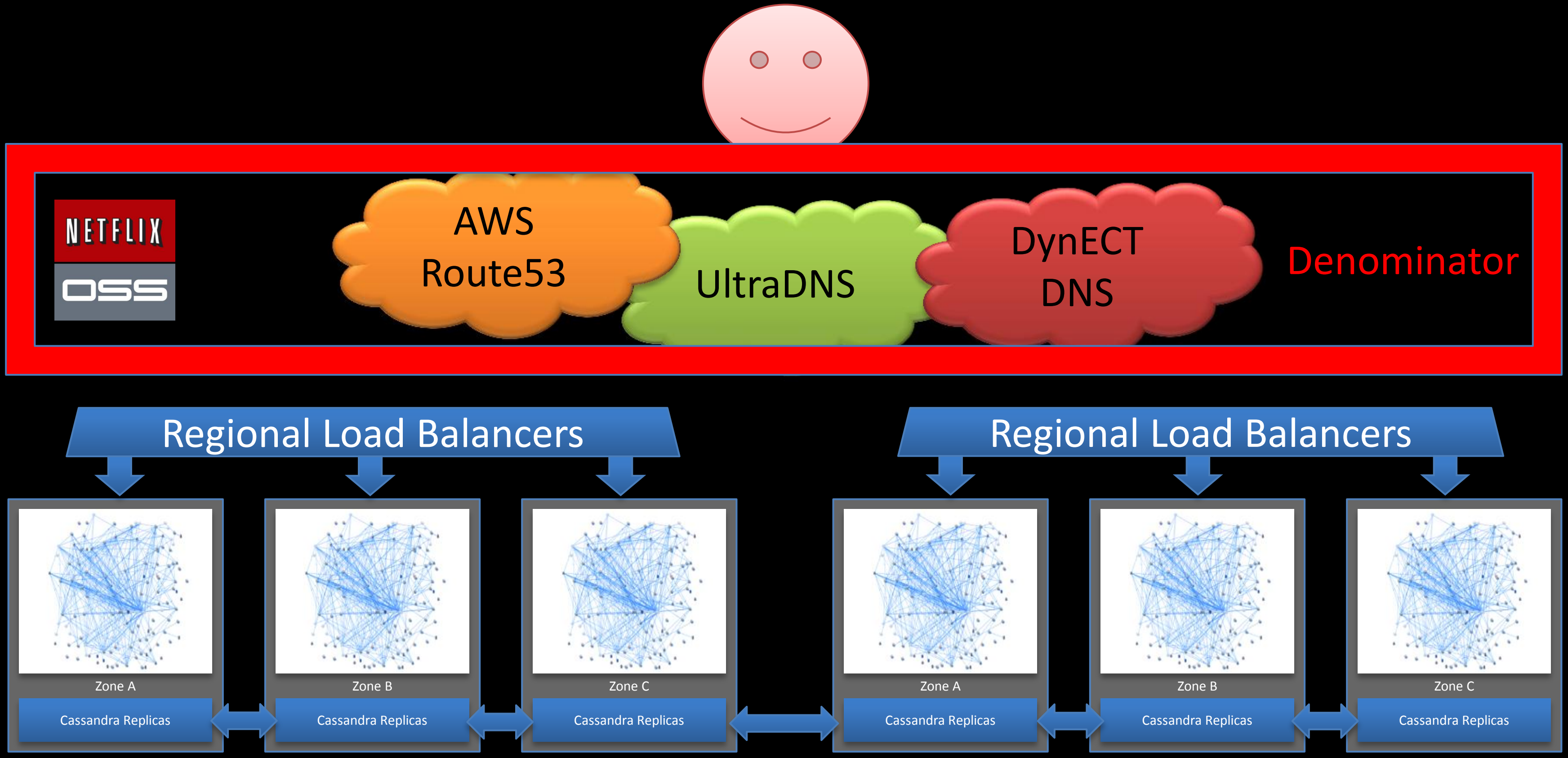
CREATED WITH PULP-O-MIZER COVER MAKER

Outages

- Running very fast with scissors
  - Mostly self inflicted – bugs, mistakes
  - Some caused by AWS bugs and mistakes

- Next step is multi-region
  - Investigating and building in stages during 2013
  - Could have prevented some of our 2012 outages

Managing Multi-Region Availability



A portable way to manage multiple DNS providers from Java

Configuration State Management

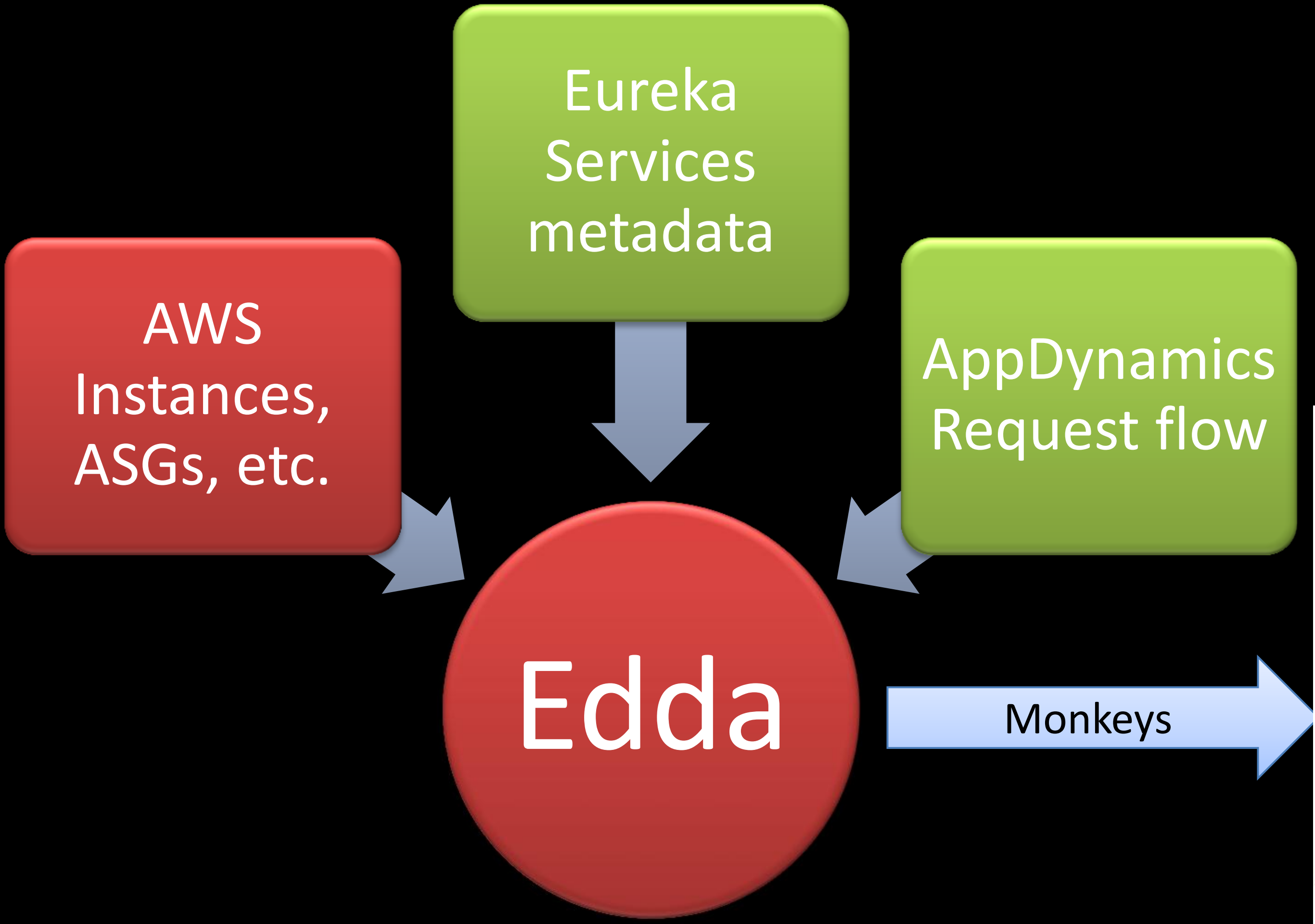
Datacenter CMDB’s woeful

Cloud native is the solution

Dependably complete

Edda – Configuration History

http://techblog.netflix.com/2012/11/edda-learn-stories-of-your-cloud.html



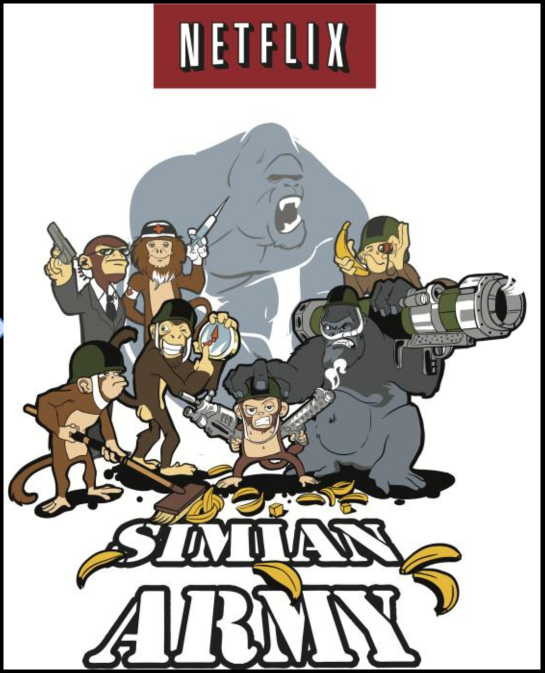
Eureka Services metadata

AWS Instances, ASGs, etc.

AppDynamics Request flow

Edda

Monkeys



Edda Query Examples

Find any instances that have ever had a specific public IP address

```
$ curl "http://edda/api/v2/view/instances;publicIpAddress=1.2.3.4;_since=0"
 ["i-0123456789","i-012345678a","i-012345678b”]
```

Show the most recent change to a security group

```
$ curl "http://edda/api/v2/aws/securityGroups/sg-0123456789;_diff;_all;_limit=2"
--- /api/v2/aws.securityGroups/sg-0123456789;_pp;_at=1351040779810
+++ /api/v2/aws.securityGroups/sg-0123456789;_pp;_at=1351044093504
@@ -1,33 +1,33 @@
 {
…
      "ipRanges" : [
        "10.10.1.1/32",
        "10.10.1.2/32",
+       "10.10.1.3/32",
-       "10.10.1.4/32"
…
 }
```

NETFLIX OSS

A Cloud Native Open Source Platform

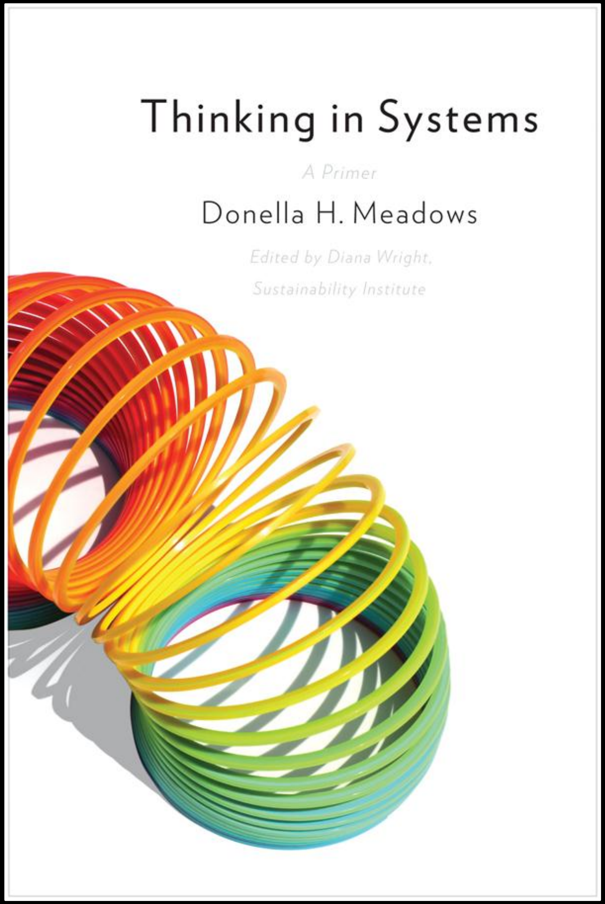
Inspiration

The Pragmatic Programmers

Release It!

Design and Deploy Production-Ready Software

Michael T. Nygard

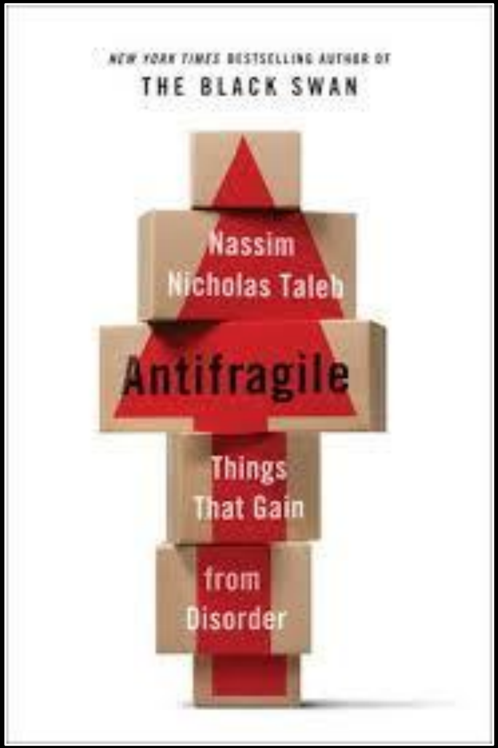


Thinking in Systems

A Primer

Donella H. Meadows

Edited by Diana Wright, Sustainability Institute

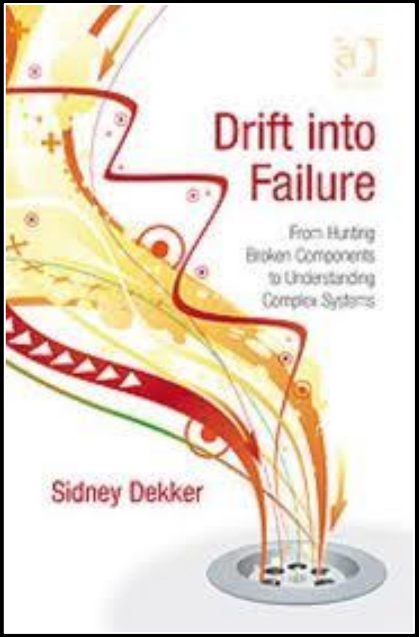


NEW YORK TIMES BESTSELLING AUTHOR OF THE BLACK SWAN

Nassim Nicholas Taleb

Antifragile

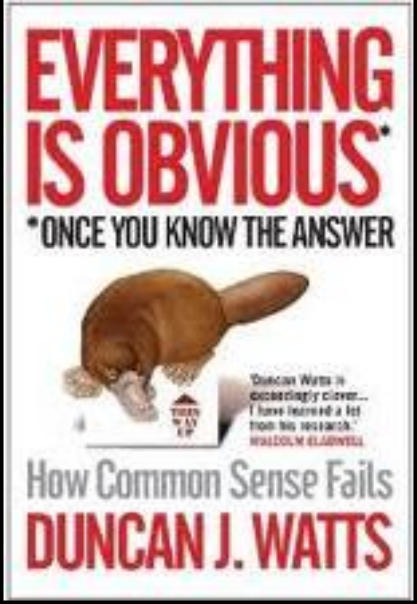
Things That Gain from Disorder



Drift into Failure

From Hunting Broken Components to Understanding Complex Systems

Sidney Dekker



EVERYTHING IS OBVIOUS*

*ONCE YOU KNOW THE ANSWER

'Duncan Watts is exceedingly clever... I have learned a lot from his research.' MALCOLM GLADWELL

How Common Sense Fails

DUNCAN J. WATTS

Three Questions

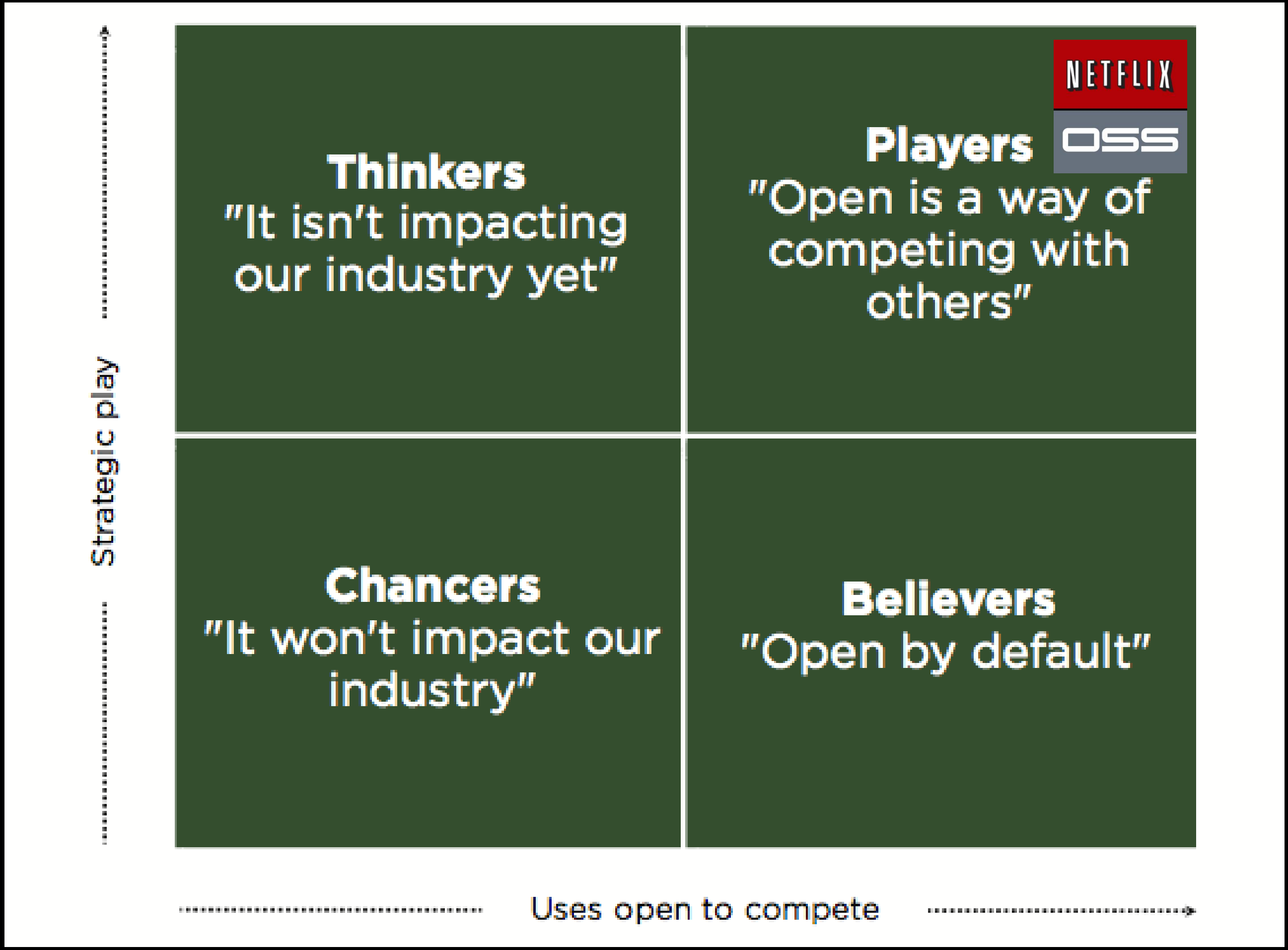
Why is Netflix doing this?

How does it all fit together?

What is coming next?

Beware of Geeks Bearing Gifts: Strategies for an Increasingly Open Economy

Simon Wardley - Researcher at the Leading Edge Forum



Thinkers
"It isn't impacting our industry yet"

Players
"Open is a way of competing with others"

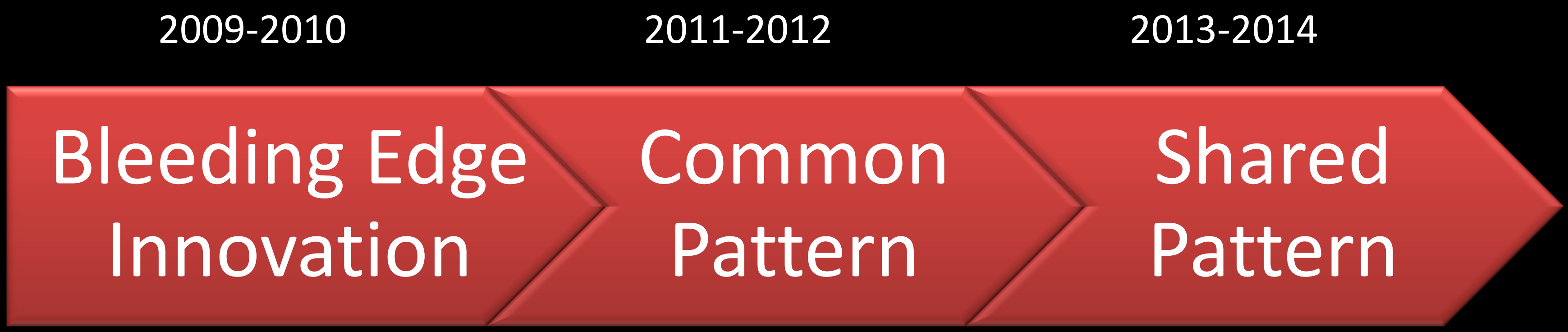
Chancers
"It won't impact our industry"

Believers
"Open by default"

How did Netflix get ahead?

Netflix Business + Developer Org	**Traditional IT Operations**
• Doing it right now	• Taking their time
• SaaS Applications	• Pilot private cloud projects
• PaaS for agility	• Beta quality installations
• Public IaaS for AWS features	• Small scale
• Big data in the cloud	• Integrating several vendors
• Integrating many APIs	• Paying big $ for software
• FOSS from github	• Paying big $ for consulting
• Renting hardware for 1hr	• Buying hardware for 3yrs
• Coding in Java/Groovy/Scala	• Hacking at scripts

Netflix Platform Evolution



2009-2010

Bleeding Edge Innovation

2011-2012

Common Pattern

2013-2014

Shared Pattern

Netflix ended up several years ahead of the industry, but it's not a sustainable position

Making it easy to follow

Exploring the wild west each time vs. laying down a shared route

California
The Cornucopia of the World
Room for Millions of Immigrants
43.795.000.
Acres of Government Lands Untaken
Railroad & Private Land for a Million Farmers
A Climate for Health & Wealth without Cyclones or Buzzards

Goals

Establish our solutions as Best Practices / Standards

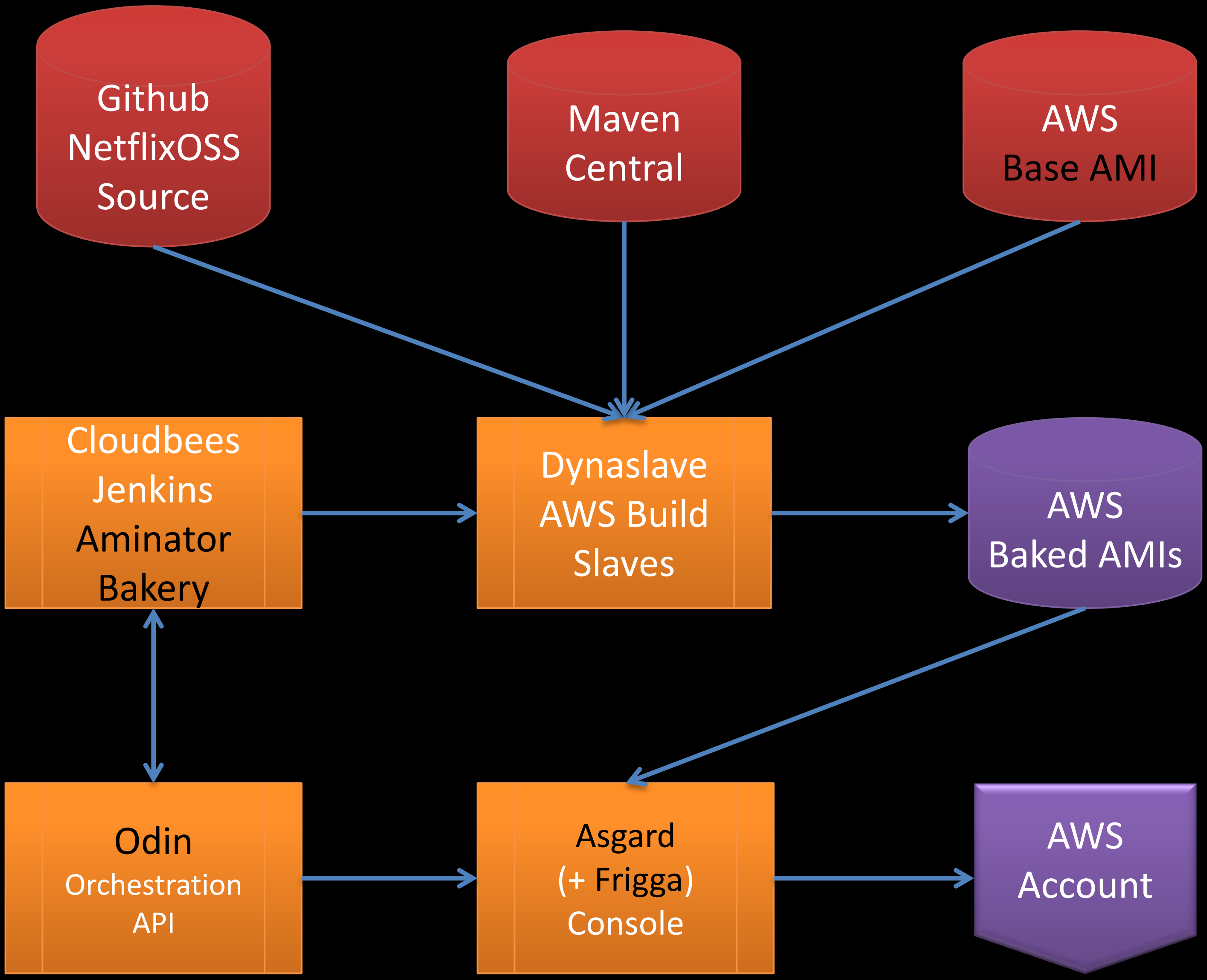
Hire, Retain and Engage Top Engineers

Build up Netflix Technology Brand

Benefit from a shared ecosystem

How does it all fit together?

NetflixOSS Continuous Build and Deployment



Github NetflixOSS Source

Maven Central

AWS Base AMI

Cloudbees Jenkins Aminator Bakery

Dynaslave AWS Build Slaves

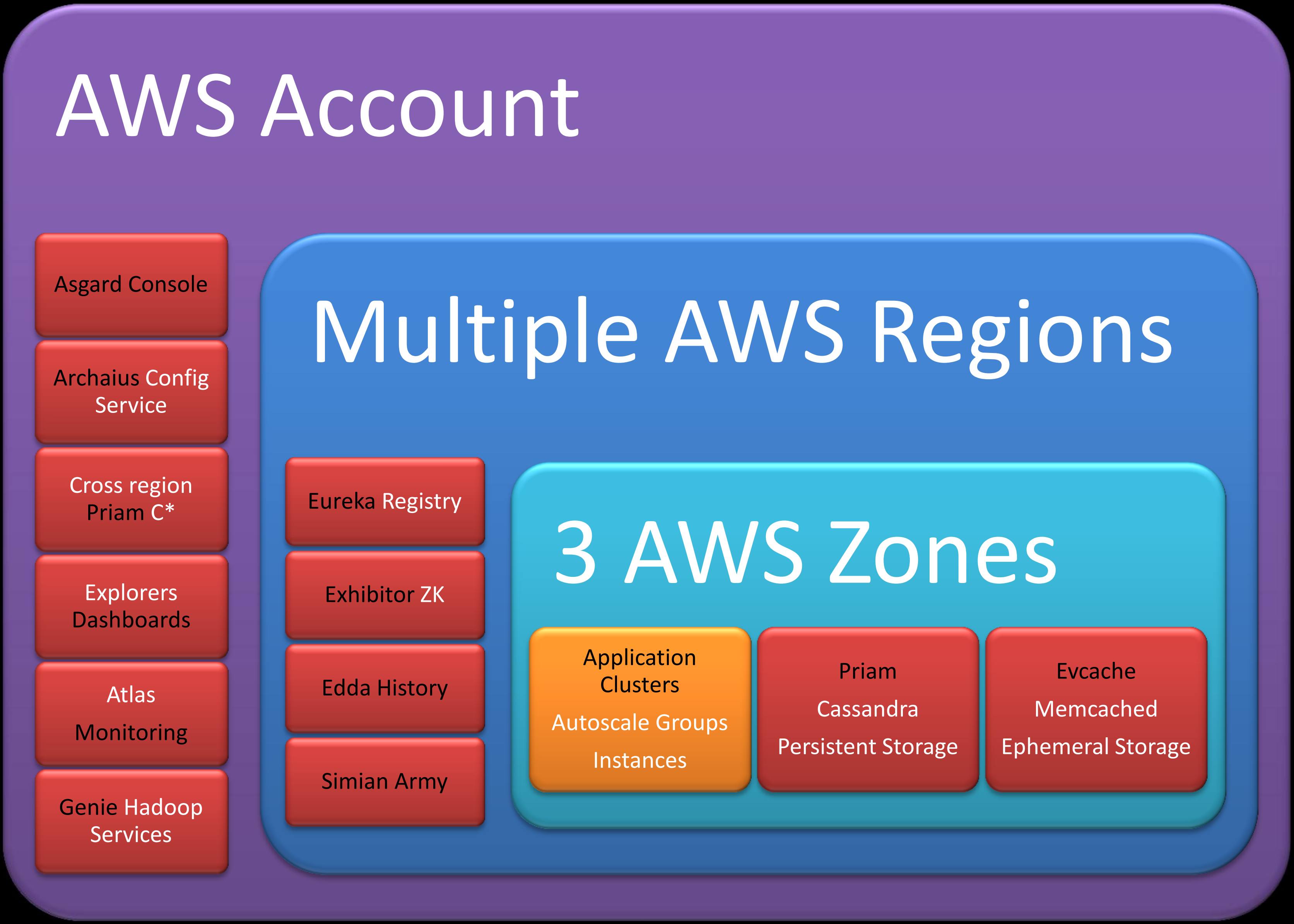
AWS Baked AMIs

Odin Orchestration API

Asgard (+ Frigga) Console

AWS Account

NetflixOSS Services Scope



AWS Account

Asgard Console

Archaius Config Service

Cross region Priam C*

Explorers Dashboards

Atlas

Monitoring

Genie Hadoop Services

Multiple AWS Regions

Eureka Registry

Exhibitor ZK

Edda History

Simian Army

3 AWS Zones

Application Clusters

Autoscale Groups

Instances

Priam

Cassandra

Persistent Storage

Evcache

Memcached

Ephemeral Storage

NetflixOSS Instance Libraries

Initialization

- Baked AMI – Tomcat, Apache, your code
- Governator – Guice based dependency injection
- Archaius – dynamic configuration properties client
- Eureka - service registration client

Service Requests

- Karyon - Base Server for inbound requests
- RxJava – Reactive pattern
- Hystrix/Turbine – dependencies and real-time status
- Ribbon - REST Client for outbound calls

Data Access

- Astyanax – Cassandra client and pattern library
- Evcache – Zone aware Memcached client
- Curator – Zookeeper patterns
- Denominator – DNS routing abstraction

Logging

- Blitz4j – non-blocking logging
- Servo – metrics export for autoscaling
- Atlas – high volume instrumentation

NetflixOSS Testing and Automation

Test Tools

- CassJmeter – Load testing for Cassandra
- Circus Monkey – Test account reservation rebalancing

Maintenance

- Janitor Monkey – Cleans up unused resources
- Efficiency Monkey
- Doctor Monkey
- Howler Monkey – Complains about expiring certs

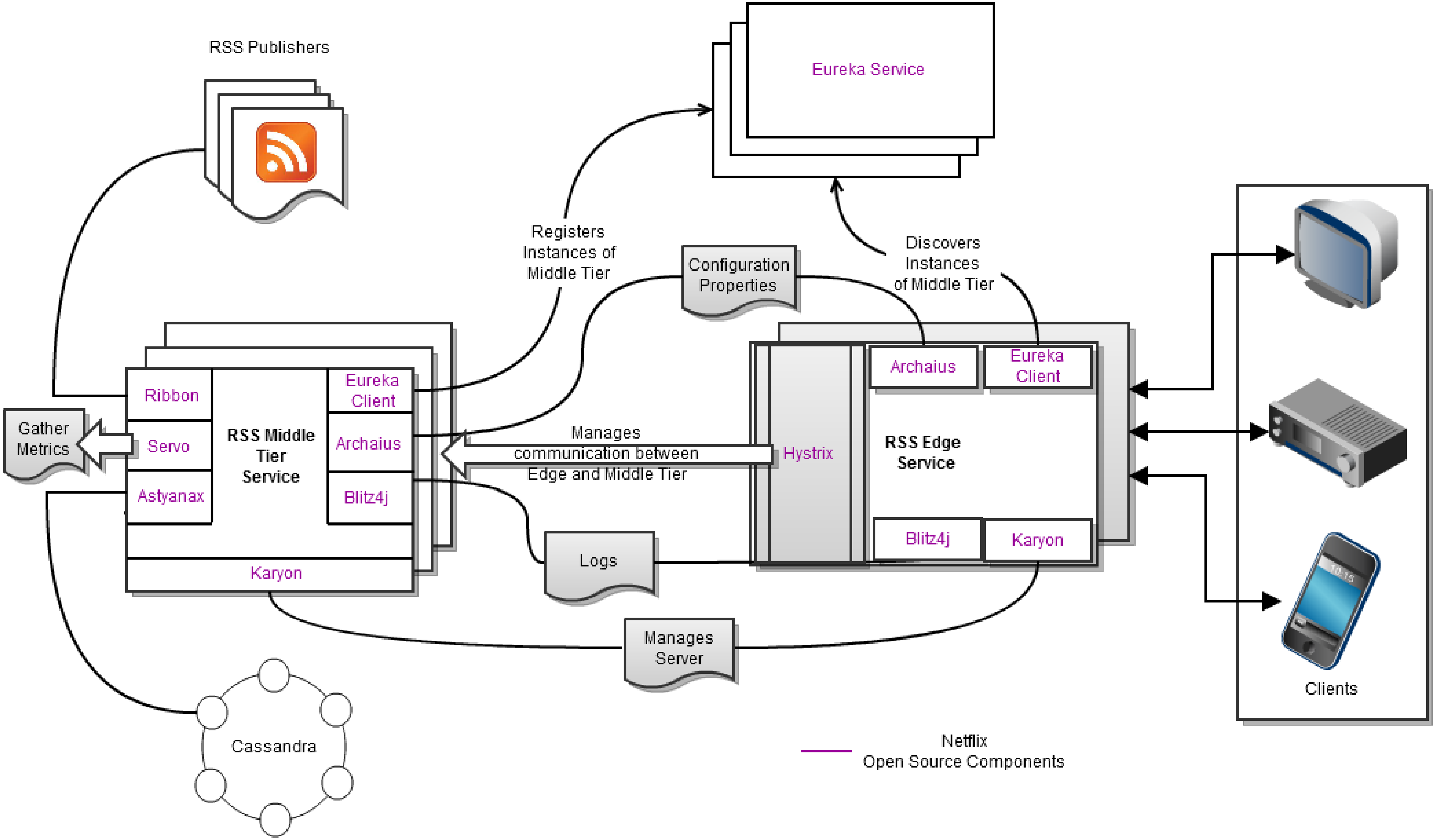
Availability

- Chaos Monkey – Kills Instances
- Chaos Gorilla – Kills Availability Zones
- Chaos Kong – Kills Regions
- Latency Monkey – Latency and error injection

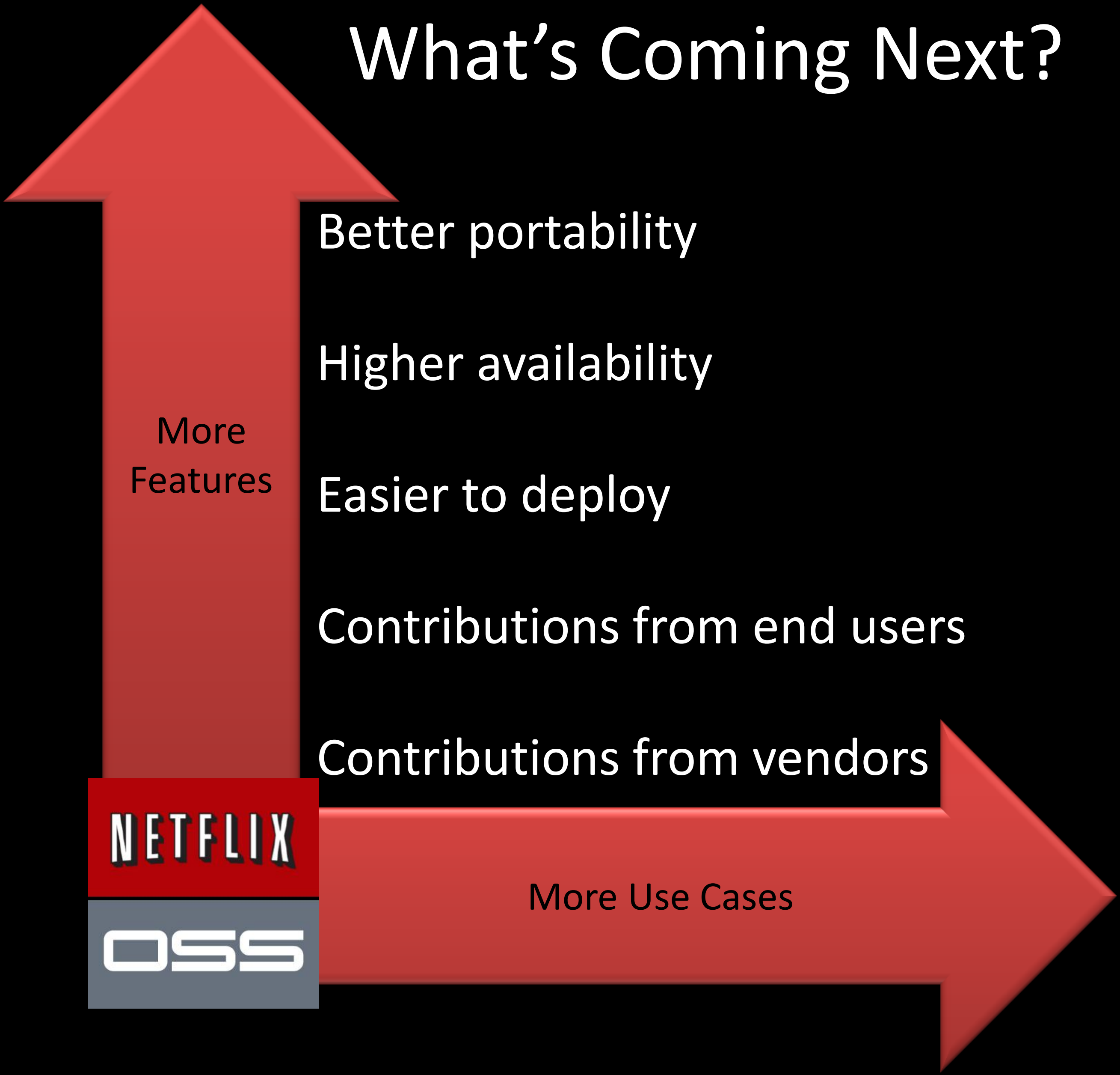
Security

- Security Monkey
- Conformity Monkey

Example Application – RSS Reader



What’s Coming Next?



Better portability

Higher availability

Easier to deploy

Contributions from end users

Contributions from vendors

Vendor Driven Portability

Interest in using NetflixOSS for Enterprise Private Clouds

NETFLIX OSS

EUCALYPTUS

"It's done when it runs Asgard"
Functionally complete
Demonstrated March
Release 3.3 in 2Q13

Some vendor interest
Needs AWS compatible Autoscaler

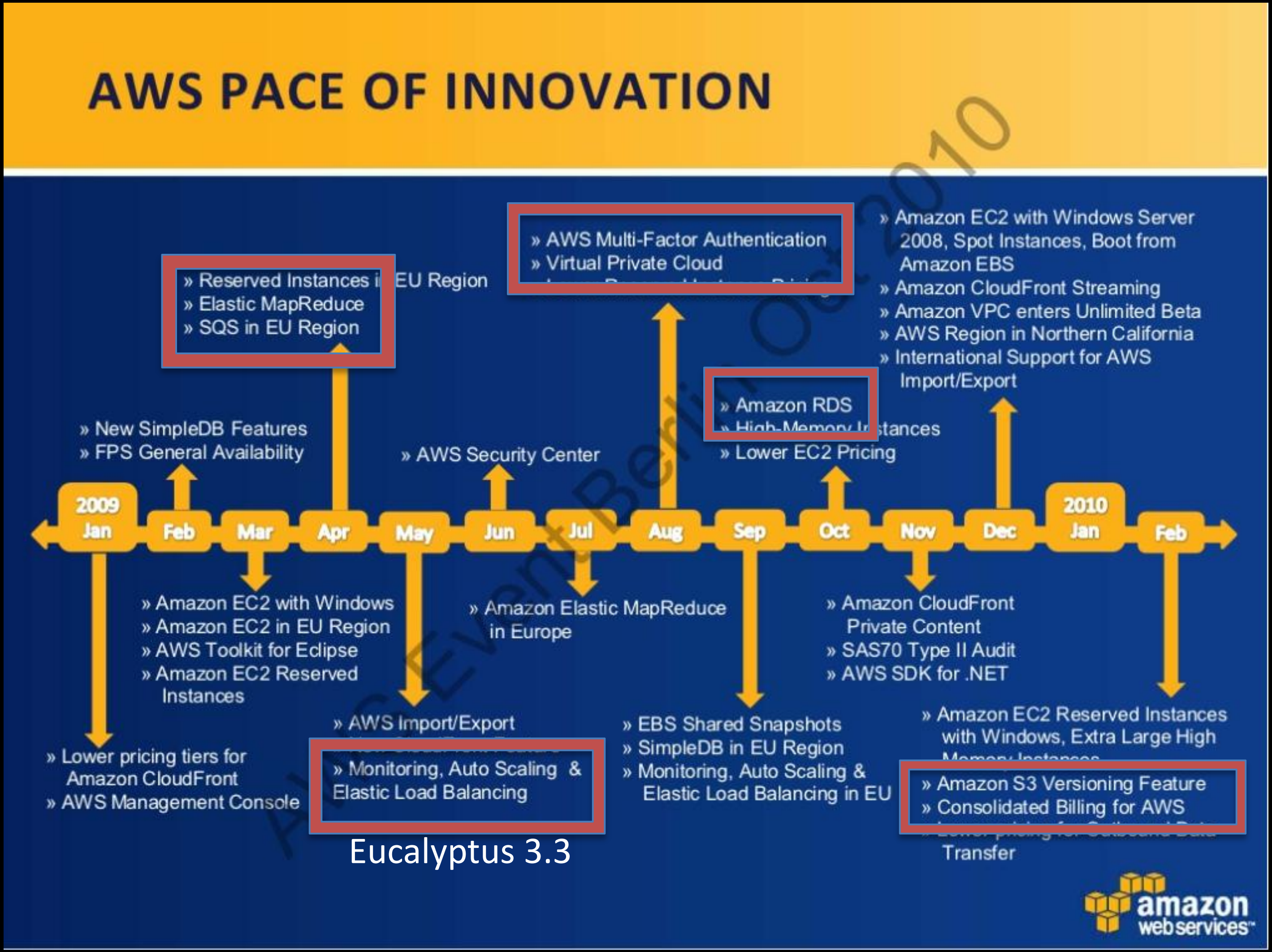
cloudstack

Some vendor interest
Many missing features
Bait and switch AWS API strategy

openstack™

AWS 2009

Baseline features needed to support NetflixOSS



AWS PACE OF INNOVATION

» Lower pricing tiers for Amazon CloudFront
» AWS Management Console

» New SimpleDB Features
» FPS General Availability

» Amazon EC2 with Windows
» Amazon EC2 in EU Region
» AWS Toolkit for Eclipse
» Amazon EC2 Reserved Instances

» Reserved Instances in EU Region
» Elastic MapReduce
» SQS in EU Region

» AWS Import/Export
» New CloudFront Feature
» Monitoring, Auto Scaling & Elastic Load Balancing

Eucalyptus 3.3

» AWS Security Center

» Amazon Elastic MapReduce in Europe

» AWS Multi-Factor Authentication
» Virtual Private Cloud
» Lower Reserved Instance Pricing

» EBS Shared Snapshots
» SimpleDB in EU Region
» Monitoring, Auto Scaling & Elastic Load Balancing in EU

» Amazon RDS
» High-Memory Instances
» Lower EC2 Pricing

» Amazon CloudFront Private Content
» SAS70 Type II Audit
» AWS SDK for .NET

» Amazon EC2 with Windows Server 2008, Spot Instances, Boot from Amazon EBS
» Amazon CloudFront Streaming
» Amazon VPC enters Unlimited Beta
» AWS Region in Northern California
» International Support for AWS Import/Export

» Amazon EC2 Reserved Instances with Windows, Extra Large High Memory Instances
» Amazon S3 Versioning Feature
» Consolidated Billing for AWS
» Lower pricing for Outbound Data Transfer

Netflix Cloud Prize

Boosting the @NetflixOSS Ecosystem

In 2012 Netflix Engineering won this..

ACADEMY OF TELEVISION ARTS & SC
2012 PRIMETIME EMMY AWARDS
ACHIEVEMENT IN ENGINEERING
STREAMING OF MOVIES AND TELEVISION
NETFLIX, INC.

We’d like to give out prizes too

But what for?
Contributions to NetflixOSS!
Shared under Apache license
Located on github

Judges choice award

Best example application mash-up

Best usability enhancement

Best portability enhancement

Best new monkey

Best new feature

Best datastore integration

Best contribution to code quality

Best contribution to operational tools

Best contribution to performance

How long do you have?

Entries open March 13th

Entries close September 15th

Six months…

Who can win?

Almost anyone, anywhere…

Except current or former Netflix or AWS employees

Who decides who wins?

Nominating Committee

Panel of Judges

Aino Corry
Program Chair for Qcon/GOTO

Simon Wardley
Strategist

Martin Fowler
Chief Scientist Thoughtworks

Werner Vogels
CTO Amazon

Joe Weinman
SVP Telx, Author "Cloudonomics"

Yury Izrailevsky
VP Cloud Netflix

What are Judges Looking For?

Eligible, Apache 2.0 licensed

Original and useful contribution to NetflixOSS

Code that successfully builds and passes a test suite

A large number of watchers, stars and forks on github

NetflixOSS project pull requests

Good code quality and structure

Documentation on how to build and run it

Evidence that code is in use by other projects, or is running in production

What do you win?

One winner in each of the 10 categories

Ticket and expenses to attend AWS Re:Invent 2013 in Las Vegas

A Trophy

$10,000 cash and $5,000 in AWS Credits

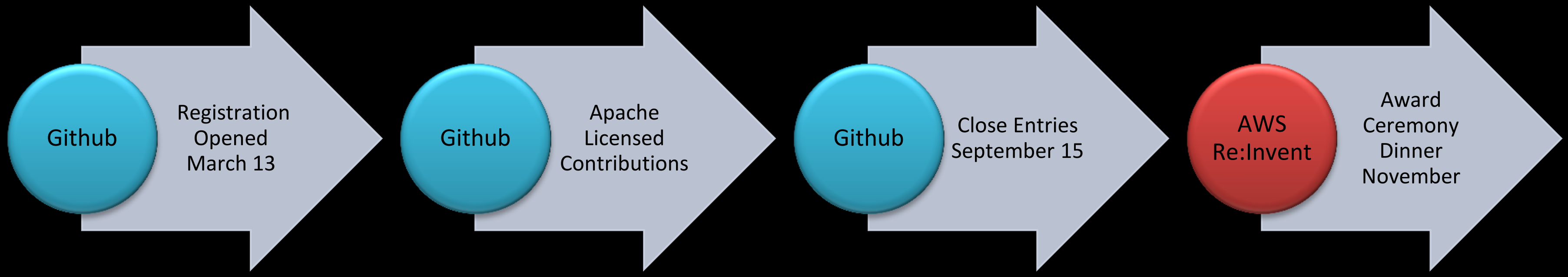
NETFLIX OSS CLOUD PRIZE

How do you enter?

Get a (free) github account
Fork github.com/netflix/cloud-prize
Send us your email address
Describe and build your entry

Twitter #cloudprize

NETFLIX OSS CLOUD PRIZE

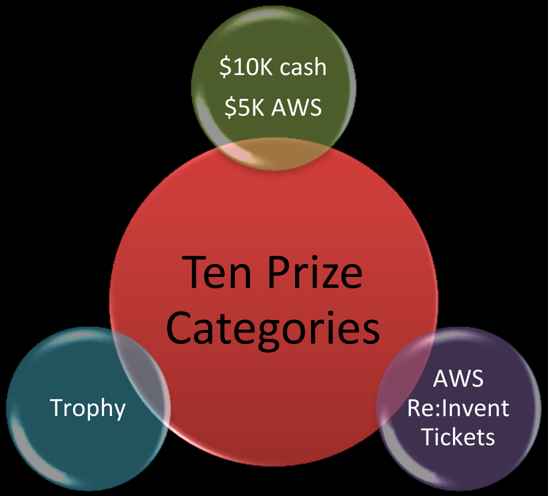


Github — Registration Opened March 13

Github — Apache Licensed Contributions

Github — Close Entries September 15

AWS Re:Invent — Award Ceremony Dinner November

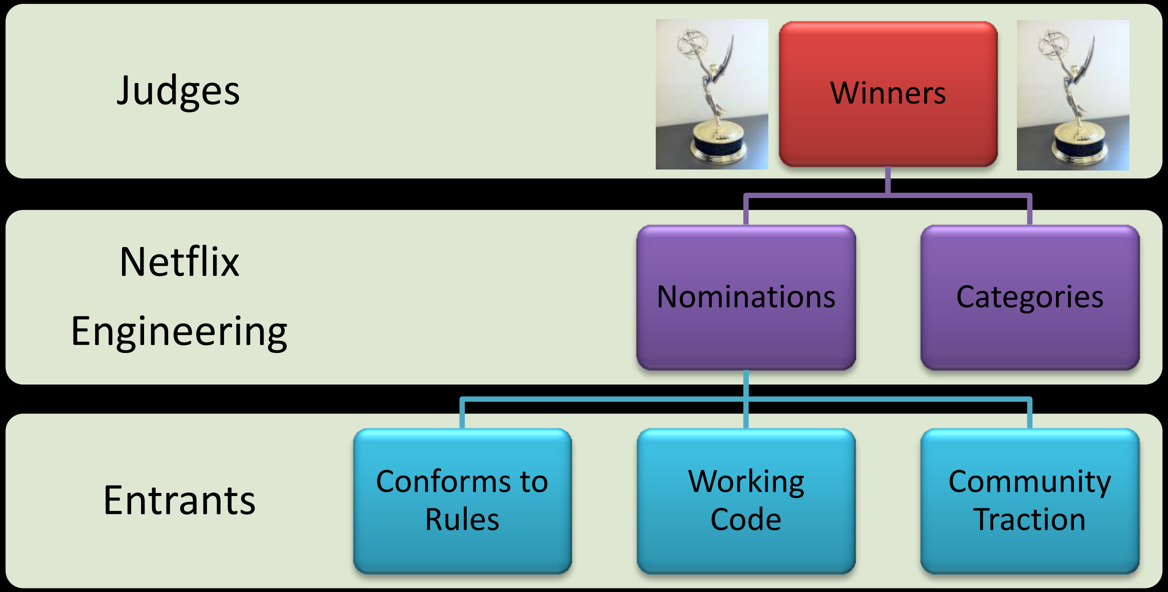


Ten Prize Categories

$10K cash
$5K AWS

Trophy

AWS Re:Invent Tickets



Judges — Winners

Netflix Engineering — Nominations, Categories

Entrants — Conforms to Rules, Working Code, Community Traction

NETFLIX OSS OPEN SOURCE SOFTWARE

Functionality and scale now, portability coming

Moving from parts to a platform in 2013

Netflix is fostering an ecosystem

Rapid Evolution - Low MTBIAMSH

(Mean Time Between Idea And Making Stuff Happen)

Takeaway

Netflix is making it easy for everyone to adopt Cloud Native patterns.

Open Source is not just the default, it's a strategic weapon.

http://netflix.github.com
http://techblog.netflix.com
http://slideshare.net/Netflix

http://www.linkedin.com/in/adriancockcroft

@adrianco #netflixcloud @NetflixOSS