Embracing Failure

Fault Injection and Service Resilience at Netflix

Josh Evans
Director of Operations Engineering, Netflix

Josh Evans

- 24 years in technology
- Tech support, Tools, Test Automation, IT & QA Management

- Time at Netflix ~15 years
- Ecommerce, streaming, tools, services, operations

- Current Role: Director of Operations Engineering

Netflix Ecosystem

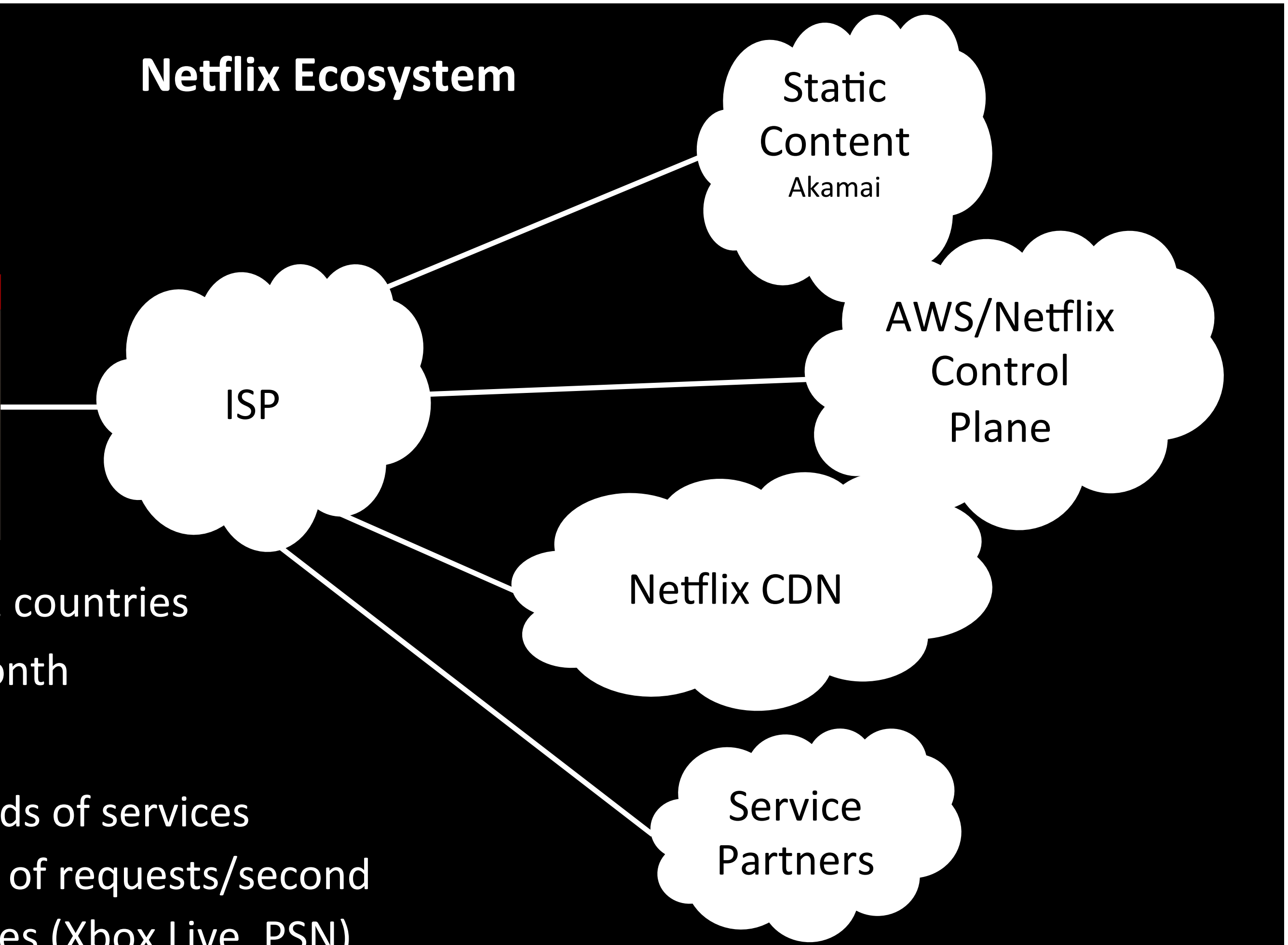
The Office

2005 TV-14 119 ep. HD

This Emmy-winning sitcom chronicles the daily foibles of office workers at the Dunder Mifflin paper company, where deluded boss Michael attempts to shepherd his disgruntled employees as a documentary film crew captures every wince-worthy moment.

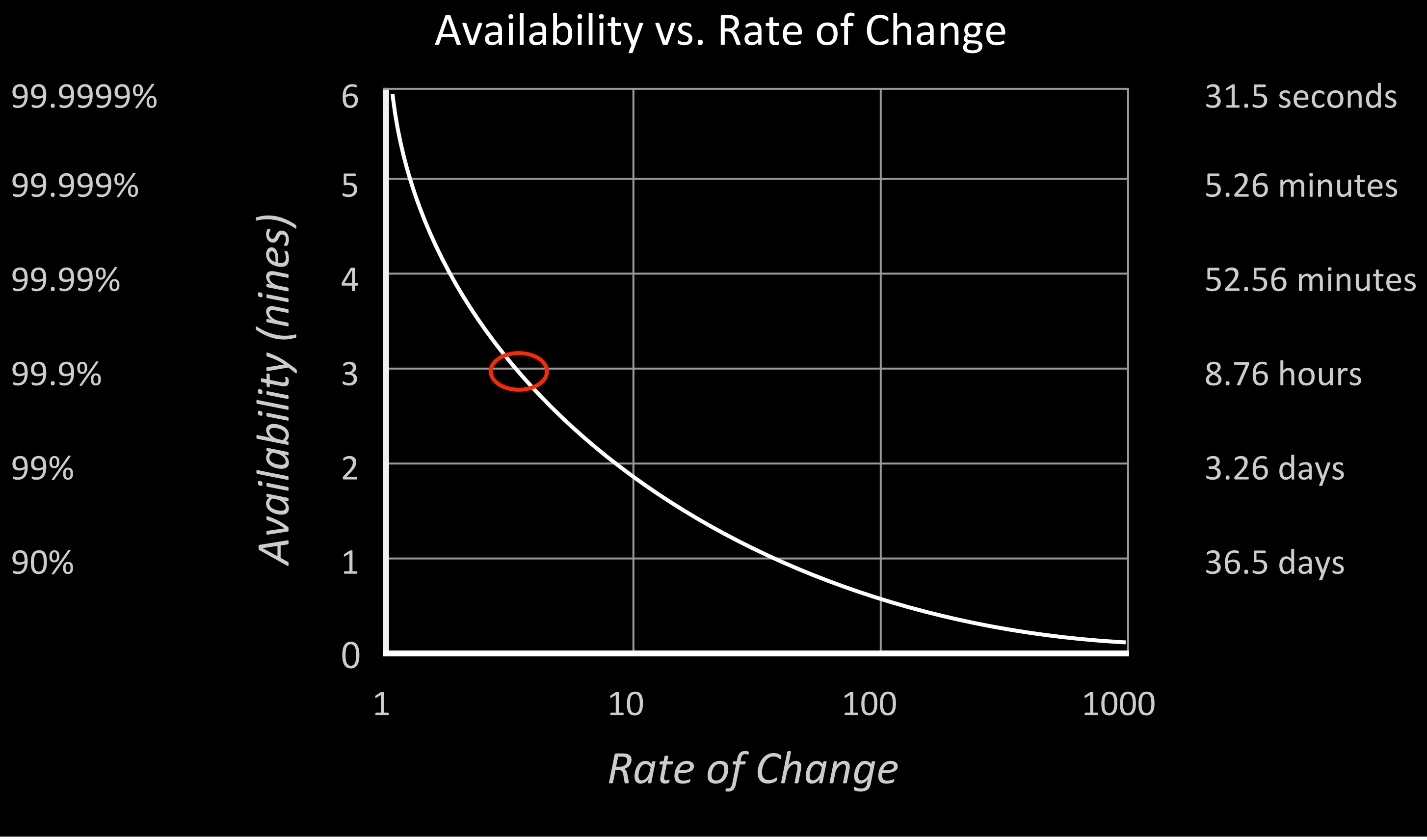
Steve Carell, John Krasinski, Jenna Fischer

Television, TV Sitcoms

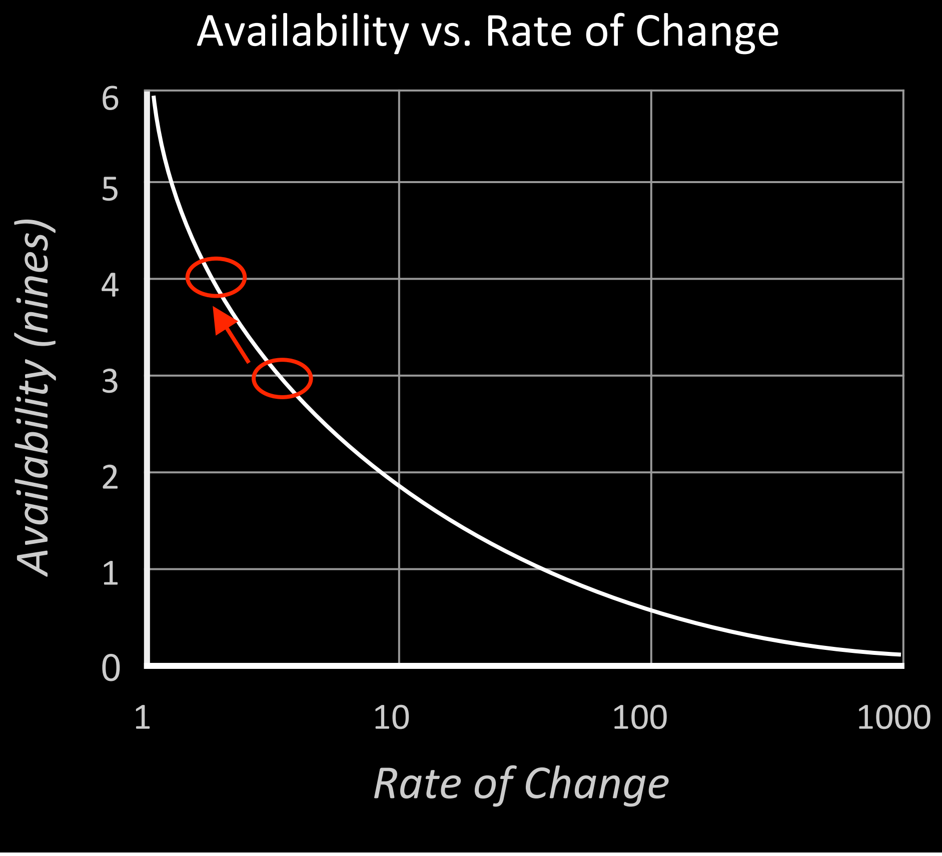


- 48 million members, 41 countries
- > 1 billion hours per month
- > 1000 device types
- 3 AWS Regions, hundreds of services
- Hundreds of thousands of requests/second
- Partner provided services (Xbox Live, PSN)
- CDN serving petabytes of data at terabits/second

Our Focus is on Quality and Velocity

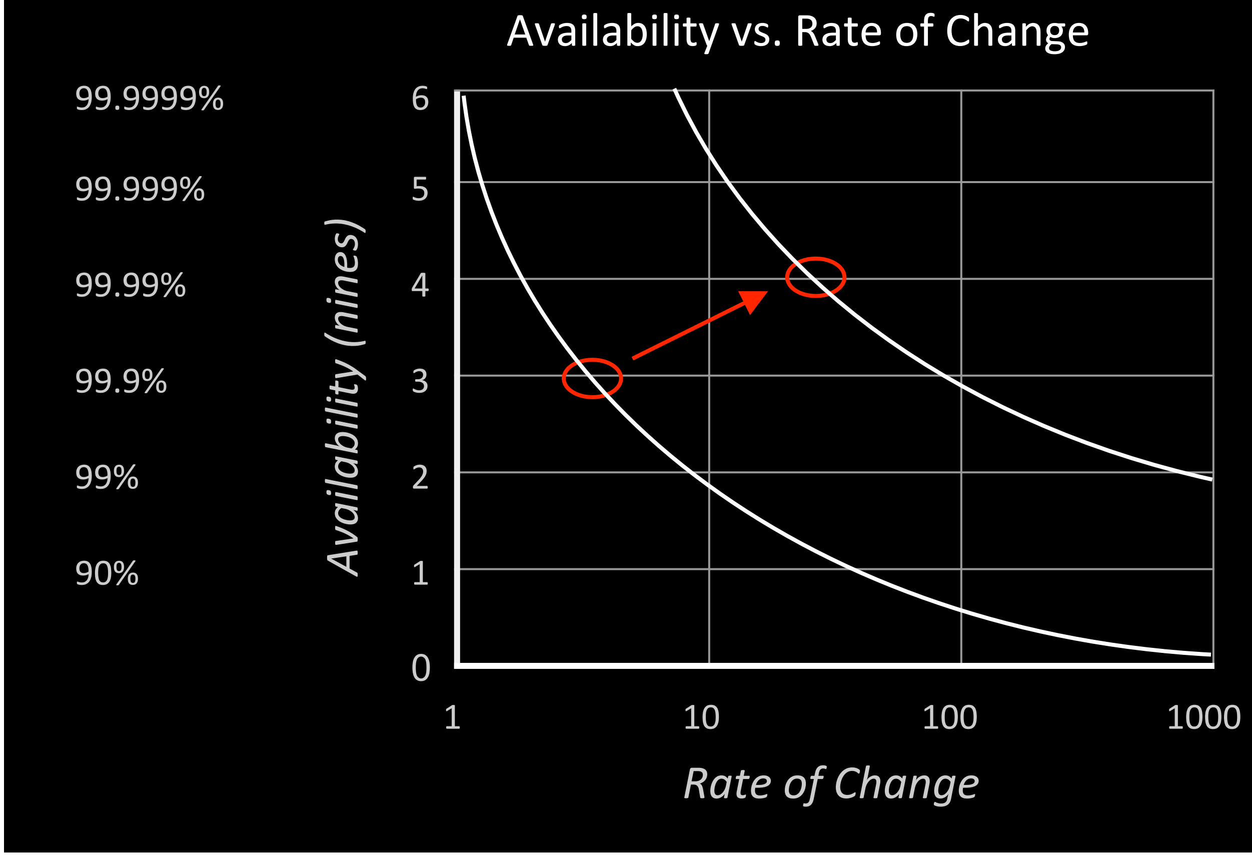


We Seek 99.99% Availability for Starts



99.9999%	31.5 seconds
99.999%	5.26 minutes
99.99%	52.56 minutes
99.9%	8.76 hours
99%	3.26 days
90%	36.5 days

Our goal is to shift the curve



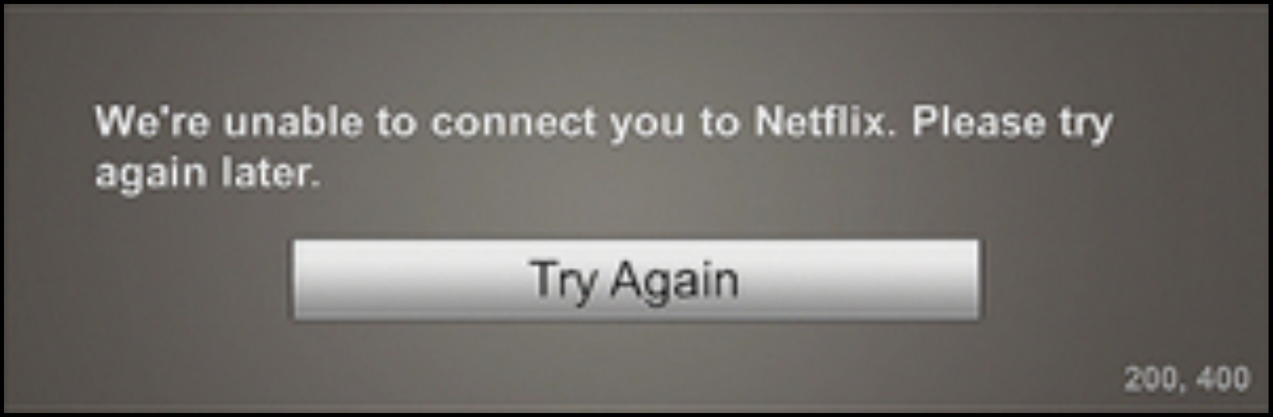
- Engineering
- Operations
- Best Practices

Continuous Improvement

Availability means that members can

- sign up
- activate a device
- browse
- watch

What keeps us up at night



We're unable to connect you to Netflix. Please try again later.

Failures happen all the time

- Disks fail
- Power goes, and your generator fails
- Software bugs
- Human error

- Failure is unavoidable

We design for failure

- Exception handling
- Auto-scaling clusters
- Redundancy
- Fault tolerance and isolation
- Fall-backs and degraded experiences
- Protect the customer from failures

Is that enough?

No

- How do we know if we’ve succeeded?
- Does the system work as designed?
- Is it as resilient as we believe?
- How do we prevent drifting into failure?

We test for failure

- Unit testing
- Integration testing
- Stress/load testing
- Simulation matrices

Testing increases confidence but…

is that enough?

Testing distributed systems is hard

- Massive, changing data sets
- Web-scale traffic
- Complex interactions and information flows
- Asynchronous requests
- 3rd party services
- All while innovating and improving our service

What if we regularly inject failures into our systems under controlled circumstances?

Embracing Failure in Production

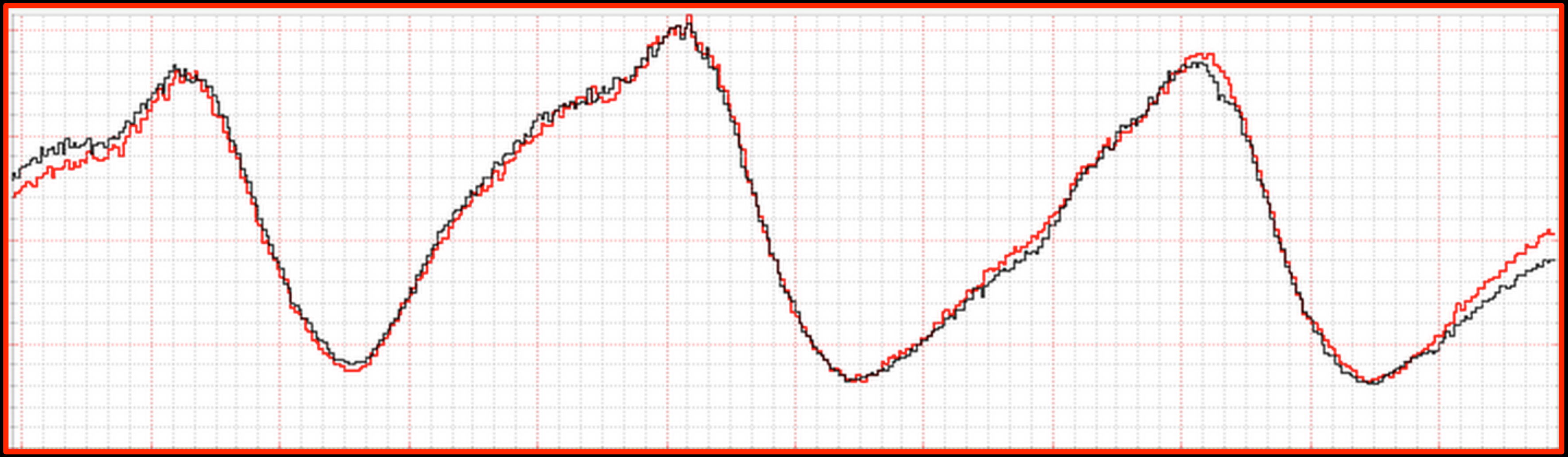
- Don’t wait for random failures
- Cause failure to validate resiliency
- Test design assumptions by stressing them
- Remove uncertainty by forcing failures regularly

NETFLIX

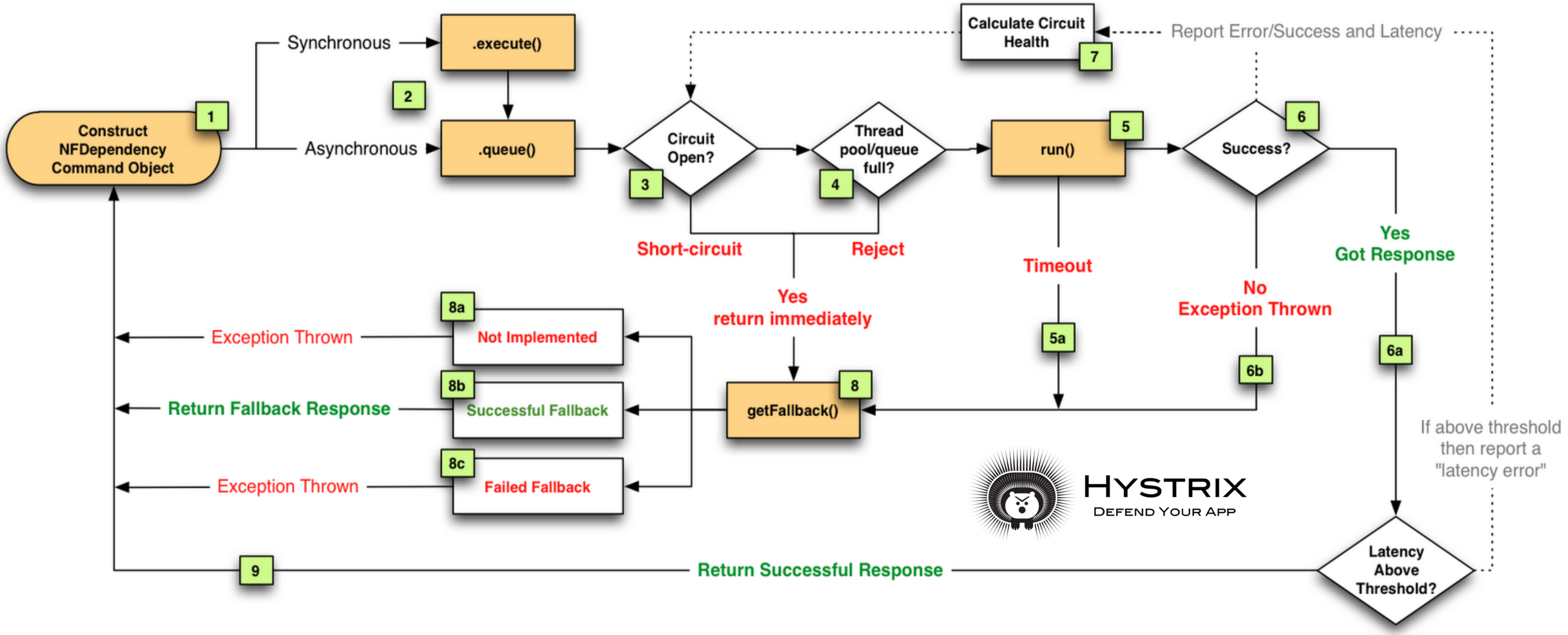
Two Key Concepts

Auto-Scaling

- Virtual instance clusters that scale and shrink with traffic
- Reactive and predictive mechanisms
- Auto-replacement of bad instances



Circuit Breakers



An Instance Fails

- Monkey loose in your DC
- Run during business hours
- Instances fail all the time

- What we learned
  - State is problematic
  - Auto-replacement works
  - Surviving a single instance failure is not enough

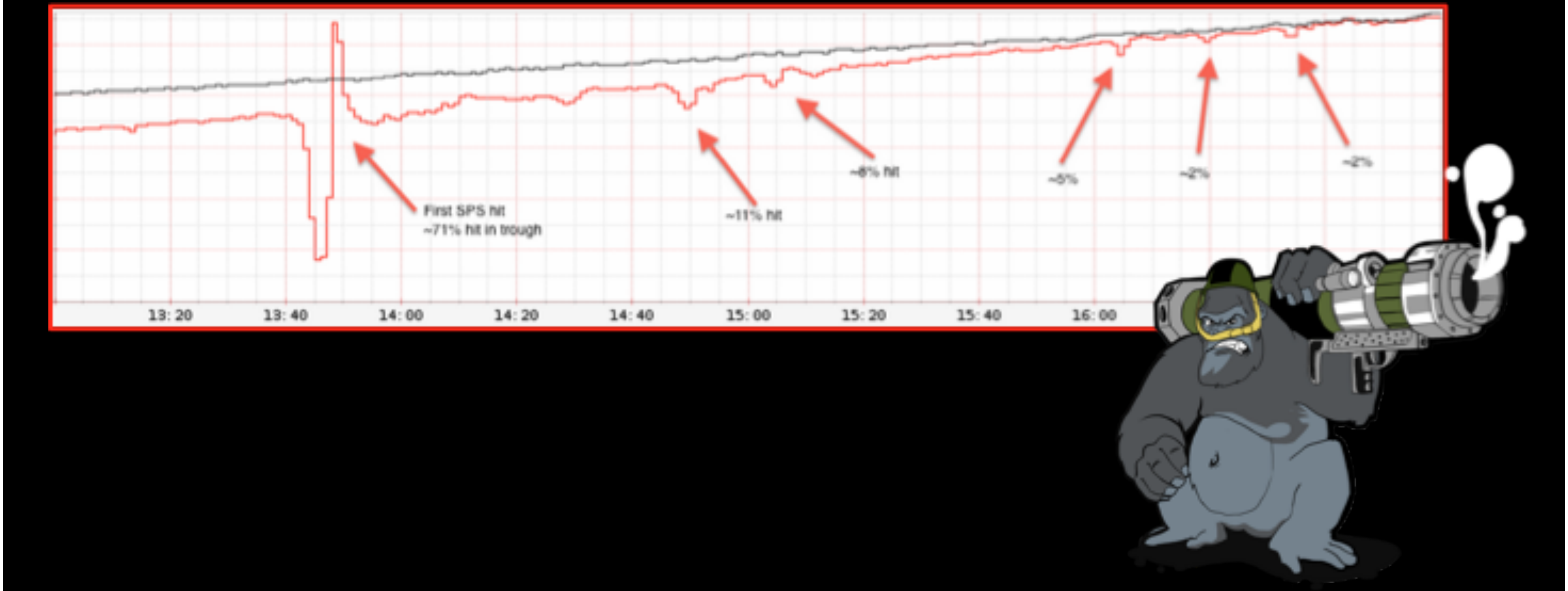
A Data Center Fails

Chaos Gorilla

Simulate an availability zone outage

- 3-zone configuration
- Eliminate one zone
- Ensure that others can handle the load and nothing breaks

What we encountered



What we learned

- Large scale events are hard to simulate
  - Hundreds of clusters, thousands of instances

- Rapidly shifting traffic is error prone
  - LBs must expire connections quickly
  - Lingering connections to caches must be addressed
  - Not all clusters pre-scaled for additional load

What we learned

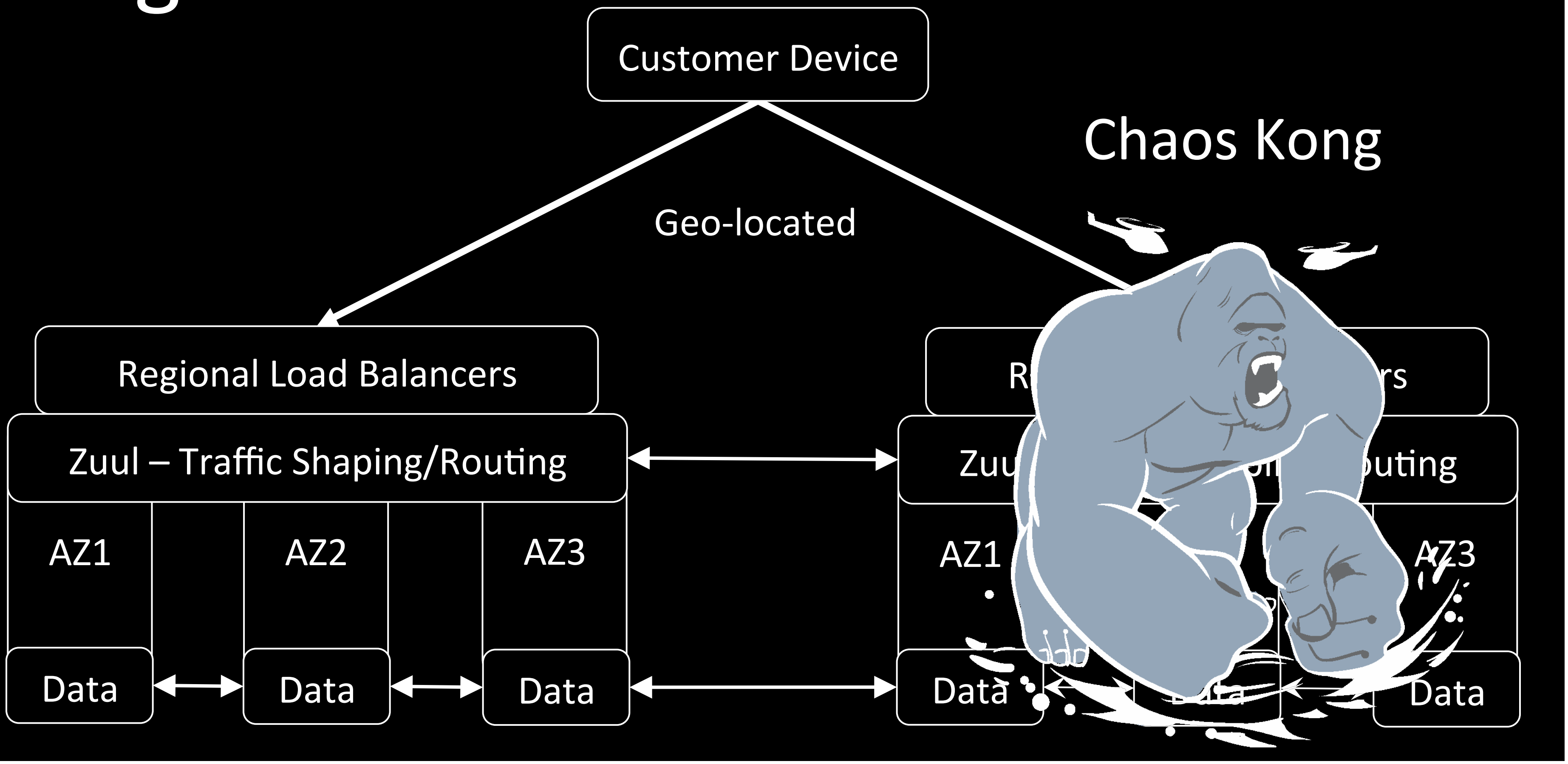
- Hidden assumptions & configurations
  - Some apps not configured for cross-zone calls
  - Mismatched timeouts – fallbacks prevented fail-over
  - REST client “preservation mode” prevented fail-over

- Cassandra works as expected

Regrouping

- From zone outage to zone evacuation
  - Carefully deregistered instances
  - Staged traffic shifts
- Resuming true outage simulations soon

Regions Fail



What we learned

- It works!
- Disable predictive auto-scaling
- Use instance counts from previous day

Room for Improvement

- Not a true regional outage simulation
  - Staged migration
  - No “split brain”

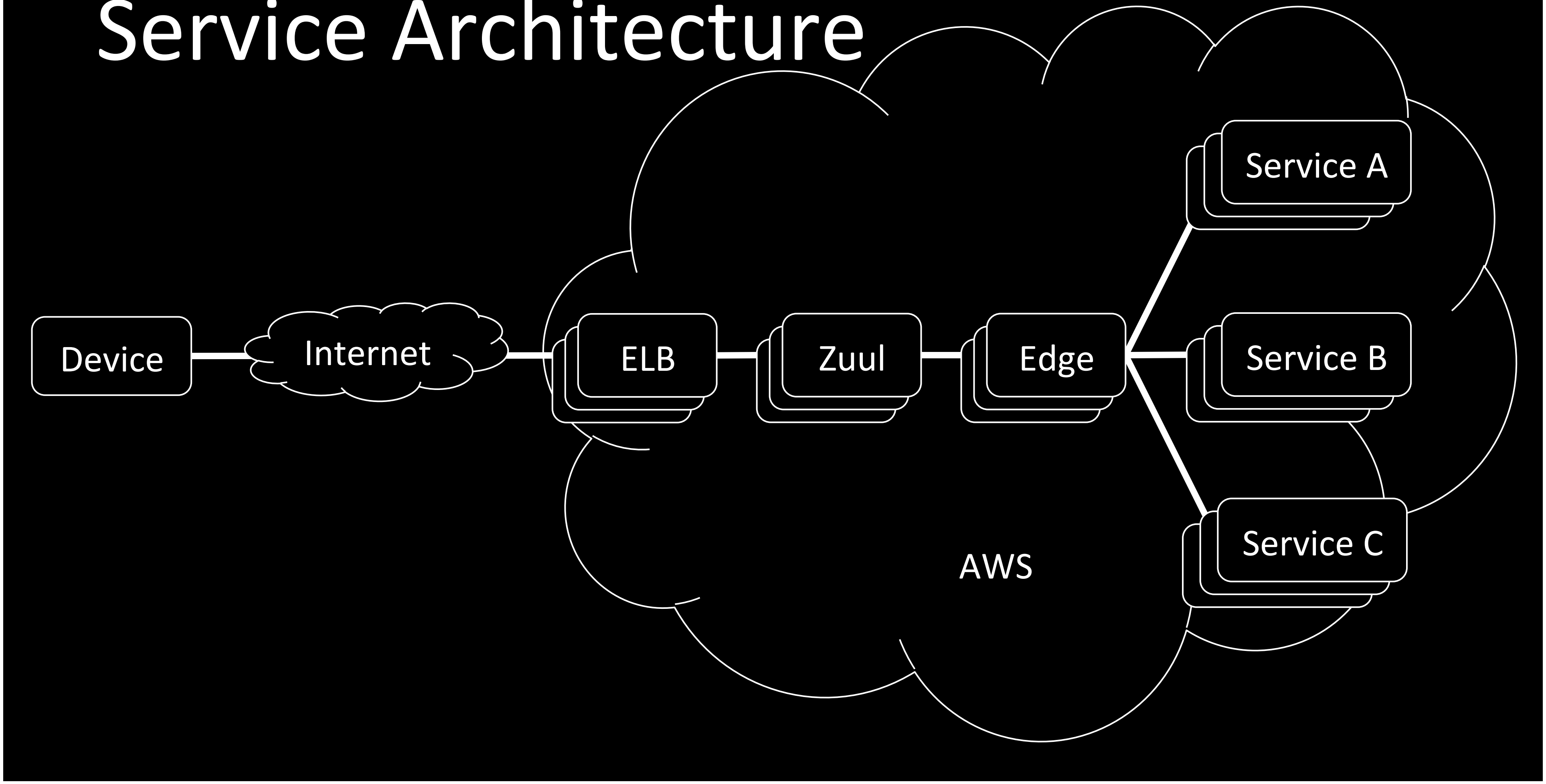
Not everything fails completely

Latency Monkey

Simulate latent service calls

- Inject arbitrary latency and errors at the service level
- Observe for effects

Service Architecture



Device

Internet

ELB

Zuul

Edge

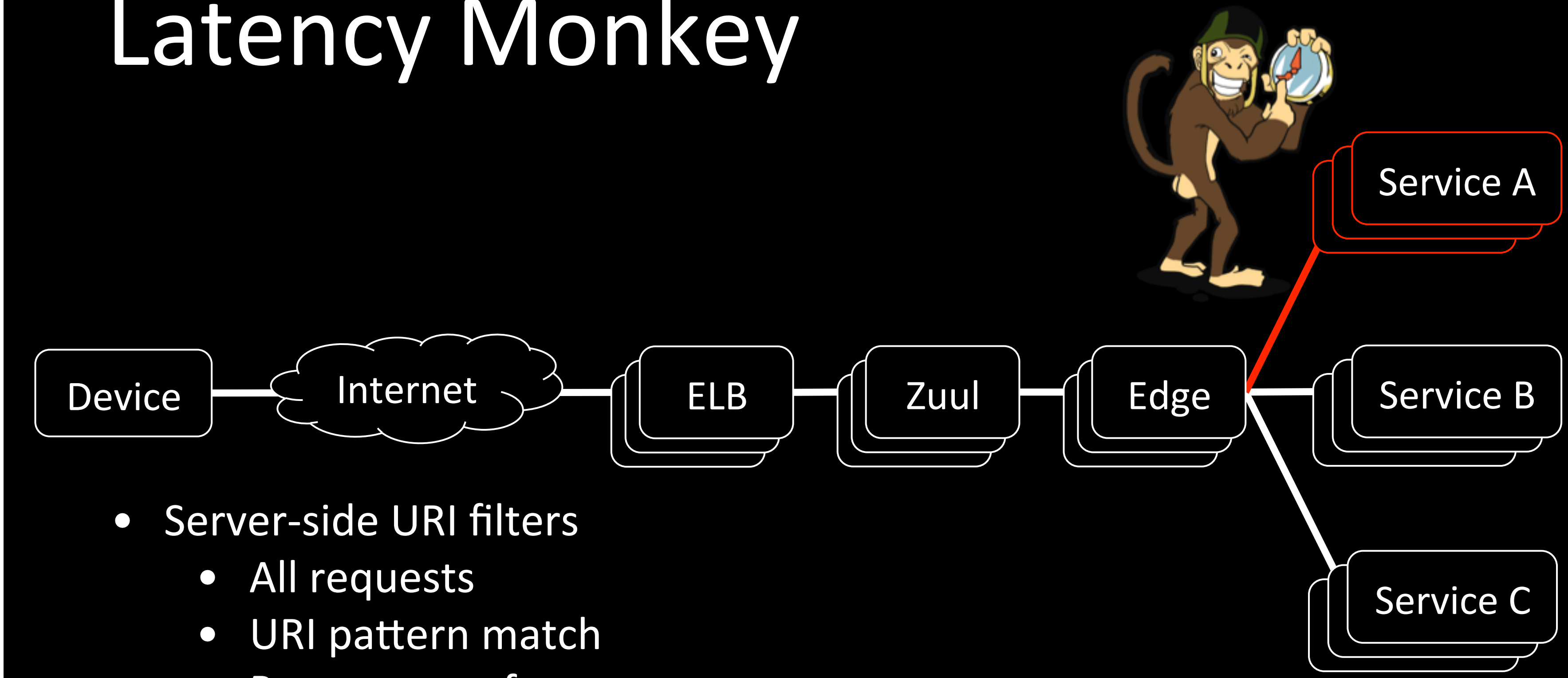
Service A

Service B

Service C

AWS

Latency Monkey



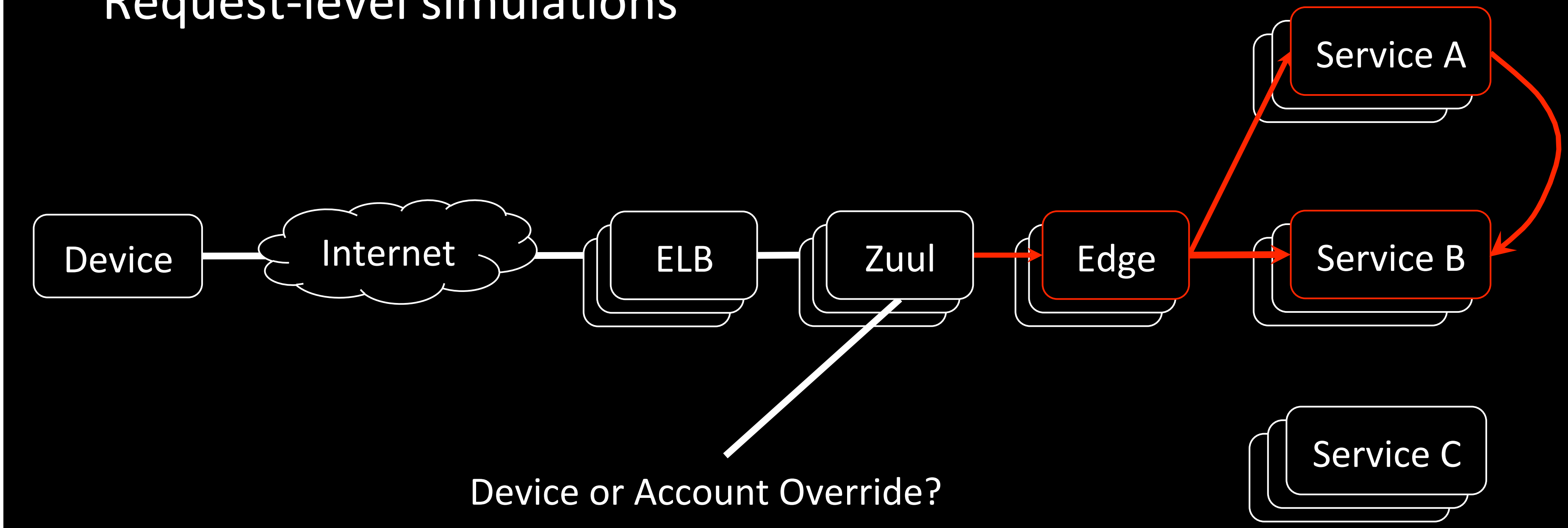
- Server-side URI filters
  - All requests
  - URI pattern match
  - Percentage of requests
- Arbitrary delays or responses

What we learned

- Startup resiliency is an issue
- Services owners don’t know all dependencies
- Fallbacks can fail too
- Second order effects not easily tested
- Dependencies change over time
- Holistic view is necessary
- Some teams opt out

Fault Injection Testing (FIT)

Request-level simulations

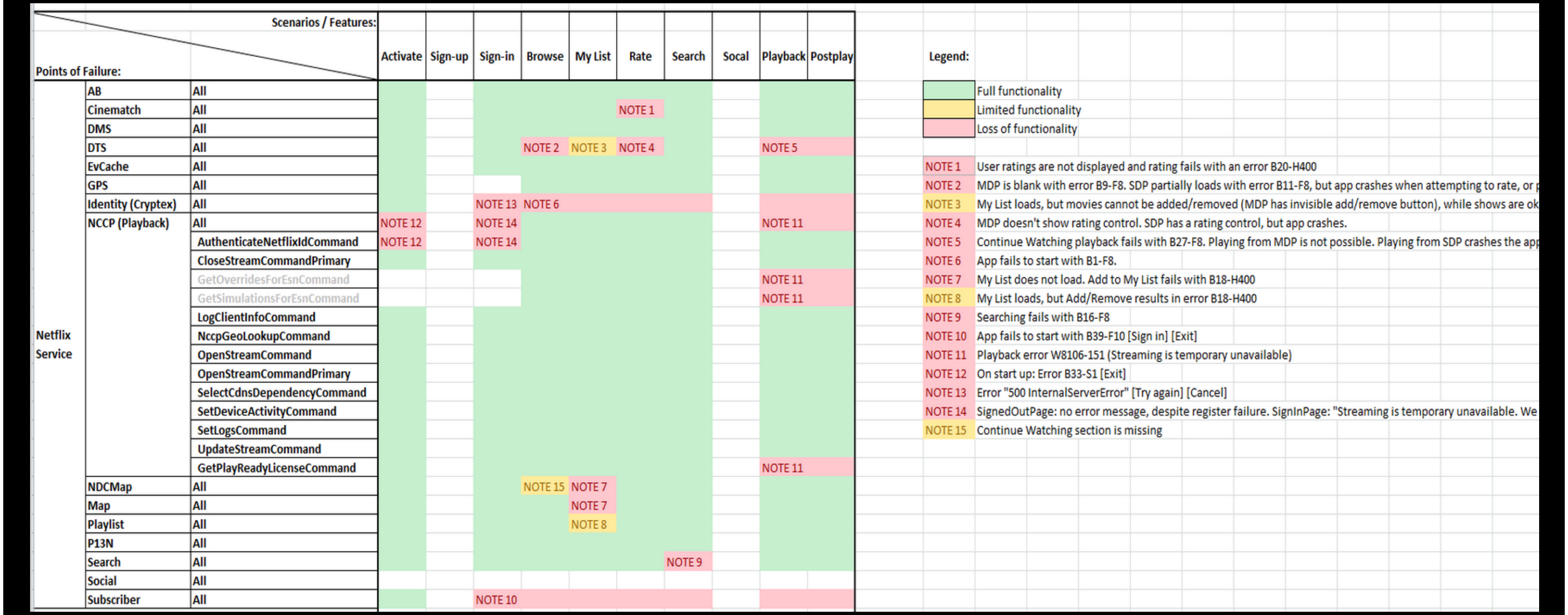


Device or Account Override?

Benefits

- Confidence building for latency monkey testing
- Continuous resilience testing in test and production
- Testing of minimum viable service, fallbacks
- Device resilience evaluation

Device Resiliency Matrix



Points of Failure:		Scenarios / Features:	Activate	Sign-up	Sign-in	Browse	My List	Rate	Search	Socal	Playback	Postplay
Netflix Service	AB	All	🟩		🟩	🟩	🟩	🟩	🟩		🟩	🟩
	Cinematch	All	🟩		🟩	🟩	🟩	🟥 NOTE 1	🟩		🟩	🟩
	DMS	All	🟩		🟩	🟩	🟩	🟩	🟩		🟩	🟩
	DTS	All	🟩		🟩	🟥 NOTE 2	🟨 NOTE 3	🟥 NOTE 4	🟩		🟥 NOTE 5	🟥
	EvCache	All	🟩		🟩	🟩	🟩	🟩	🟩		🟩	🟩
	GPS	All	🟩			🟩	🟩	🟩	🟩		🟩	🟩
	Identity (Cryptex)	All	🟩		🟥 NOTE 13	🟥 NOTE 6	🟥	🟥	🟥		🟥	🟥
	NCCP (Playback)	All	🟥 NOTE 12		🟥 NOTE 14	🟩	🟩	🟩	🟩		🟥 NOTE 11	🟥
		AuthenticateNetflixIdCommand	🟥 NOTE 12		🟥 NOTE 14	🟩	🟩	🟩	🟩		🟩	🟩
		CloseStreamCommandPrimary	🟩		🟩	🟩	🟩	🟩	🟩		🟩	🟩
		GetOverridesForEsnCommand				🟩	🟩	🟩	🟩		🟥 NOTE 11	🟥
		GetSimulationsForEsnCommand				🟩	🟩	🟩	🟩		🟥 NOTE 11	🟥
		LogClientInfoCommand	🟩		🟩	🟩	🟩	🟩	🟩		🟩	🟩
		NccpGeoLookupCommand	🟩		🟩	🟩	🟩	🟩	🟩		🟩	🟩
		OpenStreamCommand	🟩		🟩	🟩	🟩	🟩	🟩		🟩	🟩
		OpenStreamCommandPrimary	🟩		🟩	🟩	🟩	🟩	🟩		🟩	🟩
		SelectCdnsDependencyCommand	🟩		🟩	🟩	🟩	🟩	🟩		🟩	🟩
		SetDeviceActivityCommand	🟩		🟩	🟩	🟩	🟩	🟩		🟩	🟩
		SetLogsCommand	🟩		🟩	🟩	🟩	🟩	🟩		🟩	🟩
		UpdateStreamCommand	🟩		🟩	🟩	🟩	🟩	🟩		🟩	🟩
		GetPlayReadyLicenseCommand	🟩		🟩	🟩	🟩	🟩	🟩		🟥 NOTE 11	🟥
	NDCMap	All	🟩		🟩	🟨 NOTE 15	🟥 NOTE 7	🟩	🟩		🟩	🟩
	Map	All	🟩		🟩	🟩	🟥 NOTE 7	🟩	🟩		🟩	🟩
	Playlist	All	🟩		🟩	🟩	🟨 NOTE 8	🟩	🟩		🟩	🟩
	P13N	All	🟩		🟩	🟩	🟩	🟩	🟩		🟩	🟩
	Search	All	🟩		🟩	🟩	🟩	🟩	🟥 NOTE 9		🟩	🟩
	Social	All										
	Subscriber	All	🟩		🟥 NOTE 10	🟥	🟥	🟥	🟥		🟥	🟥

Legend:

🟩	Full functionality
🟨	Limited functionality
🟥	Loss of functionality

🟥 NOTE 1	User ratings are not displayed and rating fails with an error B20-H400
🟥 NOTE 2	MDP is blank with error B9-F8. SDP partially loads with error B11-F8, but app crashes when attempting to rate, or p
🟨 NOTE 3	My List loads, but movies cannot be added/removed (MDP has invisible add/remove button), while shows are ok
🟥 NOTE 4	MDP doesn't show rating control. SDP has a rating control, but app crashes.
🟥 NOTE 5	Continue Watching playback fails with B27-F8. Playing from MDP is not possible. Playing from SDP crashes the app
🟥 NOTE 6	App fails to start with B1-F8.
🟥 NOTE 7	My List does not load. Add to My List fails with B18-H400
🟨 NOTE 8	My List loads, but Add/Remove results in error B18-H400
🟥 NOTE 9	Searching fails with B16-F8
🟥 NOTE 10	App fails to start with B39-F10 [Sign in] [Exit]
🟥 NOTE 11	Playback error W8106-151 (Streaming is temporary unavailable)
🟥 NOTE 12	On start up: Error B33-S1 [Exit]
🟥 NOTE 13	Error "500 InternalServerError" [Try again] [Cancel]
🟥 NOTE 14	SignedOutPage: no error message, despite register failure. SignInPage: "Streaming is temporary unavailable. We
🟨 NOTE 15	Continue Watching section is missing

Is that it?

- Fault-injection isn't enough
  - Bad code/deployments
  - Configuration mishaps
  - Byzantine failures
  - Memory leaks
  - Performance degradation

A Multi-Faceted Approach

- Continuous Build, Delivery, Deployment
- Test Environments, Infrastructure, Coverage
- Automated Canary Analysis
- Staged Configuration Changes
- Crisis Response Tooling & Operations
- Real-time Analytics, Detection, Alerting
- Operational Insight - Dashboards, Reports
- Performance and Efficiency Engagements

It’s also about people and culture

Technical Culture

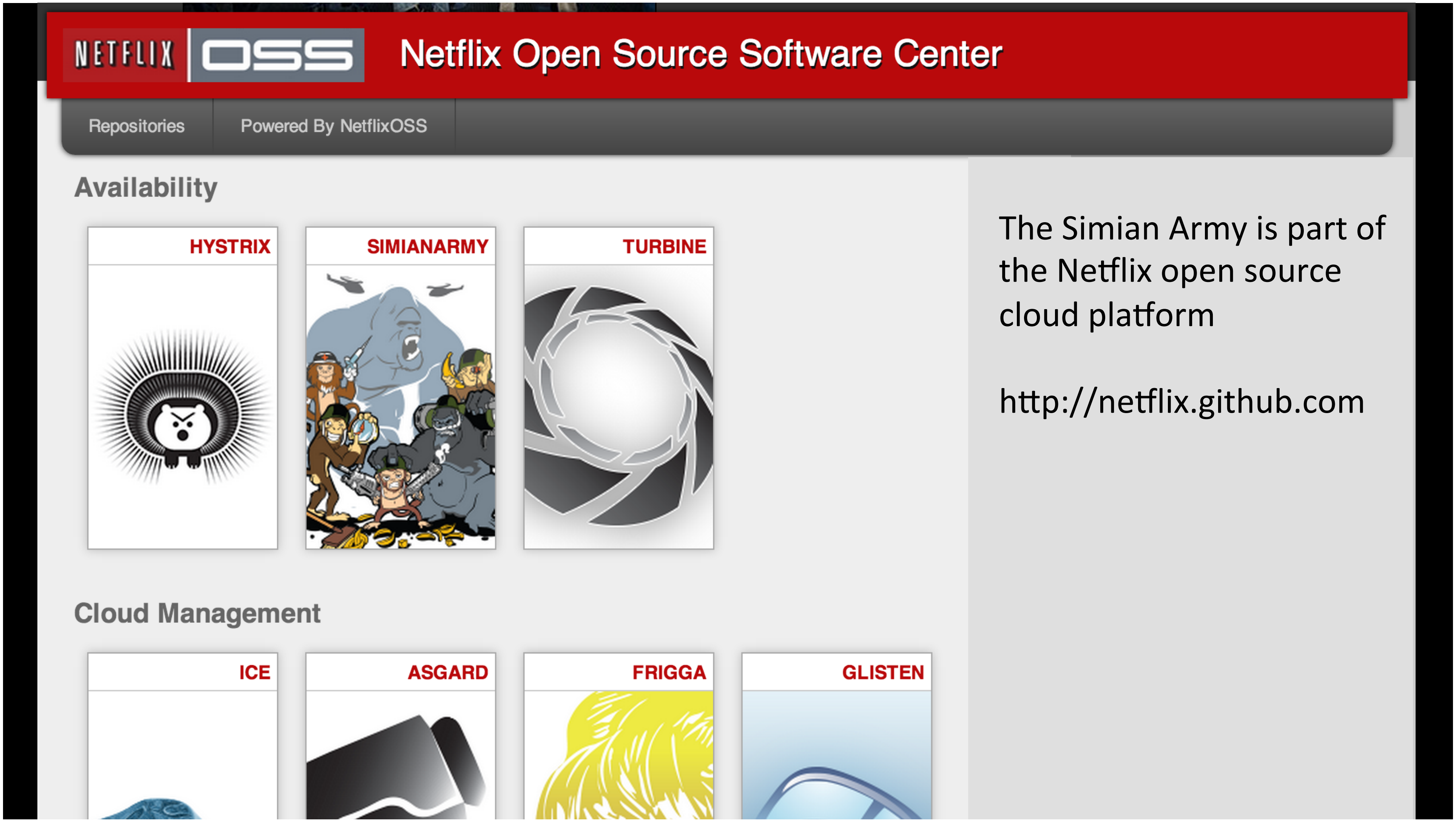
- You build it, you run it
- Each failure is an opportunity to learn
- Blameless incident reviews
- Commitment to continuous improvement

Context and Collaboration

Context engages partners

- Data and root causes
- Global vs. local
- Urgent vs. important
- Long term vision

Collaboration yields better solutions and buy-in



Netflix Open Source Software Center

Repositories | Powered By NetflixOSS

Availability

HYSTRIX | SIMIANARMY | TURBINE

Cloud Management

ICE | ASGARD | FRIGGA | GLISTEN

The Simian Army is part of the Netflix open source cloud platform

http://netflix.github.com

Operations Engineering

SHIFTING THE CURVE

Josh Evans
Director of Operations Engineering, Netflix
jevans@netflix.com, @josh_evans_nflx