I Don't Test Often ... But When I Do, I Test in Production

Gareth Bowles, Netflix

I Don't Test Often ...

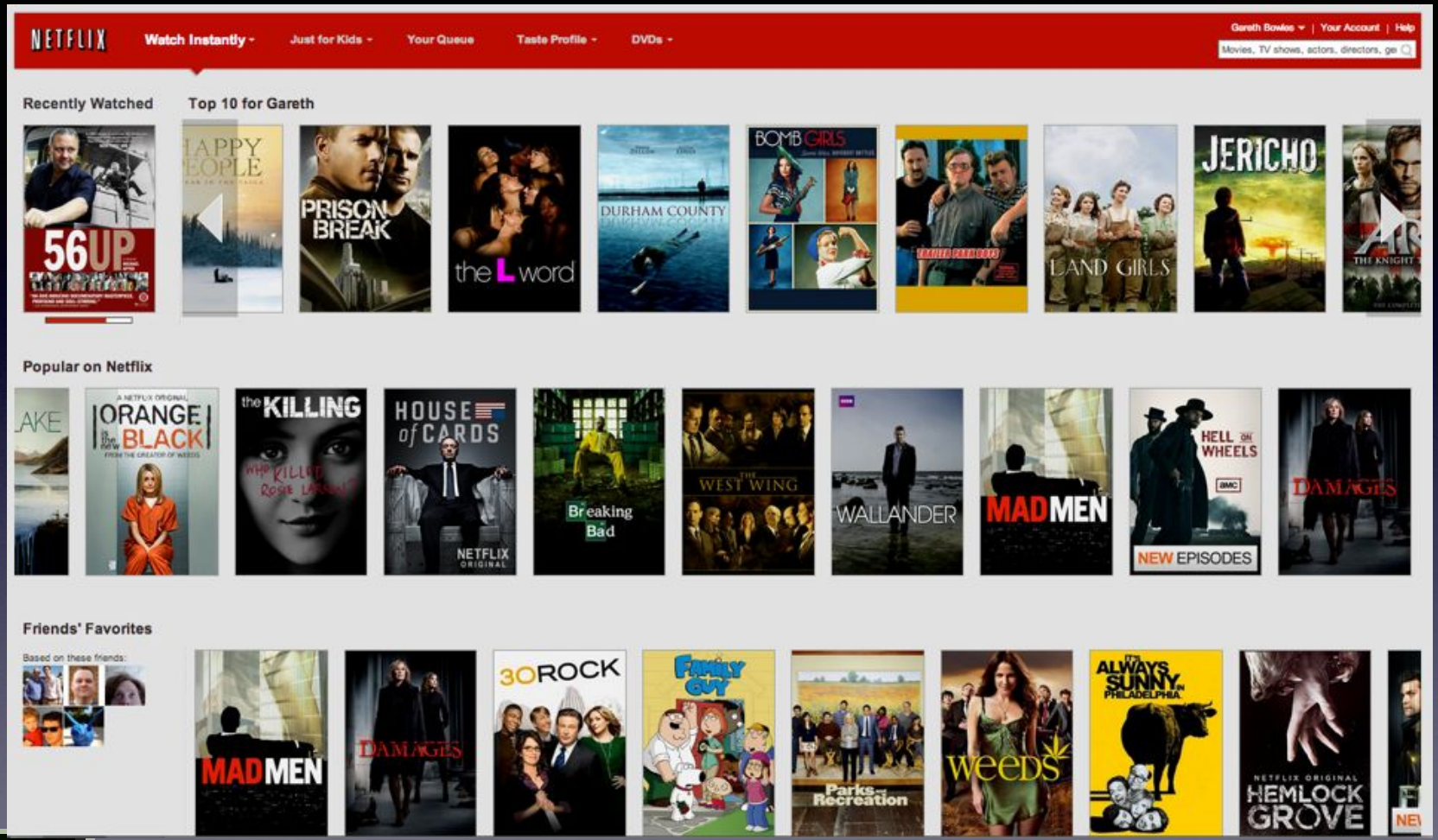
... But When I Do, I Test in Production

Building distributed systems is hard

Testing them exhaustively is even harder

Gareth Bowles

@garethbowles



@garethbowles

Netflix is the world's leading Internet television network with more than 50 million members in 50 countries enjoying more than one billion hours of TV shows and movies per month.

We account for up to 34% of downstream US internet traffic.

2B requests per day into the Netflix API

API

12B outbound requests per day to API dependencies

Personaliza-tion Engine

User Info

Movie Metadata

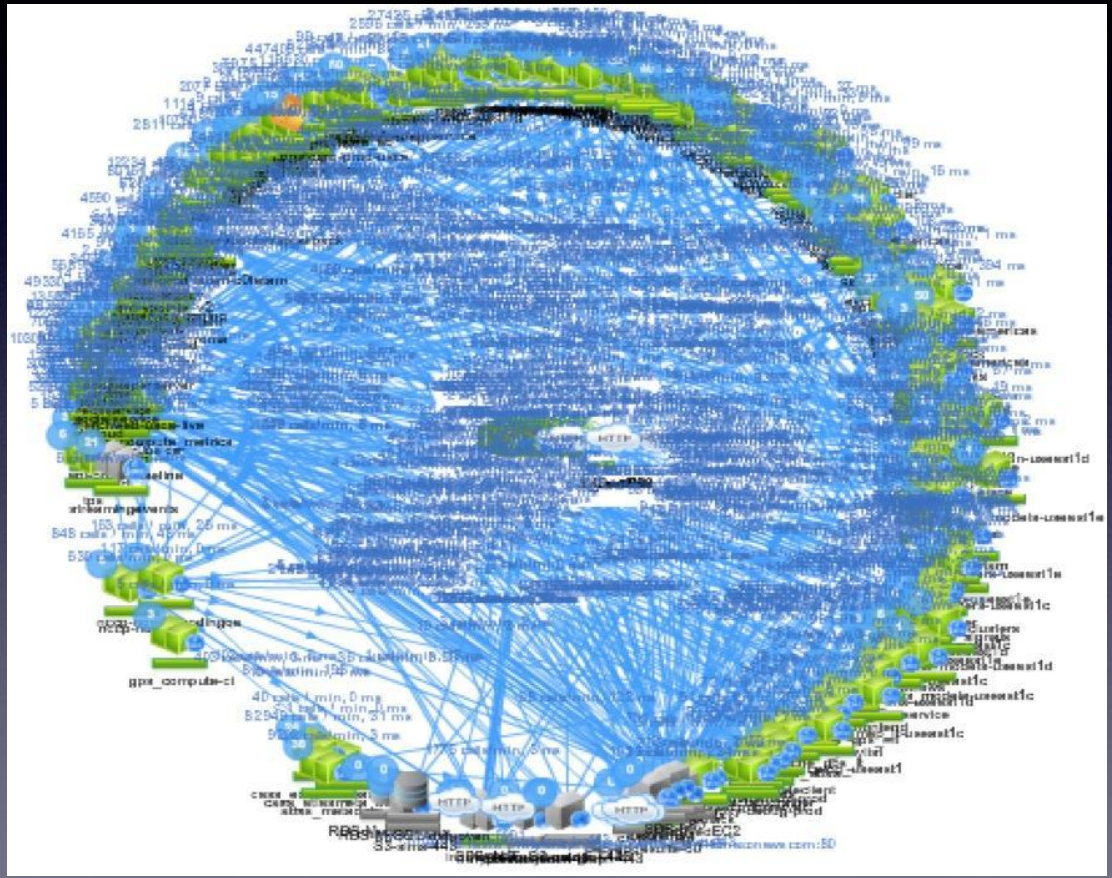
Movie Ratings

Similar Movies

Reviews

A/B Test Engine

A Complex Distributed System



Our Deployment Platform

What AWS Provides

- Machine Images (AMI)
- Instances (EC2)
- Elastic Load Balancers
- Security groups / Autoscaling groups
- Availability zones and regions

Our system has become so complex that...

...No (single) body knows how everything works.

How AWS Can Go Wrong -1

- Service goes down in one or more availability zones
- 6/29/12 - storm related power outage caused loss of EC2 and RDS instances in Eastern US
- https://gigaom.com/2012/06/29/some-of-amazon-web-services-are-down-again/

How AWS Can Go Wrong - 2

- Loss of service in an entire region
- 12/24/12 - operator error caused loss of multiple ELBs in Eastern US
- http://techblog.netflix.com/2012/12/a-closer-look-at-christmas-eve-outage.html

How AWS Can Go Wrong - 3

- Large number of instances get rebooted
- 9/25/14 to 9/30/14 - rolling reboot of 1000s of instances to patch a security bug
- http://techblog.netflix.com/2014/10/a-state-of-xen-chaos-monkey-cassandra.html

Our Goal is Availability

- Members can stream Netflix whenever they want
- New users can explore and sign up
- New members can activate their service and add devices

Netflix Culture:
Freedom & Responsibility

NETFLIX

1

http://www.slideshare.net/reed2001/culture-1798664

@garethbowles

Freedom and Responsibility

- Developers deploy when they want
- They also manage their own capacity and autoscaling
- And are on-call to fix anything that breaks at 3am!

How the heck do you test this stuff ?

Failure is All Around Us

- Disks fail
- Power goes out - and your backup generator fails
- Software bugs are introduced
- People make mistakes

Design to Avoid Failure

- Exception handling
- Redundancy
- Fallback or degraded experience (circuit breakers)
- But is it enough ?

It's Not Enough

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

More Testing !

- Unit
- Integration
- Stress
- Exhaustive testing to simulate all failure modes

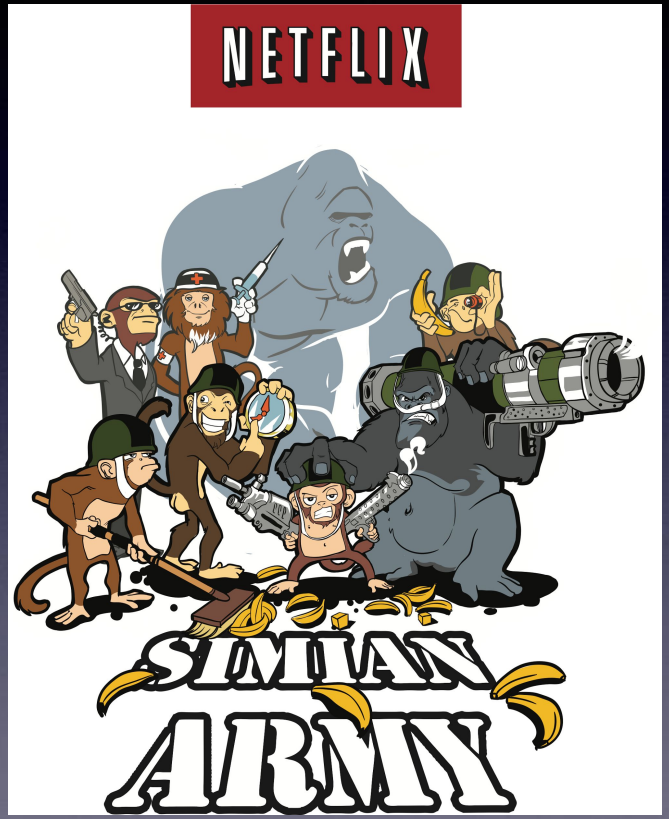
Exhaustive Testing ~ Impossible

- Massive, rapidly changing data sets
- Internet scale traffic
- Complex interaction and information flow
- Independently-controlled services
- All while innovating and building features

Another Way

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

Introducing the Simian Army



Chaos Monkey

Chaos Monkey

- The original Monkey (2009)
- Randomly terminates instances in a cluster
- Simulates failures inherent to running in the cloud
- During business hours
- Default for production services

What did we do once we were able to handle Chaos Monkey ?

Bring in bigger Monkeys !

Chaos Gorilla

Chaos Gorilla

- Simulate an Availability Zone becoming unavailable
- Validate multi-AZ redundancy
- Deploy to multiple AZs by default
- Run regularly (but not continually !)

Chaos Kong

Chaos Kong

- “One louder” than Chaos Gorilla
- Simulate an entire region outage
- Used to validate our “active-active” region strategy
- Traffic has to be switched to the new region
- Run once every few months

Latency Monkey

Latency Monkey

- Simulate degraded instances
- Ensure degradation doesn't affect other services
- Multiple scenarios: network, CPU, I/O, memory
- Validate that your service can handle degradation
- Find effects on other services, then validate that they can handle it too

Conformity Monkey

Conformity Monkey

- Apply a set of conformity rules to all instances
- Notify owners with a list of instances and problems
- Example rules
  - Standard security groups not applied
  - Instance age is too old
  - No health check URL

Some More Monkeys

- Janitor Monkey - clean up unused resources

- Security Monkey - analyze, audit and notify on AWS security profile changes

- Howler Monkey - warn if we're reaching resource limits

Try it out !

- Open sourced and available at https://github.com/Netflix/SimianArmy and https://github.com/Netflix/security_monkey
- Chaos, Conformity, Janitor and Security available now; more to come
- Ported to VMWare

What's Next ?

- Finer grained control
- New failure modes
- Make chaos testing as well-understood as regular regression testing

A message from the owners

"Use Chaos Monkey to induce various kinds of failures in a controlled environment."

AWS blog post following the mass instance reboot in Sep 2014: http://aws.amazon.com/blogs/aws/ec2-maintenance-update-2/

Production Code Coverage

If you don’t run code coverage analysis in prod, you’re not doing it properly.

How We Do It

- Use Cobertura as it counts how many times each LOC is executed
- Easy to enable - Cobertura JARs included in our base AMI, set a flag to add them to Tomcat's classpath
- Enable on a single instance
- Very low performance hit

What We Get

- Real time code usage patterns
- Focus testing by prioritizing frequently executed paths with low test coverage
- Identify dead code that can be removed

Canary Deployments

Canaries

- Push changes to a small number of instances
- Use Asgard for red / black push
- Monitor closely to detect regressions
- Automated canary analysis
- Automatically cancel deployment if problems occur

Thank You !

Email: gbowles@{gmail,netflix}.com

Twitter: @garethbowles

Linkedin: www.linkedin.com/in/garethbowles