Monday, July 30, 2012

Chaos Monkey Released Into The Wild

By Cory Bennett and Ariel Tseitlin

We have found that the best defense against major unexpected failures is to fail often. By frequently causing failures, we force our services to be built in a way that is more resilient. We are excited to make a long-awaited announcement today that will help others who embrace this approach.

We have written about our Simian Army in the past and we are now proud to announce that **the source code for the founding member of the Simian Army, Chaos Monkey, is available to the community**.

Do you think your applications can handle a troop of mischievous monkeys loose in your infrastructure? Now you can find out.

What is Chaos Monkey?

Chaos Monkey is a service which runs in the Amazon Web Services (AWS) that seeks out Auto Scaling Groups (ASGs) and terminates instances (virtual machines) per group. The software design is flexible enough to work with other cloud providers or instance groupings and can be enhanced to add that support. The service has a configurable schedule that, by default, runs on non-holiday weekdays between 9am and 3pm. In most cases, we have designed our applications to continue working when an instance goes offline, but in those special cases that they don't, we want to make sure there are people around to resolve and learn from any problems. With this in mind, Chaos Monkey only runs within a limited set of hours with the intent that engineers will be alert and able to respond.

Why Run Chaos Monkey?

Failures happen and they inevitably happen when least desired or expected. If your application can't tolerate an instance failure would you rather find out by being paged at 3am or when you're in the office and have had your morning coffee? Even if you are confident that your architecture can tolerate an instance failure, are you sure it will still be able to next week? How about next month? Software is complex and dynamic and that "simple fix" you put in place last week could have undesired consequences. Do your traffic load balancers correctly detect and route requests around instances that go offline? Can you reliably rebuild your instances? Perhaps an engineer "quick patched" an instance last week and forgot to commit the changes to your source repository?

There are many failure scenarios that Chaos Monkey helps us detect. Over the last year Chaos Monkey has terminated over 65,000 instances running in our production and testing environments. Most of the time nobody notices, but we continue to find surprises caused by Chaos Monkey which allows us to isolate and resolve them so they don't happen again.

Auto Scaling Groups

The default instance groupings that Chaos uses for selection is Amazon's Auto Scaling Group (ASG). Within an ASG, Chaos Monkey will select an instance at random and terminate it. The ASG should detect the instance termination and automatically bring up a new, identically configured, instance. If you are not using Auto Scaling Groups that should be the first step to making your application handle these isolated instance failure scenarios. Check out Asgard to make deploying and managing ASGs easy. There are many great features for ASGs beyond replacing terminated instances, like enabling the use of Amazon's Elastic Load Balancers (ELBs) to distribute traffic to all instances in your application. Netflix has a best-practice where all instances should be run within an ASG and we have Janitor Monkey to remind us by terminating all instances not following this best-practice.

Configuration

Chaos Monkey allows for an Opt-In or an Opt-Out model. At Netflix, we use the Opt-Out model, so if an application owner does nothing, Chaos Monkey will be acting on their application. For your organization, you have the option to choose what is right for you. This allows you to "test the water" and try out Chaos Monkey on a specific application to see how it reacts. Not every application can trivially handle an instance going offline. Sometimes it takes a human to manually recover instances, perhaps exercising backups to bring them back. Ideally, engineers work towards making that process easier and faster and eventually automatic. For those applications, there is the ability to Opt-Out of Chaos Monkey. There is also a tunable "probability" that Chaos Monkey uses to control the chance of a termination. A probability of 1 (or 100%) will terminate one instance per day per ASG. If instance recovery is difficult and you only want a termination weekly, you can reduce the probability to 0.2 or 20% (daily is 100%, it runs 5 work days per week, so weekly is 20%). Note that this is still a probability and only meaningful when sampled multiple times. With a 20% probability, Chaos Monkey would terminate one instance a week **on average**. In practice, it might be 2 days in a row

Links

Netflix US & Canada Blog
Netflix America Latina Blog
Netflix Brasil Blog
Netflix Benelux Blog
Netflix DACH Blog
Netflix France Blog
Netflix Nordics Blog
Netflix UK & Ireland Blog
Netflix ISP Speed Index
Open positions at Netflix
Netflix Website
Facebook Netflix Page
Netflix UI Engineering
RSS Feed

About the Netflix Tech Blog

This is a Netflix blog focused on technology and technology issues. We'll share our perspectives, decisions and challenges regarding the software we build and use to create the Netflix service.

Blog Archive

► 2016 (51)
► 2015 (50)
► 2014 (37)
► 2013 (52)
▼ 2012 (37)
► December (6)
► November (3)
► October (2)
► September (2)
▼ July (6)
Chaos Monkey Released Into The Wild
Benchmarking High Performance I/O with SSD for Cas...
Enabling Support for IPv6
Open Source at Netflix
Embracing the Differences : Inside the Netflix API...
Lessons Netflix Learned From The AWS Storm
► June (5)
► May (1)
► April (2)

followed by 2 weeks of no terminations, but given a large enough sample it will terminate weekly on average. For an environment as large as Netflix, the configuration can get a bit tricky to manage and for this we have developed a dashboard to help that we hope to open source soon. You can read more about how to configure Chaos Monkey on the documentation wiki.

REST

Currently, there is a simple REST interface that allows you to query Chaos Monkey termination events. We keep records of what was terminated and when, so if something disappears, you can see if Chaos Monkey was responsible. You could use this API to get notifications of terminations, but we encourage you to use a more general application monitoring solution like servo to discover what is happening to your applications at runtime.

Costs

The termination events are stored in an Amazon SimpleDB table by default. There could be associated costs with Amazon SimpleDB but the activity of Chaos Monkey should be small enough to fall within Amazon's Free Usage Tier. Ultimately the costs associated with running Chaos Monkey are your responsibility.

Cost references: http://aws.amazon.com/simpledb/pricing/

More Monkey Business

We have a long line of simians waiting to be released. The next likely candidate will be Janitor Monkey which helps keep your environment tidy and your costs down. Stay tuned for more announcements.

If building tools to automate the operations and improve the reliability of the cloud sounds exciting, we're always looking for new members to join the team. Take a look at jobs.netflix.com for current openings or contact @atseitlin.

Chaos Monkey

- Home Page
- Quick Start Guide
- REST API
- Source Code

Netflix Cloud Platform

- The Netflix Simian Army
- Asgard Web Based Cloud Management
- 5 Lessons We've Learned Using AWS
- Netflix Open Source Projects
- Auto Scaling in the Amazon Cloud
- Servo (Publish application metrics for auto scaling)
- @NetflixOSS Twitter Feed

Amazon Web Services

- Auto Scaling
- Elastic Load Balancing (ELB)
- SimpleDB Costs

59 Comments Sort by Oldest

Add a comment...

Hans Granqvist
Another great tool!
Like · Reply · 5 · Jul 30, 2012 6:48pm

> **John Haren**
> That's what SHE said! (sorry, had to do it).
> Like · Reply · 12 · Jul 30, 2012 7:51pm

Steven Hepting ·
IOS Developer at Twitter

- ► March (2)
- ► February (4)
- ► January (4)
- ► 2011 (17)
- ► 2010 (8)

Labels

- "cloud architecture" (4)
- A/B Testing (3)
- accelerated compositing (2)
- adwords (1)
- Aegisthus (1)
- AES-GCM (1)
- AES-NI (1)
- algorithms (6)
- aminator (2)
- analytics (5)
- Android (3)
- angular (1)
- animations (1)
- Apache Mesos (1)
- api (16)
- appender (1)
- AppsAndDevices (3)
- Archaius (2)
- architectural design (1)
- architecture (2)
- artwork (1)
- Asgard (1)
- Astyanax (4)
- authentication (1)
- automation (4)
- autoscaling (3)
- availability (4)
- AWS (31)
- bake (1)
- benchmark (3)
- big data (11)
- billing (4)
- Blitz4j (1)
- build (4)
- Cable (1)
- caching (5)
- Cassandra (15)
- chaos (1)
- chaos engineering (2)
- chaos monkey (6)
- chukwa (1)
- ci (1)

Great stuff!

Like · Reply · Jul 30, 2012 10:16pm

Jason Riedel ·
Technical Director & Architect at Symantec

Awesome!

Like · Reply · Jul 30, 2012 10:21pm

Matt Geis ·
Principal Member of Technical Staff at Salesforce

Take your stinking paws off me, you damned dirty ape!

Like · Reply · 2 · Jul 30, 2012 11:00pm

> **Beno Bernard Lloyd Hwang** ·
> Mechanical Engineer at Applied Materials
>
> i'm not too sure what your post is about, but your comment reminded me of the _planet_of_the_apes_ musical from _the_simpsons_: http://www.youtube.com/watch?v=gMnG3gOqigE
>
> Like · Reply · Jul 30, 2012 11:54pm
>
> **Alex McConnell** ·
> Works at Millbank Holdings
>
> Probably because it's from the actual film Planet of the Apes...
>
> Like · Reply · 1 · Jul 31, 2012 12:48pm

Mike Alatortsev ·
Stamford (Connecticut)

"Chaos Monkey is a service which runs in the Amazon Web Services (AWS)..." Except, AWS already is the chaos monkey.

Like · Reply · 14 · Jul 30, 2012 11:33pm

Max Allan ·
Chippenham, Wiltshire

I'm looking forward to seeing "Destruction gorilla" tested on someone else's infrastructure. It will delete your ASGs, ELBs, S3 storage pool. Or sometimes removes your AWS account entirely. 😉

Like · Reply · 6 · Jul 31, 2012 12:21pm

me

nice way to test DR...with monkeys!

Like · Reply · 1 · Jul 31, 2012 3:39pm

Dr. Branden R. Williams

I love this! I wrote about your Chaos Monkey last year in a blog post and related it to information security. Several of us in that community are looking to take the concept to that arena as well! Blog post here: https://www.brandenwilliams.com/.../where-is-your-chaos....

Like · Reply · 2 · Jul 31, 2012 3:39pm

Satender Saroha ·
Sunnyvale (California)

Hey Cory, nice work! and Congrats on your new gig.

Like · Reply · Jul 31, 2012 5:58pm

Matt Phelps ·
Administrador de sistemas at Harvard-Smithsonian Center for Astrophysics

I appreciate your efforts in open source. Can we get your main product to run on THE open source OS, Linux please? Using Silverlight for Netflix streaming is most annoying.

Like · Reply · 3 · Aug 7, 2012 2:12pm

Load 10 more comments

classloaders (1)
Clojure (1)
cloud (28)
cloud architecture (19)
cloud prize (3)
CO2 (1)
collection (1)
complex event processing (1)
computer vision (2)
concurrency (1)
configuration (2)
configuration management (2)
conformity monkey (1)
content delivery (2)
content metadata (2)
content platform engineering (4)
content quality (1)
continue watching (1)
continuous delivery (4)
coordination (2)
cost management (1)
crypto (1)
Cryptography (2)
CSS (2)
CUDA (1)
DaRE (1)
dart (1)
data (1)
data migration (1)
data pipeline (5)
data science (8)
data visualization (1)
database (7)
DataStax (2)
deadlock (1)
deep learning (1)
Denominator (2)
dependency injection (1)
device (3)
device proliferation (1)
devops (3)
distributed (12)
DNS (1)
Docker (1)
Dockerhub (1)
DSL (1)
Dyn (1)
DynECT (1)
Dynomite (3)

Facebook Comments Plugin

Posted by Ariel Tseitlin at 7:59 AM

G+1 +427 Recommend this on Google

Labels: AWS, chaos monkey, fault-tolerance, reliability, resiliency, simian army

Newer Post Home Older Post

efficiency (2)
Elastic Load Balancer (1)
elasticsearch (4)
ELB (1)
EMR (2)
encoding (7)
encryption (1)
energy (1)
eucalyptus (1)
eureka (2)
evcache (2)
Experimentation (2)
failover (2)
falcor (2)
fault-tolerance (12)
fill (1)
flamegraphs (2)
Flow (1)
flux (1)
fmeasure (1)
footprint (1)
FreeBSD (1)
FRP (1)
functional reactive (1)
garbage (1)
garbage collection (1)
gc (1)
Genie (4)
git (1)
google spreadsheet (1)
Governator (1)
GPU (2)
gradle (1)
green (1)
Groovy (1)
Hack Day (3)
Hadoop (12)
HBase (1)
high volume (4)
high volume distributed systems (11)
Hive (2)
HTML5 (8)
https (2)
Hystrix (5)
IBM (1)
ice (1)
images (1)
IMF (4)
IMSC (3)

infrastructure (2)
initialization (1)
innovation (3)
insights (1)
Integration Testing (1)
inter process communication (1)
Interoperable Master Format (4)
iOS (1)
Ipv6 (2)
ISA-L (1)
isolation (1)
ISP (1)
java (5)
JavaScript (19)
jclouds (1)
jenkins (2)
kafka (4)
Karyon (2)
keystone (2)
lifecycle (1)
linux (2)
lipstick (2)
load balancing (3)
localization (1)
localization platform engineering (1)
locking (1)
locks (1)
log4j (1)
logging (2)
machine learning (9)
Mantis (1)
Map-Reduce (1)
media pipeline (2)
meetup (3)
memcache (2)
memcached (1)
Meson (2)
message security layer (1)
microservice (1)
migration (1)
Mobile (3)
modules (1)
monitoring (1)
msl (2)
nebula (1)
negative keywords (1)
Netflix (21)
Netflix API (8)
netflix graph (1)

- Netflix OSS (17)
- NetflixDevices (4)
- NetflixOSS (13)
- neural networks (1)
- NGINX (1)
- node.js (4)
- NoSQL (5)
- observability (1)
- Open Connect (2)
- Open source (12)
- operational excellence (2)
- operational insight (2)
- operational visibility (1)
- optimization (3)
- Originals (2)
- OSS (6)
- outage (1)
- page generation (1)
- payments (1)
- Paypal (1)
- performance (26)
- personalization (9)
- personalization infrastructure (1)
- phone (1)
- Photon (1)
- Pig (4)
- pipeline (1)
- pki (1)
- Playback (2)
- precision (1)
- prediction (2)
- predictive modeling (3)
- Presto (2)
- prize (1)
- prs (1)
- pubsub (1)
- pytheas (1)
- python (3)
- Quality (3)
- quality control (3)
- quality metric (2)
- queries (1)
- query (1)
- Queue (1)
- rca (2)
- React (3)
- Reactive Programming (2)
- real-time insights (2)
- real-time streaming (3)

- recall (1)
- Recipe (1)
- recommendations (11)
- Redis (5)
- reinvent (2)
- relevancy (1)
- reliability (7)
- remote procedure calls (1)
- renewable (1)
- research (3)
- resiliency (7)
- REST (3)
- Ribbon (2)
- Riot Games (1)
- root-cause analysis (2)
- Route53 (1)
- rule engine (1)
- Rx (2)
- Samza (1)
- scalability (12)
- scale (1)
- scripting library (1)
- search (4)
- security (10)
- sendfile (1)
- Servo (1)
- shared libraries (1)
- simian army (5)
- SimpleDB (3)
- site reliability (1)
- solr (1)
- spark (2)
- spinnaker (1)
- sqoop (1)
- ssd (2)
- ssl (4)
- STAASH (1)
- Stamos (1)
- stream processing (4)
- streaming (2)
- suro (1)
- SWF (1)
- synchronization (1)
- tablet (2)
- test (2)
- testability (1)
- testing (2)
- Timed Text (1)
- Titus (1)

tls (3)
traffic (1)
traffic optimization (1)
TTML (3)
TV (5)
UI (15)
UltraDNS (1)
unit test (3)
uptime (2)
user interface (5)
Velocity (1)
video quality (4)
visualization (1)
vizceral (1)
WebKit (3)
websockets (1)
Wii U (1)
windows (1)
winner (1)
winners (1)
Winston (1)
workflow (1)
workshop (1)
ZeroToDocker (1)
ZooKeeper (1)
zuul (1)