The Netflix Simian Army

Netflix Technology Blog Follow
Jul 19, 2011 · 4 min read

We've talked a bit in the past about our move to the cloud, and John shared some of our lessons learned in going through that transition in a previous post. Recently, we've been focusing on ways to improve availability and reliability and wanted to share some of our progress and thinking.

5 Lessons We've Learned Using AWS

Dorothy, you're not in Kansas anymore.

medium.com

The cloud is all about redundancy and fault-tolerance. Since no single component can guarantee 100% uptime (and even the most expensive hardware eventually fails), we have to design a cloud architecture where individual components can fail without affecting the availability of the entire system. In effect, we have to be stronger than our weakest link. We can use techniques like graceful degradation on dependency failures, as well as node-, rack-, datacenter-/availability-zone-, and even regionally-redundant deployments. But just designing a fault tolerant architecture is not enough. We have to constantly test our ability to actually survive these "once in a blue moon" failures.

Imagine getting a flat tire. Even if you have a spare tire in your trunk, do you know if it is inflated? Do you have the tools to change it? And, most importantly, do you remember how to do it right? One way to make sure you can deal with a flat tire on the freeway, in the rain, in the middle of the night is to poke a hole in your tire once a week in your driveway on a Sunday afternoon and go through the drill of replacing it. This is expensive and time-consuming in the real world, but can be (almost) free and automated in the cloud.

This was our philosophy when we built **Chaos Monkey**, a tool that randomly disables our production instances to make sure we can survive this common type of failure without any customer impact. The name comes from the idea of unleashing a wild monkey with a weapon in your data center (or cloud region) to randomly shoot down instances and chew through cables — all the while we continue serving our customers without interruption. By running Chaos Monkey in the middle of a business day, in a carefully monitored environment with engineers standing by to address any problems, we can still learn the lessons about the weaknesses of our system, and build automatic recovery mechanisms to deal with them. So next time an instance fails at 3 am on a Sunday, we won't even notice.

Inspired by the success of the Chaos Monkey, we've started creating new simians that induce various kinds of failures, or detect abnormal conditions, and test our ability to survive them; a virtual Simian Army to keep our cloud safe, secure, and highly available.

Latency Monkey induces artificial delays in our RESTful client-server communication layer to simulate service degradation and measures if upstream services respond appropriately. In addition, by making very large delays, we can simulate a node or even an entire service downtime (and test our ability to survive it) without physically bringing these instances down. This can be particularly useful when testing the fault-tolerance of a new service by simulating the failure of its dependencies, without making these dependencies unavailable to the rest of the system.

Conformity Monkey finds instances that don't adhere to best-practices and shuts them down. For example, we know that if we find instances that don't belong to an auto-scaling group, that's trouble waiting to happen. We shut them down to give the service owner the opportunity to re-launch them properly.

Doctor Monkey taps into health checks that run on each instance as well as monitors other external signs of health (e.g. CPU load) to detect unhealthy instances. Once unhealthy instances are detected, they are removed from service and after giving the service owners time to root-cause the problem, are eventually terminated.

Janitor Monkey ensures that our cloud environment is running free of clutter and waste. It searches for unused resources and disposes of them.

Security Monkey is an extension of Conformity Monkey. It finds security violations or vulnerabilities, such as improperly configured AWS security groups, and terminates the offending instances. It also ensures that all our SSL and DRM certificates are valid and are not coming up for renewal.

10–18 Monkey (short for Localization-Internationalization, or l10n-i18n) detects configuration and run time problems in instances serving customers in multiple geographic regions, using different languages and character sets.

Chaos Gorilla is similar to Chaos Monkey, but simulates an outage of an entire Amazon availability zone. We want to verify that our services automatically re-balance to the functional availability zones without user-visible impact or manual intervention.

With the ever-growing Netflix Simian Army by our side, constantly testing our resilience to all sorts of failures, we feel much more confident about our ability to deal with the inevitable failures that we'll encounter in production and to minimize or eliminate their impact to our subscribers. The cloud model is quite new for us (and the rest of the industry); fault-tolerance is a work in progress and we have ways to go to fully realize its benefits. Parts of the Simian Army have already been built, but much remains an aspiration — waiting for talented engineers to join the effort and make it a reality.

Ideas for new simians are coming in faster than we can keep up and if you have ideas, we'd love to hear them! The Simian Army is one of many initiatives we've launched to put the spotlight on increasing the reliability of our service and delivering to our customers an uninterrupted stream of entertainment. If you're interested in joining the fun, check out our jobs page.

- Yury Izrailevsky, Director of Cloud & Systems Infrastructure
- Ariel Tseitlin, Director of Cloud Solutions

. . .

Originally published at techblog.netflix.com on July 19, 2011.

Chaos Monkey Cloud Computing Netflix Simian Army Netflixsecurity