CHAOS MONKEY: A BEGINNER'S GUIDE



Chaos Monkey rides next in line to chaos engineering. Chaos engineering has risen as a best-practice form of engineering for any software development team using cloud computing. It is a kind of engineering designed to make computer systems more resilient, and a commonly used tool is Chaos Monkey.

What is Chaos Monkey?

The chaos engineering community operates under a consistent philosophy. It underlies the motivation for the creation of Chaos Monkey described in Netflix's blog post:

> *"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… 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."*

Chaos Monkey is the birth child of Netflix's engineering team. It was created at a time when Netflix shifted from providing its services via physical servers to cloud computing.

Cloud computing offers new challenges to software teams: computers are linked via network connections and there is less control over the cloud-based computers. Computers can also be of many different types, or old computers working in tandem with new computers. They can have

different operating systems, memory capacities, computer processing speeds, graphics processing speeds, and graphics memories. Because of their network-bound usability, choosing which task is delegated to each computer can be based on a network's latency and bandwidth.

When testing the system's resiliency, chaos engineering asks, If this computer goes down, can we move the task it was handling to another computer? And what other computer? Finally, how well does a system handle moving tasks from one computer to another? Chaos Monkey helps this with just this problem.

At its core, Chaos Monkey is designed to see how well a system shifts its resources when there is a computer outage. It also enforces better, proactive engineering practices by assuming that failure will and does occur, effectively creating the engineering community's popularly touted, non-intuitive statement, "building systems to fail".

Its name, Monkey, came about by envisioning a monkey running through a data center. It presses all the buttons, removes cables, and plugs in cables from one server rack to another. Given that a rampant monkey is now tampering with all parts of the IT infrastructure, how can an engineer design the system to work regardless of what the monkey does?

Getting started with Chaos Monkey

According to the Chaos Monkey GitHub, "Chaos Monkey randomly terminates virtual machine instances and containers that run inside of your production environment. Exposing engineers to failures more frequently incentivizes them to build resilient services."

To get started, Chaos Monkey is built upon another Netflix-made tool, Spinnaker, which is an open-source, multi-cloud continuous delivery platform. Spinnaker can be built on top of a cloud provider and works with all major providers. The providers it supports are:

- Google's App Engine
- Amazon Web Services
- Azure
- Cloud Factory
- DC/OS
- Google Compute Engine
- Kubernetes V2 (manifest based)
- Oracle

Once you've installed Spinnaker, you can install Chaos Monkey. For a closer look at how to use Chaos Monkey, see this page of the documentation.

A suite of Chaos tools

Chaos Monkey is not alone in the army. Since it was developed and its popularity rose, a whole suite of Chaos tools has been developed to simulate outages and test system response times. Chaos engineers now have the Simian Army to work with and Chaos Monkey falls in ranks with:

- **Chaos Kong** drops a whole AWS Region.
- **Chaos Gorilla** drops a whole AWS Availability Zone.
- **Latency Monkey** simulates network outages or delays.

- **Doctor Monkey** performs health checks.
- **Janitor Monkey** identifies unused resources.
- **Conformity Monkey** identifies non-conforming instances based on a set of rules.
- **Security Monkey** tests for known vulnerabilities. End of life in 2020, with alternatives available.
- **10-18 Monkey** detects configuration and run-time problems in instances serving customers in multiple geographic regions.

Additional resources

For more on this topic, check out these BMC Blogs and Guides:

- AWS Guide
- DevOps Guide
- Kubernetes Guide
- Advantages of Cloud Computing: 5 Benefits
- Using Spinnaker with Kubernetes for Continuous Delivery