Agile DevOps: Unleash the Chaos Monkey

How to deliberately use failure to ensure system resiliency

Paul Duvall

October 23, 2012

When would it ever be a good idea to randomly and intentionally try to terminate parts of your software system — including the hardware it runs on? How about early and often? In this Agile DevOps installment, DevOps expert Paul Duvall describes approaches to creating a Chaos Monkey (as it's been dubbed by Netflix) to ensure that your production infrastructure can recover from inevitable system failures.

View more content in this series

As I mentioned in the introductory article in this series, the CTO at Amazon.com, Werner Vogels, follows a simple principle: "Everything fails, all the time." I've often said that when it comes to software, you should never assume anything. If you continually prepare for failure in all the resources you manage — be it your hardware or software — you're more likely to succeed. This paradox is at the core of various tools that randomly disable or terminate production resources to test and ensure that automatic recovery mechanisms are a part of your infrastructure.

The most notable of these tools is the Chaos Monkey (see Related topics for this and other tools), which was developed by the Netflix technical team and open sourced earlier this year. In this article, I introduce the principles and steps for incorporating the Chaos Monkey into your infrastructure to ensure that it can handle the inevitable failures that will occur.

Tools like the Chaos Monkey are the result of an evolution toward ephemeral environments (discussed in the previous installment) brought on by infrastructure commoditization, virtualization, and cloud computing. It used to be that an infrastructure — with its physical machines, network switches, firewall, load balancers, software servers, and other resources — was something a team of engineers would set up manually one time. Then they'd monitor its usage, continually making manual tweaks to modify configuration, improve performance, and perform other activities. This is...
no longer considered a good practice and is quite simply impossible to do for any infrastructure of nontrivial scale. Tools like the Chaos Monkey apply monitoring, diagnostics, randomization, and disruption to the infrastructure to ensure that engineers apply automation to limit the impact users experience when big problems do occur.

Seek and destroy

A high-level list of the steps that make up the process for creating an environment for continuously testing an infrastructure includes:

- **Launching instances**: Start up some compute instances.
- **Creating an autonomic infrastructure**: Configure an infrastructure that launches new instances (based on the same template) when the infrastructure identifies unhealthy instances (see IBM’s portal on autonomic computing in Related topics).
- **Applying automatic testing to ensure automatic recovery**: Run the tests during hours when engineers are ready to react and fix.
- **Learning and preventing**: When failures do happen, react and prevent the failure from happening again.

<table>
<thead>
<tr>
<th>Why test the infrastructure in production?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some argue that intentionally disrupting an infrastructure should only be done in nonproduction environments — never in production. The Chaos Monkey deliberately destroys resources in production environments at a time of day when most engineers can fix any errors that occur. Moreover, nothing compares to finding and fixing problems in real environments.</td>
</tr>
</tbody>
</table>

The basis for a continuously tested and self-healing infrastructure is the conviction that:

- An infrastructure will fail.
- You need to test for these failures in production when engineers are available.
- When the failure occurs again, your infrastructure must automatically recover from it without users ever noticing.

In many ways, it’s the perfect technical implementation of an organization truly committed to continuous improvement or *kaizen* (see Related topics).

In summary, this new breed of *resiliency* tool and accompanying infrastructure has the following features:

- **Monitoring**: Daemon processes are continuously run to *diagnose* errors.
- **Diagnostics**: Diagnostic tools are run as part of system monitoring.
- **Disruption**: The infrastructure is intentionally disrupted by shutting down instances and other disruptive activities.
- **Randomization**: To prevent expected outcomes and behavior, the *disruption* is randomly applied to the infrastructure.
- **Self-healing infrastructure**: Although not a part of the resiliency tool, the expected resultant behavior is that teams continue to apply and improve on an autonomic infrastructure capable of recovering from service disruptions without users noticing.
The Chaos Monkey

Netflix heavily utilizes a cloud infrastructure for streaming movies to users, along with other functionality. In July 2012, it was reported that Netflix users streamed more than 1 billion hours in June 2012. In other words, Netflix isn't a trivial user of the cloud; it uses it on a massive scale.

A Quick Start Guide on GitHub authored by the Netflix tech team (see "Quick Start Guide for Chaos Monkey" in Related topics) describes the steps to go through to get the Chaos Monkey up and running. The following list gives you some more information on the tools that the Chaos Monkey uses. Be sure to run the commands described in the guide to remove any unused resources, or else you will be continually charged for usage.

- **Auto Scaling**: Auto Scaling is a specific feature of Amazon Web Services that enables you to scale compute capacity up and down based on demand — through rules that you define. Although it's an AWS-specific feature, you can create this type of scalable environment with your — private or public — cloud infrastructure. Auto Scaling has two key components: a launch configuration and an Auto Scaling Group. A launch configuration defines how an instance within an Auto Scaling Group is launched. An Auto Scaling Group is a collection of instances for which to apply a particular launch configuration.
- **SimpleDB**: SimpleDB is a NoSQL database you can use to persist data. You need to define a SimpleDB domain. It's used by the Chaos Monkey to store state.
- **Gradle**: Gradle is a build tool. It's used to build the Chaos Monkey and to start the Jetty application container.
- **Properties file**: You need to modify a simianarmy.properties file with credentials and other configurable information.
- **Jetty**: The in-memory Jetty server runs the Chaos Monkey to disrupt your infrastructure randomly.

A Simian Army

The Chaos Monkey is the first entry in the Netflix technical team's *Simian Army*. In Table 1, I list other tools that Netflix has proposed that will constitute the Simian Army (see Related topics):

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chaos Gorilla</td>
<td>Simulates outage of an entire availability zone</td>
</tr>
<tr>
<td>Conformity Monkey</td>
<td>Shuts down instances that don't adhere to best practices</td>
</tr>
<tr>
<td>Doctor Monkey</td>
<td>Performs health checks (such as CPU)</td>
</tr>
<tr>
<td>Janitor Monkey</td>
<td>Searches for unused resources and disposes of them</td>
</tr>
<tr>
<td>Latency Monkey</td>
<td>Creates artificial delays in client-server communication</td>
</tr>
<tr>
<td>Security Monkey</td>
<td>Finds security vulnerabilities such as improperly configured security groups</td>
</tr>
</tbody>
</table>

These are just a few ideas. The possibilities for other ways to apply a combination of monitoring, diagnostics, testing, and intentional destruction in cloud-based production environments are endless.
Unleash the fury

Get involved

developerWorks Agile transformation provides news, discussions, and training to help you and your organization build a foundation on agile development principles.

In this article, you learned that you can truly begin creating autonomic infrastructure capable of healing itself with the help of tools such as the Chaos Monkey and a cloud environment.

In the next article, you'll learn about test-driven infrastructure. In it, you'll learn how to apply test-driven development techniques — commonly used by developers for application code — for your infrastructure, using tools such as Cucumber.
Related topics

- "Failure as a Service" (Haryadi S. Gunawi et al., University of California at Berkeley Technical Report, July 2011): This paper discuss routinely performing large-scale failure drills in real deployments.
- "Netlix streaming tops 1 billion hours in month for first time" (Rachel King, CNET, July 2012): Netflix's digital streaming service hit a major milestone recently: more than 1 billion hours viewed in a month.
- Kaizen: Wikipedia describes this approach, which originated in Japan, to continuous improvement of processes.
- "Chaos Monkey Released Into The Wild" (Cory Bennett and Ariel Tseitlin, Netflix, July 2012): Announcement of the official release of the open source Chaos Monkey on GitHub.
- Simian Army: Netflix's open source Simian Army. The Chaos Monkey is one entry in what will be a suite of open source tools.
- Bees with machine guns: A utility for arming (creating) many bees (micro EC2 instances) to attack (load test) targets (web applications).
- IBM Tivoli Provisioning Manager: Tivoli Provisioning Manager enables a dynamic infrastructure by automating the management of physical servers, virtual servers, software, storage, and networks.
- IBM Tivoli System Automation for Multiplatforms: Tivoli System Automation for Multiplatforms provides high availability and automation for enterprise-wide applications and IT services.
- Evaluate IBM products in the way that suits you best.

© Copyright IBM Corporation 2012
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)