What is chaos engineering?

Published November 14, 2023 | Updated April 29, 2024 | 8 min read

Saif Gunja

DevOps

Testing for mishaps you can predict is essential. But with the complexity that comes with digital transformation and cloud-native architecture, teams need a way to make sure applications can withstand the “chaos” of production. Chaos engineering answers this need so organizations can deliver robust, resilient cloud-native applications that can stand up under any conditions.

What is chaos engineering?

Chaos engineering is a method of testing distributed software that deliberately introduces failure and faulty scenarios to verify its resilience in the face of random disruptions. These disruptions can cause applications to respond unpredictably and break under pressure. Chaos engineers ask why.

Practitioners subject software to a controlled, simulated crisis to test for unstable behavior. The crisis could be technical, natural, or malicious events, for example, an earthquake affecting data center availability, or a cyberattack infecting applications and websites. As software performance degrades or fails, the chaos engineers' findings enable developers to add resiliency to the code, so the application remains intact in an emergency.

As chaos engineers grow confident in their testing, they change more variables and broaden the scope of the disaster. The many disaster scenarios and outcomes allow chaos engineers to better model what happens to applications and microservices, which gives them increasing intelligence to share with developers to perfect software and cloud-native infrastructure.

The history of chaos engineering

Netflix pioneered chaos engineering out of necessity. In 2009, the purveyor of online videos migrated to AWS cloud infrastructure to deliver its entertainment to a growing

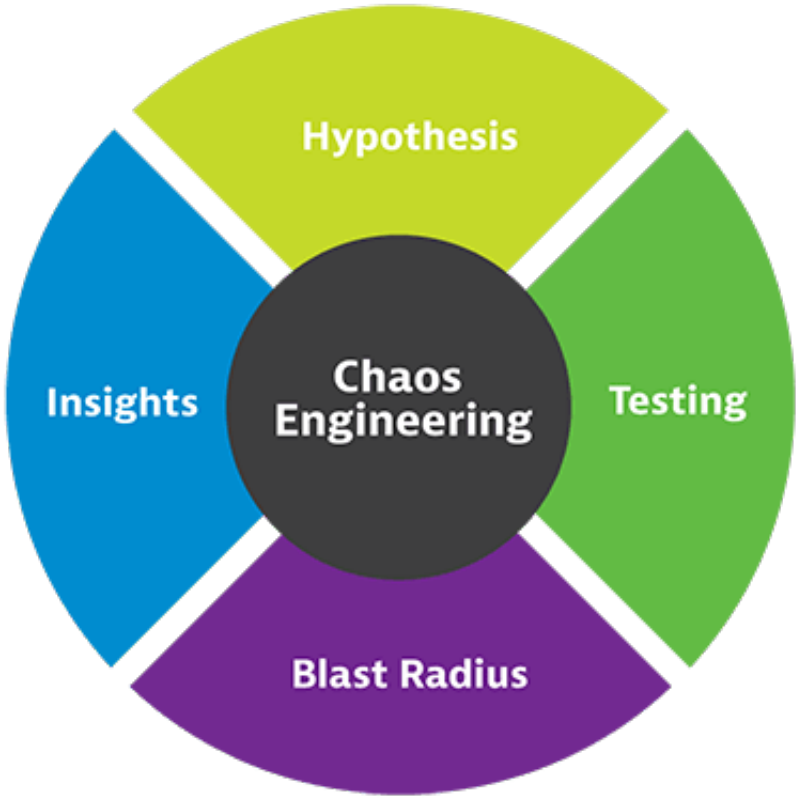
audience. But the cloud brought new complexities, such as increasing connections and dependencies. It created more uncertainty than the load-balancing issues the entertainment firm saw in its data centers. If any touchpoint in the cloud failed, the quality of the viewers' experience could degrade. So, the organization sought to reduce complexity and raise production quality.

In 2010, Netflix introduced a technology to switch production software instances off randomly—like setting a monkey loose in a server room—to test how the cloud handled its services. Thus, the tool Chaos Monkey was born.

Chaos engineering matured at organizations such as Netflix, and gave rise to technologies such as Gremlin (2016), becoming more targeted and knowledge-based. The science has spawned specialized chaos engineers who dedicate themselves to disrupting cloud software and the on-prem systems they interact with to make them resilient. Now, chaos engineering is an established profession, stirring up managed trouble to stabilize cloud software.

How does chaos engineering work?

Chaos engineering starts with understanding the software's expected behavior.



1. **Hypothesis**. Engineers ask themselves what should happen if they change a variable. If they terminate services randomly, they assume the service will continue uninterrupted. The question and the assumption form a hypothesis.
2. **Testing**. To test the hypothesis, chaos engineers orchestrate simulated uncertainty combined with load testing and watch for signs of upheaval in the services, infrastructure, networks, and devices that deliver the application. Any failures in the stack break the hypothesis.
3. **Blast radius**. By isolating and studying failures, engineers can understand what happens under unstable cloud conditions. Any damage or influence caused by

the test is known as the ‘blast radius.’ Chaos engineers can manage the blast radius by controlling the tests.

4. **Insights**. The discoveries form inputs into the software development and delivery process, so new software and microservices will better stand up to unforeseeable events.

To mitigate damage to production environments, chaos engineers start in a non-production environment, then slowly extend to production in a controlled way. Once established, chaos engineering becomes an effective way to fine-tune service-level indicators and objectives, improve alerting, and build more efficient dashboards, so you know you are collecting all the data you need to accurately observe and analyze your environment.

Who uses chaos engineering?

Chaos engineering generally originates from small teams within DevOps, often involving applications running in both pre-production and production environments. Because it can touch many systems, chaos engineering can have broad implications, affecting groups and stakeholders across the organization.

A disruption spanning hardware, networks, and cloud infrastructure can require input and participation from network and infrastructure architects, risk experts, security teams, and even procurement officers. That's a good thing. The greater the scope of the test, the more useful chaos engineering becomes.

Although a small team generally owns and manages the chaos engineering effort, it's a practice that often requires input from—and provides benefits to—the village.

The benefits of chaos testing

The insights you can gain by testing the limits of your applications deliver a lot of benefits for your development teams and your overall business. Here are just a few benefits of a healthy, well-managed chaos engineering practice.

- **Increases resiliency and reliability**. Chaos testing enriches the organization's intelligence about how software performs under stress and how to make it more resilient.

- **Accelerates innovation**. Intelligence from chaos testing funnels back to developers who can implement design changes that make software more durable and improve production quality.
- **Advances collaboration**. Developers aren't the only group to see advantages. The insights chaos engineers glean from their experiments elevate the expertise of the technical group, leading to response times and better collaboration.
- **Speeds incident response**. By learning what failure scenarios are possible, these teams can speed up troubleshooting, repairs, and incident response.
- **Improves customer satisfaction**. Increased resilience and faster response times mean less downtime. Greater innovation and collaboration from development and SRE teams means better software that meets new customer demands quickly with efficiency and high performance.
- **Boosts business outcomes**. Chaos testing can extend an organization's competitive advantage through faster time-to-value, saving time, money, and resources, and producing a better bottom line.

The more resilient an organization's software is, the more consumers and business customers can enjoy its services without distraction or disappointment.

The challenges and pitfalls of chaos engineering

Although the benefits of chaos testing are clear, it is a practice that should be

undertaken with deliberation. Here are the top concerns and challenges.

- **Unnecessary damage**. The major concern with chaos testing is the potential for unnecessary damage. Chaos engineering can lead to a real-world loss that exceeds the allowances of justifiable testing. To limit the cost of uncovering application vulnerabilities, organizations should avoid tests that overrun the designated blast radius. The goal is to control the blast radius so you can pinpoint the cause of failure without unnecessarily introducing new points of failure.
- **Lack of observability**. Establishing that control can be easier said than done. Lack of end-to-end observability and monitoring of all systems a blast radius might affect is a common problem. Without comprehensive observability, it can be difficult to understand critical dependencies vs non-critical dependencies, or to have adequate context to understand the true business impact of a failure or degradation in order to prioritize fixes. Lack of visibility can also make it difficult for teams to determine the exact root cause of an issue, which can complicate remediation plans.
- **Unclear starting system state**. Another issue is having a clear picture of the starting state of the system before the test is run. Without this clarity, teams can have difficulty understanding the true effects of the test. This can diminish the effectiveness of chaos testing and can put downstream systems at greater risk, which makes it harder to control the blast radius.

How to get started with chaos engineering

As with any scientific experiment, getting started with chaos engineering requires a little preparation, organization, and the ability to monitor and measure results.

1. **Know the starting state of your environment**. To plan a well-controlled chaos test, you should understand the applications, microservices, and architectural design of your environment so you can recognize the effects of the test. Having a baseline you can compare to the end state creates a blueprint for monitoring during the testing and analyzing results after.
2. **Ask what could go wrong and establish a hypothesis**. With a clear idea of the system's starting state, ask what could go wrong. Understand service-level indicators and service-level objectives and use them as a basis for establishing an assumption for how the system should work under pressure.
3. **Introduce chaos one variable at a time**. To keep control of the blast radius, introduce only a little chaos at a time, so you can appreciate the results. Be prepared to abort the experiment under certain conditions, so no harm comes to production software, and also have a roll-back plan if something goes wrong. During the test, try to disprove the hypothesis to discover areas to focus on to improve system resilience.
4. **Monitor and record the results**. Monitor the experiment to record any nuances in application behavior. Analyze the results to see how the application responds and whether testing achieved the teams' expectations. Use investigation tools to understand the precise root causes of slow-downs and failures.

Controlling the chaos

Solutions like Gremlin provide crucial management tools to plan and execute chaos engineering experiments. It makes experiments repeatable and scalable so teams can apply them to future experiments of the same or larger stacks.

Automatic and intelligent observability from Dynatrace delivers insights into the effects of chaos testing, so engineers can steer chaos experiments with care. To monitor the blast radius, Dynatrace observes the systems undergoing chaos experiments. With visibility across the full software stack, Dynatrace provides crucial contextual analysis to isolate the root cause of failures exposed by chaos testing.

Effective monitoring from Dynatrace offers an essential panoramic lens for the engineers driving chaos testing, helping them understand dependencies and predict how outages will affect the system at large. Should the chaos reach further than intended, insights from Dynatrace help teams quickly remediate any actual harm to the application's functionality.

Organizations can achieve application resiliency in any stage of digital transformation, and chaos engineering is a great tool. However, before playing with fire, it's critical to have the right measures in place to predict and cope with the multitude of failure scenarios this approach can bring.

To hear more about chaos engineering in action, listen to the PurePerformance podcast, Chaos Engineering Stories that could have prevented a global pandemic with Ana Medina, Sr. Chaos Engineer at

Gremlin.

Listen to podcast!

Blog posts

Product news

Subscribe now

Tags: app performance testing, chaos engineering, devops cloud

Saif Gunja

The Author

Saif led product marketing for the Dynatrace Automation solution, focusing on DevOps, platform engineering, and SRE teams. Saif brings 10+ years of IT and marketing experience from his previous roles at VMware, Apple and Deloitte.

Disclaimer: The views expressed on this blog are my own and do not reflect the views of Dynatrace LLC or its affiliates.

You may also like

Unlock enhanced CI/CD insights in GitLab for pipeline visibility

By Maher Husein - July 9, 2025

Stay focused, code faster: How Google Gemini Code Assist keeps developers in

Insights into your Azure DevOps pipelines

By Rohan Shah - March 11, 2025

the zone

By Michael Villiger - April 9, 2025

Read now →

Read now →

Read now →

Looking for answers?

Start a new discussion or ask for help in our Q&A forum.

Go to forum