Gremlin

2026 EDITION

Buyer's Guide for Chaos Engineering Tools

EVALUATION CRITERIA AND TOOL COMPARISON

Table of contents

Introduction

As modern systems become more complex and the speed of software delivery increases, Chaos Engineering and resilience testing have become a critical part of software testing and operations. With Chaos Engineering and resilience testing, you can uncover and remediate hidden reliability risks before they cause an outage that impacts your customers.

But with a variety of Chaos Engineering tools on the market, how do you know which one is the right one? Choose wisely, and you'll enable your entire organization at a whole new level of reliability, one with a culture built around proactively finding and fixing issues instead of firefighting. Choose poorly, and you could end up with underutilized shelfware that's more trouble than it's worth and leaves you finding risks the old-fashioned way—when they take down your systems.

Ultimately, choosing the right Chaos Engineering tool comes down to a combination of technical and organizational factors. The best Chaos Engineering tool for you should be able to safely run the experiments you need, when and where you need them, while also enabling teams across your organization with capabilities that help them meet their reliability goals.

Gremlin has helped hundreds of the world's largest enterprise companies navigate the selection process and successfully launch effective, thriving reliability practices. We've also seen what happens when there's a mismatch between the tool and the organization: stalled initiatives, engineer burnout, and no escape from the break/fix cycle.

Based on that experience, we've compiled a list of key questions and capabilities that every enterprise should ask to narrow down the right tool for both getting their Chaos Engineering initiative off the ground and systematically improving their reliability posture for years to come.

Platform and Pipeline Support

> "We wanted a practice that would allow us to experiment and uncover problems we hadn't even thought of yet. With Chaos Engineering, we can learn a lot about how our systems work and how we can make them better."
>
> —Senior Site Reliability Engineer, Consumer Tech Industry

Look at the technical capabilities to see if the Chaos Engineering tool can run the experiments you need, where and when you need them to run.

1 Does it support all of your infrastructure?

Experiments and tests need to be able to run on any infrastructure your system uses—and on all application layers.

If you can only run tests on your cloud-based services, but have key applications and dependencies running on-prem, then you have a blind spot. The right Chaos Engineering tool for your organization will be able to perform experiments everywhere your applications run. It doesn't matter if your applications run on **bare metal, on-prem, or cloud**—and if you run a multi-cloud architecture, then that includes **multiple cloud providers.**

Ideally, you should be able to use a single Chaos Engineering tool across all your systems. This helps prevent tool sprawl and unnecessary costs, but, more importantly, **using a single tool prevents blind spots.** When you use separate tools for different parts of your infrastructure, this creates silos with the potential to cause gaps in testing coverage—especially when it comes to cross-platform dependencies.

But coverage goes beyond infrastructure to making sure every layer of your application is covered. This is especially important if your architecture uses microservices or fully managed services, such as **container orchestration** (e.g. Kubernetes) or **serverless** (e.g. AWS Lambda) environments. If you want to test the reliability of these applications, you need a tool **capable of testing on the application layer.**

Look for a tool that can **run across the full spectrum of infrastructure** and **application layers.**

2 Can it cover all of your failure conditions?

Chaos Engineering experiments should cover the state, resource, and network capabilities of any service you run.

The whole point of Chaos Engineering is to make the unknowns known, and you can't do that if your Chaos Engineering tool has a blind spot in its capabilities. Experiments should be able to cover the **state of the target itself** (e.g. shutdown/restart it, change the time, or kill specific processes), **resources used by the target** (including memory, CPU, I/O, and storage), and **potential network issues** (such as latency, packet loss, dropped connections, or DNS issues).

Look for a **broad range of coverage** that can test all key aspects of your services—this includes the ability to **test catastrophic failures** across your company to verify **disaster recovery plans.**

3 Does it support all phases of your SDLC?

Experiments should be able to run at the point where they give your organization the most value.

Chaos Engineering experiments and reliability tests can be used in a variety of stages along the software development lifecycle. Testing has been successfully used in staging environments, within a CI/CD pipeline, immediately post-deploy, on Beta products before they go GA, and as regular testing in production environments.

Each of these environments has different technological requirements, so you want to **make sure your tool can run tests and experiments where you need them to happen.**

At the same time, you should **consider where you're going to test in the future.** Companies will often test in multiple stages as their reliability practice matures and evolves, but transitioning away from one tool and onboarding a new one can be a rough process with hidden costs and an adoption ramp-up. You'll want to make sure your tool is flexible enough to cover your needs now and in the future.

Look for a tool that can safely run tests in **staging and performance environments**, as part of a **CI/CD pipeline**, and in **production environments.**

Safety and Security

> "We're building a digital platform, and that means a lot of our tools need to have enterprise support maturity. It needed to be secure because we're using the tools to do specific infrastructure certifications. So the tool needed to be enterprise-scale and comply with our security."
>
> —Manager of DevOps Engineering, IT Industry

Make sure you can get insights without damaging your systems or making them vulnerable.

4 Is it safe?

Tests or experiments shouldn't damage your systems or cause outages.

The whole point of Chaos Engineering is to artificially create environments where incidents and outages can occur. But wanting to understand what happens when your data center burns down doesn't mean you actually want to light servers on fire.

Chaos Engineering tools that run safely will be able to **limit the scope and scale of experiments** so teams can gain confidence as they scale up, and **respond quickly to a negative result**, stop the experiment, and restore the previous state to minimize any impact. They'll also come with **an obvious way to stop all experiments** (like a big red button) if you see responses you don't like.

Chaos Engineering should be approached with a deliberate strategy and minimal impact—and your tool should enable that. A tool that makes it easy to break things indiscriminately without specificity can make it hard to learn from your experiments, while tools that don't easily halt or roll back experiments if things escalate can cause more damage than they'd prevent.

Look for a tool that **automatically halts in response to service or infrastructure monitors**, has a **halt all experiments** button, and **automatic rollback** capability.

5 Is it secure?

The right tools should comply with your security policies and best practices—including meeting common standards.

Chaos Engineering tools will need some core permissions to create specific conditions during a test. It's essential that they can **run tests and handle your security credentials securely**. At the same time, Chaos Engineering experiments will often run on systems that store or use **sensitive information**, especially in industries like finance or healthcare.

Additionally, many companies (especially enterprise organizations) have strict standards for their technology, such as **SOC 2 compliance**. The right Chaos Engineering tool will need to have the documentation and compliance standards that your security team needs to sign off on. Otherwise, it's never going to be able to run any meaningful experiments in production or production-like environments.

Some security standards also require everything to **run on a company's private network**. This means your tool needs to be able to deploy both the **agent and control layer within your firewall.**

Look for a tool that **meets your security requirements, complies with the standards** you need, and has a **proven track record of securely handling data and credentials.**

Predictable Cost

> **"Our value proposition was: This is how you help close the gaps from what you don't know. And if we fix two or three P1 issues, then we have more than paid for this tool."**
>
> —Director of Quality Assurance, Travel Industry

Analyze the true cost of running the Chaos Engineering tool against the value it creates—which includes the cost of engineering resources.

6 What are the ongoing engineering costs?

Chaos Engineering and resilience testing should save your engineering time by spending less of it fixing outages, not taking up more in maintenance.

With any software tool, you have to weigh all resources when looking at its cost, and Chaos Engineering tools are no different. It might seem free to create your own tool or use an open source tool, but you need to **account for the engineering time spent building and maintaining** that tool.

You also want to **look at the long-term costs as you scale**. The ability to scale needs to come either from the platform itself (in the form of standardization and automation) or your internal product team needs to scale the tool. For example, open source costs might start small when it's only a handful of engineers using the tool, but as you scale, so will the maintenance, configuration, training, and integration requirements, which, in turn, will drive up the cost.

That engineering cost is also more than just the cost of salaries. When you're spending engineering resources building and maintaining a tool, that **time is taken away from innovating your core product.**

Look for a Chaos Engineering tool that **fits the level of engineering resources** you're willing to commit to setting it up and maintaining it. And if that amount is little to none, then you may want to look at a SaaS option.

7 Are the software fees clear and predictable?

Chaos Engineering and resilience testing should create more value for the business as you scale up—without billing surprises.

Chaos Engineering and resilience testing should create more value for the business as you scale up—without billing surprises.

When you use a SaaS tool, your costs are potentially more straightforward, since the SaaS vendor bears development and maintenance expenses. But at the same time, you could end up spending more than you want if there are hidden costs or unclear pricing structures. **Pricing transparency is essential** to make sure that you **only pay for the tools your company needs.**

You should also consider future costs as your practice grows. A SaaS tool might have a pricing model that costs very little for a small team, such as a per-seat model, but can get very expensive when scaling up to an enterprise-level organization with thousands of seats.

On the other hand, while **consumption-based pricing** can directly tie cost to use, it can also be a barrier to adoption by actively discouraging tests and experiments to keep costs low. Not only can this prevent scaling your testing efforts, but it can also leave you with blind spots where crucial tests are skipped to save money.

Finally, you also want to **look at the costs of any tool dependencies**. For example, Chaos Engineering tools from cloud providers often require integration with their monitoring products. If you're already using those products, then this won't have a huge impact, but if you aren't, then it creates additional costs to account for.

Look for a Chaos Engineering tool that has a **transparent pricing model**, has **costs that won't get in the way** as you scale, and **doesn't have hidden dependency costs.**

Enterprise Capabilities

> "We need tools that make it so all of our engineers can easily do these things. We'd like to get it to the point where tests are automatic. And you don't have to just turn over all of that responsibility to individual developers—you can do it in a managed way."
>
> —Managing Director and Head of Core Engineering, Banking and Finance Industry

Make sure the tool has the additional capabilities for wide adoption across an enterprise-level organization.

8 Is it proven with your use case?

The right Chaos Engineering tool will have proven that it can handle the unique pitfalls and use case requirements of your industry and company size.

Every industry has its own unique use case requirements. Some, such as financial services or transportation, have to **meet strict regulatory standards for security and reliability.** Others, like retail or events, need to be able to **test massive spikes in traffic from events like Black Friday or ticket launches**, where even a short outage can cost millions.

Look at other customers using the Chaos Engineering tool. If the tool has successfully been used in your industry (or a similar industry), then you and your leadership team can have more confidence in its ability to meet your needs.

You should also look at the scope of the current customers to see whether it can **meet the demands of an enterprise-scale organization.** This will help you verify that it has the technological capacity to safely and securely run experiments on sprawling, enterprise-scale architectures.

Look for a tool that has a **history of successful use in your industry** (or similar industries) and is **proven to run at enterprise scale.**

9 Does it have the organizational capabilities you need?

Testing will need to be done by teams across your organization, and they'll need metrics and collaborative capabilities to align their efforts.

Chaos Engineering and reliability tests are at their most effective when they cover a wide breadth of your services to give you as complete a picture as possible. While there should be **advanced experiments** available to experts, there should also be **standardized test suites** that can be run by service owners, performance engineers, QA engineers, and others. Technology leaders, cloud architects, and centralized site reliability teams should also be able to **customize reliability test suites** to fit specific business goals.

Just as importantly, there needs to be capabilities that will align the teams. You'll need **forward-looking reliability scores or metrics** that can be used by any service or team, as well as **team and organization-level dashboards.**

10 Does it have automation for testing and deployment?

Automated tests and deployment capabilities make adoption easier and faster, which makes reliability efforts more effective.

Speedy and easy adoption is essential if you're going to reap the benefits of a Chaos Engineering tool. Manual deployment and testing processes are both big hurdles to adoption. Engineering teams have enough to do already, and adding a lot of manual work to their plate often leads to them skipping testing altogether.

Adoption can be sped up with a combination of **centralized administrative controls** and **automatic service catalog creation** that can easily set up services and roll out testing suites to teams. Once rolled out, there should be **automated testing** that can be set up to run on a schedule or with each deployment. At the same time, there should be **management capabilities and policies** that can interrupt or control testing schedules for easy organization-wide management.

11 Does it integrate with your existing tools?

The tool should integrate smoothly with your existing tools and be able to function independently without having to buy into a whole new ecosystem.

Tool and vendor sprawl is all too common in modern organizations. Different teams will use different tools, platforms, and infrastructures, covering everything from pipeline tools to observability platforms to cloud providers. It's important to consider how any prospective Chaos Engineering tool will fit into your existing process and technology stack.

Some Chaos Engineering tools are **a part of a larger ecosystem**, which can be a benefit if your entire organization is already using that system. But if you're working with a variety of technologies, or want to have the flexibility to change things in the future, then you should look at **purpose-built Chaos Engineering tools** that are designed with **native integrations for the most common related tools** (e.g. observability or cloud providers) and **APIs for custom integrations.**

The right Chaos Engineering tool for your organization will be able to integrate with the tools used by all of your teams, allowing you to get a complete picture of your reliability posture.

AI Capabilities

> **"In high-velocity environments, reliability can't be an afterthought. Reliability Intelligence equips SRE and performance teams with deep, real-time insights from telemetry and trace data—enabling early detection of reliability regressions, faster root cause isolation, and proactive remediation without disrupting release velocity."**
>
> —Director of Performance Engineering, Global Retailer

Ensure your tool is using AI responsibly to effectively increase your velocity without increasing risk—and that it allows you to improve your AI product.

12 Does it provide expert insight?

The right Chaos Engineering tool will use AI to help you move faster and more strategically based on expertise instead of guesses.

When integrated into Chaos Engineering tools, Large Language Models (LLMs) can be valuable tools for **speeding up analysis** and helping you **take effective action faster.** But the **quality of their output is dependent on the quality of their data**, and any analysis or recommendation has to be backed by expert, reliable data.

Your tool evaluation should include an examination of both the **capabilities** and the **data source**. For capabilities, focus on features that will amplify your efforts. This includes remediation **recommendations based on AI analysis** of your test results or **suggestions for focusing your efforts**, like where to test next.

More importantly, the AI capability should be built on **reliability expertise** gained by **successfully improving reliability at scale.** Be sure to examine the sources behind the contextual data. What are the data sources? Does it **incorporate real-world use cases**? Were the techniques in the data effective?

Look for a tool that **incorporates AI capabilities** built on **reliability expertise** that will enable your team to **move faster and more effectively** improve reliability.

13 Can it improve the reliability of your AI applications?

The right Chaos Engineering tool will include specific capabilities that will help you improve the reliability of your AI applications.

AI applications are built on similar infrastructure to standard systems, but there are usually a few key differences. So make sure that the Chaos Engineering tool can **test the infrastructure underpinning your AI applications**, including the capability to **test GPU exhaustion** for LLM training.

Third-party integrations, such as calls to LLMs, are also a crucial part of AI applications. Your Chaos Engineering tool needs to be able to **fully map these dependencies** and **run tests on them** to verify how your application responds to failures.

The nature of LLMs means these tests need to go beyond simple failure tests. For example, latency can impact the type or quality of response returned by LLMs, so your Chaos Engineering tool needs to be able to **run a wide range of tests on LLM dependencies**.

Look for a tool that will help you **make all of your applications more reliable**, which includes your **AI applications and systems.**

14 Does it integrate with your AI tooling?

The right Chaos Engineering tool will integrate with the AI tooling already part of your workflows and processes.

AI has been rapidly integrated into workflows across industries. Developers are using agents to code, test, and deploy their applications, and your Chaos Engineering tool should be able to **work with these new AI processes**.

There are two key markers to look for during your evaluation: **MCP server integrations** and **API capabilities**.

MCP, or Model Context Protocol, servers allow applications to **integrate directly with LLMs**, allowing you to use your model of choice to interact with your Chaos Engineering tool.

But if you want your LLM to take actions, then your Chaos Engineering tool needs **robust API capabilities**. The MCP server will be able to use the API calls to pull data, analyze results, and more. This does give your LLM a lot of access to your systems, so make sure the tool has an **API-first approach** coupled with **comprehensive security and safety measures**.

15 Can it keep up with AI-driven development?

The right Chaos Engineering tool needs to be able to work smoothly with the increased release frequency from AI-driven development.

Agentic AI has drastically increased deployment rates. This increase in release frequency has been accompanied by **more bugs, more dependencies, and more potential issues**.

Chaos Engineering and resilience testing are more important than ever, but **gating your deployments risks nullifying the velocity gains**. Additionally, rapid deployment means **your production environment is constantly changing**, introducing new risks that **can't be found by testing in staging** alone.

In order to keep up with AI-driven development, your tool needs to be able to **test safely and automatically in production** environments. This is the only way to **get accurate visibility into your reliability** so you can uncover and address issues introduced by rapid AI deployment. The best way to **catch these risks** and **get ahead of reliability fluctuations** is by running a set group of tests on a regular schedule so you can **track your reliability over time**.

Look for a tool that can run **automated, standardized groups of tests** with a **proven track record** of running tests **safely and securely in production environments**.

Comparison Chart

To help you evaluate your choices, we've taken the key traits from the questions above and compared them to the most common Chaos Engineering tools.

	Gremlin	AWS FIS & Resilience Hub	Azure Chaos Studio	Harness Resilience Testing	LitmusChaos	Steadybit	Chaos Mesh	Chaos Monkey
DESCRIPTION	Enterprise platform combining Chaos Engineering tools and reliability tests	Policy-based resilience testing and governance hub for AWS applications	Managed service for testing applications and workloads on Azure	Chaos Engineering module for Harness CI/CD platform built on LitmusChaos	Kubernetes-native Chaos Engineering platform and CNCF project	Chaos Engineering platform with open source extensions and heavy UI focus	A CNCF project that brings various types of fault simulation to Kubernetes	Single test tool that randomly terminates instances in production
LICENSE	Commercial	Commercial	Commercial	Commercial	Open source	Commercial	Open source	Open source
PLATFORM & PIPELINE SUPPORT	 *Full coverage of environments, test types, and SDLC stages*	 *AWS only, limited services and tests*	 *Azure only, limited services and tests*	 *Limited platform and test support. Focused on CI/CD testing*	 *Kubernetes only, library of built-in and crowdsourced experiments*	 *Full coverage of environments, test types, and SDLC stages*	 *Kubernetes only, wide test coverage across SDLC stages*	 *One attack type on limited environments*
SAFETY & SECURITY	 *Auto rollback, auto and manual halt, meets highest security standards (inc. SOC 2)*	 *Auto-stop conditions, "Shared responsibility model" for security*	 *Policy limited test scope, manual halting, manual, role-based security*	 *Limited rollback, potential vulnerabilities from crowdsourced tests, SOC2-compliant*	 *Safety and security burdens are on the user*	 *Limited rollback/halt with manual safety efforts. Reliance on third-party extensions*	 *Limited auto-stop status checks. Role-based permission by service account*	 *No rollback or blast radius limits, no longer developed or maintained*
PREDICTABLE COST	 *All-inclusive price per agent with unlimited experiments/tests*	 *Consumption-based pricing, requires using additional services (e.g. CloudWatch)*	 *Consumption-pricing per action-minute, which can be difficult to predict at scale*	 *Seat-based pricing, requires using additional Harness services and platform*	 *Unpredictable engineering costs for build, maintenance, and operations*	 *Team-based pricing is predictable, but price jumps could limit scaling or adoption*	 *Unpredictable engineering costs for build, maintenance, and operations*	 *Unpredictable engineering costs for build, maintenance, and operations*
ENTERPRISE CAPABILITIES	 *Reliability scoring, management, and standardization, and enterprise-proven in multiple industries*	 *Built on AWS for service owners and developers, with Resilience Hub for architect-level management*	 *Built on Azure for service-owner use with team-based role and permission controls*	 *Organizational features built around CI/CD platform, limited CE-specific controls and limited reliability scoring*	 *Basic functionality is available, but more advanced features and integrations must be built by your engineers*	 *Designed for software engineers with minimal management capabilities and no standardized control*	 *Basic functionality is available, but more advanced features and integrations must be built by your engineers*	 *Single-function tool that relies on Spinnaker for organizational or dashboard features*
AI CAPABILITIES	 *Reliability Intelligence recommendations built on decades of expertise. MCP Server with API-first approach and built-in safety controls*	 *Recommendations for Cloudwatch and Systems Manager. MCP access via AWS MCP server projects*	 *No current AI analysis or recommendations. No explicit support or integration with MCP servers*	 *Recommendations for tests, but no result analysis or remediation. Limited MCP server integration with Cursor IDE*	 *MCP server integration, but no recommendations for test or remediation*	 *MCP server support, but no recommendation capabilities for either tests or remediations*	 *Limited open source MCP servers. No recommendation capabilities for either tests or remediations*	 *Support ended before LLMs, so no AI capabilities exist*

Conclusion:

When done right, Chaos Engineering can transform your organization

Chaos Engineering is about more than a few experiments. When used correctly, Chaos Engineering can increase the reliability and resiliency of systems across your organization to create a better experience for your customers.

But it's only the beginning. Once you've gotten good results with Chaos Engineering, you need to scale it across your organization with systematic reliability management. That's the only way to measure, manage, and improve the reliability of all your systems.

Make sure the partner you choose is the right one to help your entire organization achieve its reliability goals.

Gremlin

Gremlin is the reliability management platform trusted by the world's largest enterprises across financial services, SaaS, retail, and media to help meet uptime and performance goals.

Instead of lagging indicators that show past reliability, Gremlin gives engineering teams predictive reliability data so they can systematically measure, manage, and improve reliability.

By combining failure testing, risk detection, and dependency mapping with automation, standardization, and in-depth reporting, Gremlin makes it easy to see where systems are at risk, prioritize what to fix, and track progress across teams over time to prove the results.

Learn how to manage your reliability at gremlin.com