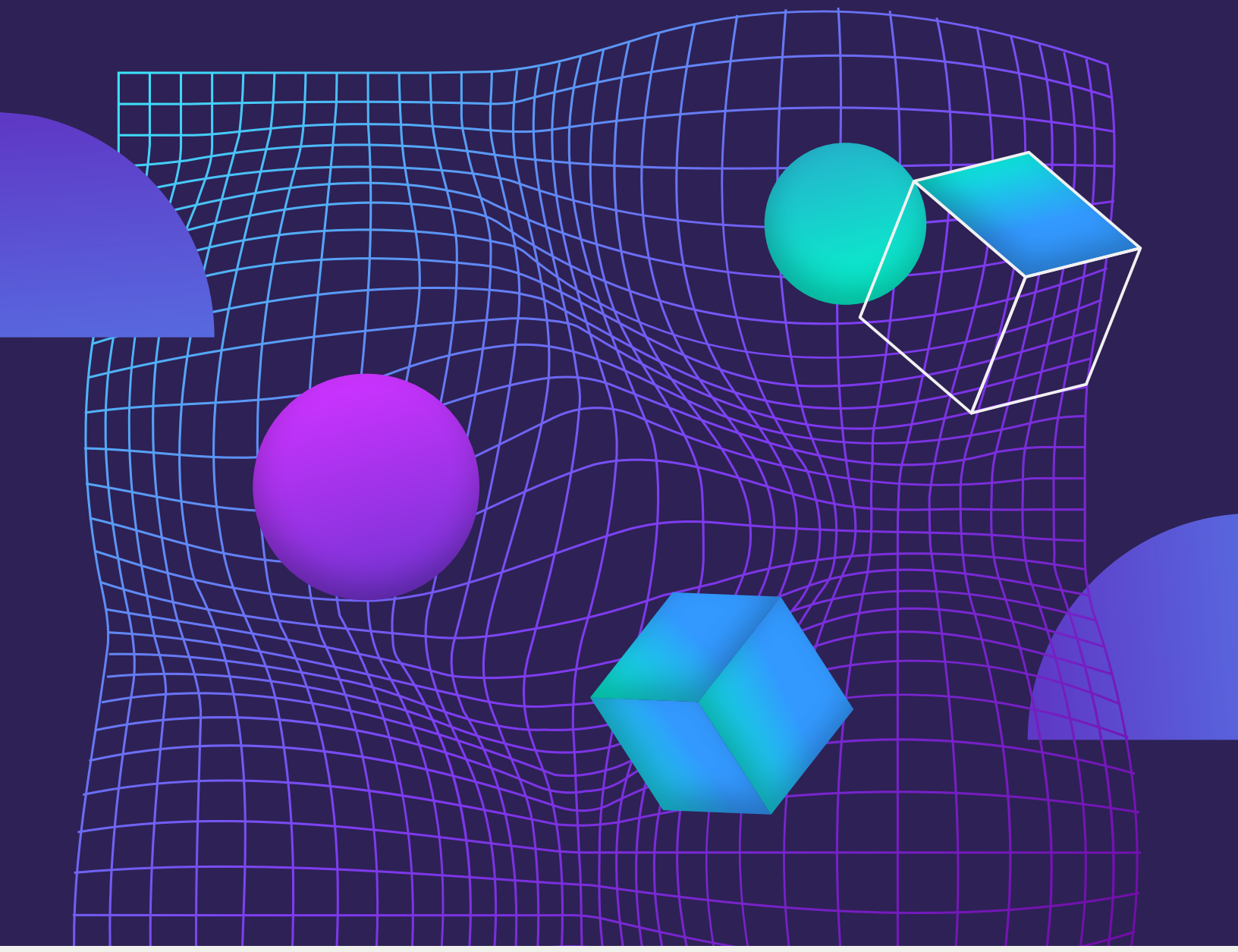
Gremlin

Modernizing Disaster Recovery Testing

HOW TO VALIDATE OUTAGE RESILIENCE AT ENTERPRISE SCALE



DISASTER RECOVERY TESTING

EBOOK

Outages happen.

In fact, there were over 15,000 outages in 2025, with a global price tag of over $37 billion. And many of them were caused by third-party failures.

The outage could be from a major cloud provider, like when AWS us-east-1 went down, affecting 70,000 companies and costing up to $581 million. Or it could be a major dependency incident, like CrowdStrike's bad config push that hit 8.5 million Windows systems and cost Fortune 500 companies $4.5 billion.

Every organization needs to be ready for when things go catastrophically wrong. Unfortunately, many companies are woefully unprepared.

According to Cockroach Labs' State of Resilience report, only 20% of organizations consider themselves prepared for outages, while only 33% regularly perform disaster recovery simulations. And 71% of organizations don't do any failover testing to make sure outage prevention protocols work.

71% **of organizations don't do any failover testing**

Engineering leaders know it's important to build and test disaster recovery plans. But when there's this level of disconnect between importance and adoption, that means there's a problem deeper than simple prioritization by leadership.

There is something fundamentally broken about the disaster recovery testing process.

It needs to be modernized.

Current approaches to disaster recovery testing

Disaster recovery testing is happening at companies, but it's a painful process that doesn't produce the results teams and companies need. Let's look at where the most common methods are effective and when they fall short.

Individual reporting

This is the most basic method of disaster recovery verification. Individual engineering teams are given a checklist they need to go through in order to comply with regulations or standards. They check the box that a specific action or mechanism exists, but there's no further verification or validation.

Pros	Cons
• Minimal disruption that takes up engineering time but doesn't impact systems. • Taps into the detailed knowledge of teams who know their systems best. • Can be done asynchronously.	• Relies on assumptions and best intentions. • No verification. • Responses and standards can vary between teams. • Doesn't account for the entire system.

Result: *Individual teams set up resilience and recovery mechanisms, but there's no proof they work.*

Tabletop exercises

People from various teams get together in a conference room or on a large call to walk through a simulated disaster scenario. For many teams, this is where their disaster recovery testing stops. They'll run through the simulation, go through all the boxes on compliance checklists, and assume everything will go smoothly during an actual disaster.

Pros	Cons
• Catches big holes and makes sure everyone knows their roles. • No system disruption since no failures are actually tested. • Builds cross-functional awareness.	• Validates human knowledge, not system behavior. • Doesn't test recovery mechanisms under real-world conditions. • Compliance can't be backed up with auditable test data.

Result: *Teams know the theory and what should happen during a disaster scenario, but there's no confirmation that the plans will be effective.*

Backup restoration tests

A backup isn't a true backup until you test it. In this method, engineering teams test the backups themselves, verifying that the backup systems are set up correctly to spin up within the expected recovery time.

Pros	Cons
• Verifies backups are actual backups. • Safely tests restoration without disrupting core systems. • Uncovers config and provisioning issues to make sure backups can handle full system loads.	• Only tests restoration, not whether the failover will be engaged during a disaster scenario. • Tests in a controlled environment, not real-world conditions. • Usually done on a team basis rather than the entire system.

Result: *Teams have confidence that their backups will spin up, but it assumes that failover mechanisms work correctly under failure conditions.*

Failover simulations

Very few companies perform these tests, and with good reason. These tests validate failover by shutting down and/or rerouting traffic from your primary systems to your backup systems. And since they have a high likelihood of disruption, everyone involved is either in the room or on the call. They're messy, often painful exercises that risk your system and often involve coordination with external vendors.

Pros	Cons
• Verifies failover mechanisms work. • Validates redundancy provisioning to make sure recovery and performance standards are met. • Comprehensive results for compliance and reporting.	• Extremely disruptive to systems and engineering teams. • Usually involves controlled shutdown or rerouting, which doesn't accurately simulate a disaster. • Doesn't compensate for system shifts that happen between annual tests. • Relies on custom scripts or manual efforts that don't scale and increase risk due to manual rollbacks.

Result: *Teams know their failover plans work on a broad scale, but don't have the time to dig deeper due to the immense disruption and time commitment.*

Engineering organizations need a new approach

What would a different approach to disaster recovery testing look like?
Right now, most organizations with formal disaster recovery plans stop at tabletop exercises or get bogged down in expensive and manual testing. In both cases, the new approach would have to enable them to easily perform actual testing and verification. For organizations already performing failover testing, it would need to remove the disruption and toil while building on the engineering foundation of backup restoration testing.

All told, it would need to be:

- **Systematic:** The method has to be easily applied to teams across the organization to verify standards. It should also enable engineering teams to repair any issues that come up and validate their fixes on their own between major testing actions.

- **Non-disruptive:** The testing method should minimize disruption to systems and minimize disruption to engineering roadmaps and deployment schedules.

- **Accurate:** Instead of relying on assumptions or best intentions, the method needs to be able to simulate realistic failure conditions and prove that recovery mechanisms operate correctly.

- **Auditable:** The testing results need to be gathered and presented in reports that can be used for compliance verification with enough granularity to tie data back to specific tests.

- **Ongoing:** Full disaster recovery testing usually happens annually, but systems are constantly changing. Teams need to be able to verify resilience between annual tests, providing better test coverage and making annual testing exercises easier.

The path to modern disaster recovery testing

There are two practices at the core of any modern disaster recovery testing approach. The first is fault injection testing. By safely simulating a failure, such as losing all network access to a specific region, engineering teams can see exactly how their systems respond. These tests can be used to accurately simulate specific disaster scenarios.

As part of the process, each test is paired with health checks that set the exact parameters for when a test passes or fails. During a test, if a service's performance metrics violate that health check, then the test is automatically stopped and the previous state is restored. These health checks allow teams to safely simulate failures without causing disruptions.

Combining these allows you to perform tests at scale across your company and create a systematic approach to disaster recovery testing:

1. Start with service-level testing
Run tests on individual services against the most common failure modes, such as single-zone failure, dependency loss, resource exhaustion, or network latency. Each of these tests is lightweight and easily repeatable by teams. If a test fails, they can address the issue and retest to validate the fixes.

And since the tests can be performed safely, they can be run during business hours as part of an engineering team's normal sprint, minimizing disruption. Once teams have run the initial batch of tests, they should test regularly to stay on top of any shifts in systems caused by deployments, configuration drift, topology changes, and more.

2. Test a system-wide disaster scenario
The only difference between a system-wide scenario and service-level testing is the number of services being tested at once. Unfortunately, catastrophic failures don't happen to individual services. If a cloud region goes down, it goes down for everyone connected to it, and you need to make sure your system is prepared. You'll need to be able to answer questions like: Can your redundant systems handle the sudden surge as all of your traffic is rerouted?

These scenarios should be done from a central place and use the tests you ran on individual services. That way, you don't need to get everyone in the room during the test. Each team will know how their service reacts, and your resilience test will roll back if anything was missed.

3. Continuous minor testing

Between company-wide disaster scenarios, teams should continue testing their services. Rapid deployment means systems are constantly changing, and that pace has only increased with agentic AI. Just because a service passed a test two weeks ago doesn't mean it will pass this week. By regularly testing, teams can address these shifts and changes as they come up to maintain resilience.

And then instead of dreading the next large disaster recovery test, teams can confidently go into it knowing that their systems are reliable. Which means less disruption, easier, faster tests, and better resilience for your company.

How Gremlin makes disaster recovery testing easy

Gremlin has been helping organizations understand how their systems react to failure for over 10 years and has worked with companies across the Fortune 500 to improve their availability and reliability.

The Gremlin reliability management platform makes it easy for teams to measure, manage, and improve their reliability. And that includes verifying your disaster recovery plans. This approach takes advantage of specific Gremlin features:

- **Reliability test suites:** Pre-built groups of tests that contain the most common outage causes, including zone/region outages, resource exhaustion, and more. These are centrally managed and can be rolled out across your entire company for consistent, standardized testing. Any service set up with test suites has its dependencies identified for testing, can be set up for passive risk detection, and can use automation to regularly run tests.

How to use for disaster recovery testing: *Run a standardized test suite against each service to verify resilience to failures from disaster scenarios on a service level.*

- **Reliability reporting:** Reliability test results and detected risks are reported on the individual service, team, and company-wide level. Leaders and execs can see their systems' reliability posture at a glance, while individual teams can quickly identify reliability risks and track their reliability over time.

How to use for disaster recovery testing: *Track service-level results between large tests to verify reliability over time. Teams can use reports to quickly uncover and address risks, while leaders can use the data to guide investments and prioritization decisions.*

- **Disaster Recovery Testing:** Run the same test simultaneously across your entire team, organization, or company to accurately simulate disaster scenarios. Any tests that fail are stopped and the services are restored to minimize disruption while the rest of the tests continue.

How to use for disaster recovery testing: *Replace massive failover simulations that take thousands of engineering hours with centralized tests done by a single person in a small fraction of the time. Since it uses the same tests as service-level testing, there are no sudden surprises, which means you don't need a massive call or meeting while you do it.*

- **Disaster Recovery Reporting:** Bring all of the Disaster Recovery Testing results together into a single auditable report that provides both a high-level view and the ability to dive into each individual service's results. Reports are all stored for each disaster scenario you run, creating an easy track record for compliance and reporting.

How to use for disaster recovery testing: *Leadership and teams can use the results to align on any work that will be done to address failed tests. Results can be exported for regulatory compliance or record-keeping.*

By combining the regular, standardized testing of reliability management with company-wide Disaster Recovery Testing, companies are using Gremlin to accurately and systematically test their disaster recovery plans 90% faster than other methods.

> "Disaster Recovery Testing gives us a fast, centralized way to continuously validate and demonstrate our resilience to catastrophic events so we can stay prepared and keep services online."
>
> —Sreekanth Rajagopal, Head of Non-Functional Testing, Visa Cross-Border Solutions

Make sure your recovery plan works before the disaster

No matter how carefully you design your disaster recovery plan or how skillfully your engineering teams build your recovery mechanisms, you won't know if they actually work until they face real failure conditions.

Which means you have two options: you can find out when there's a disaster or by running tests.

Up until now, testing has been arduous, painful, and disruptive enough that waiting for a disaster almost seems like the better option.

The systematic approach changes that.

This approach turns disaster recovery testing from a compliance cost center into a competitive advantage, one where reliability issues are found and addressed before they impact customers, where resilience is proven with data to leaders and regulators, and where you have confidence that the next time there's a major outage, your systems will recover smoothly.

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