Gremlin

Break things on purpose

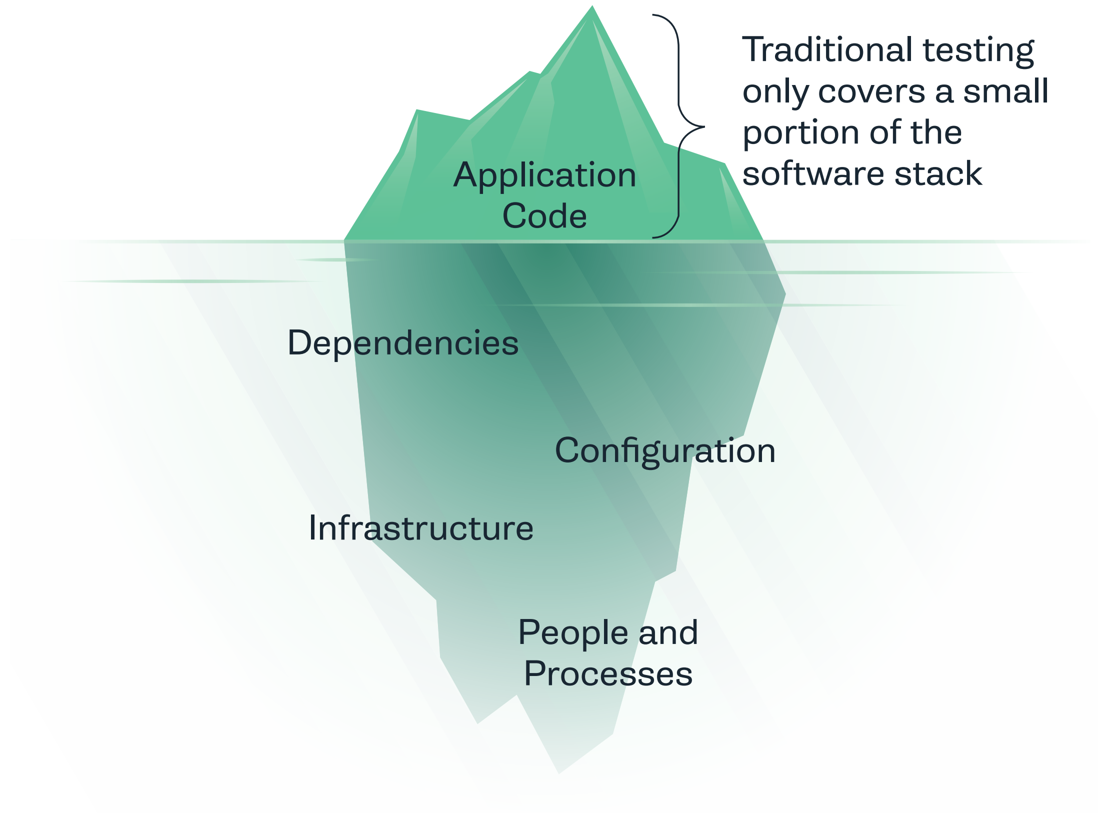
Build reliable systems

In our evolving digital economy, speed of innovation is a competitive advantage. However, the evolution in our software architectural designs has led to increased complexity. Traditional testing techniques are insufficient to test these complicated new environments for reliability.

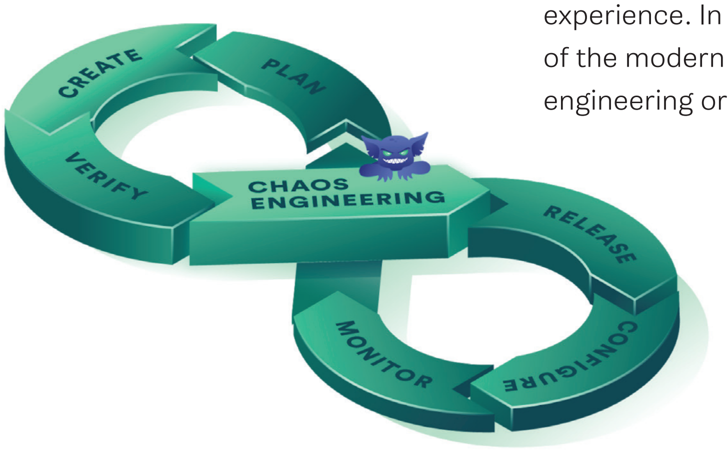
If we want continuous delivery with less downtime, we need a way to test against common failure scenarios in distributed systems at the speed that code is pushed into production.

Applications are the driver behind business success, and uptime is critical for flawless customer experiences and brand reputation. Architecture and cultural shifts such as cloud, microservices, and DevOps were developed to rapidly add features and continually improve the customer experience. With that said, these new approaches exponentially increase the number of components to manage, as well as the rate of change. This means critical systems are failing today, causing financial loss, customer dissatisfaction, and employee burnout.

Traditional QA testing is not enough to keep up with these new paradigms and the complexity of these new environments. Much of modern testing has moved to be automated in CI/CD pipelines to check code quality quickly, but they still miss testing the application end-to-end. The complexity inherent in modern systems means small changes in systems, from configuration changes to autoscaling behaviors, have drastic downstream impacts. Traditional testing, such as unit testing and integration testing, can never catch all of the edge cases. If we want continuous delivery with less downtime, we need a way to test against common failure scenarios in distributed systems at the speed that code is pushed into production.



Chaos Engineering is the best way to proactively test our applications for reliability. It is the practice of performing thoughtful, controlled tests to reveal weaknesses in a system before they can impact our customers. By running real-world failure scenarios, we can build confidence that our complex distributed systems will deliver an uninterrupted customer experience. In this way, Chaos Engineering is the capstone of the modern development toolchain and forward looking engineering organizations are rolling out the discipline.



> *“By 2023, 40% of organizations will implement chaos engineering practices as part of DevOps initiatives, reducing unplanned downtime by 20%.*
>
> **Gartner, Predicts 2020, Agile and DevOps Are Key to Digital Transformation**

In order to begin adopting Chaos Engineering as a part of our development lifecycle, we need a tool that can easily integrate into existing environments. Gremlin was purpose-built by Chaos Engineering experts to be easy to install anywhere so we can test our entire application stack for common failure scenarios. Enterprises across retail, financial services, high tech, hospitality, travel & logistics, healthcare, and media & entertainment leverage Gremlin to build more reliable systems.

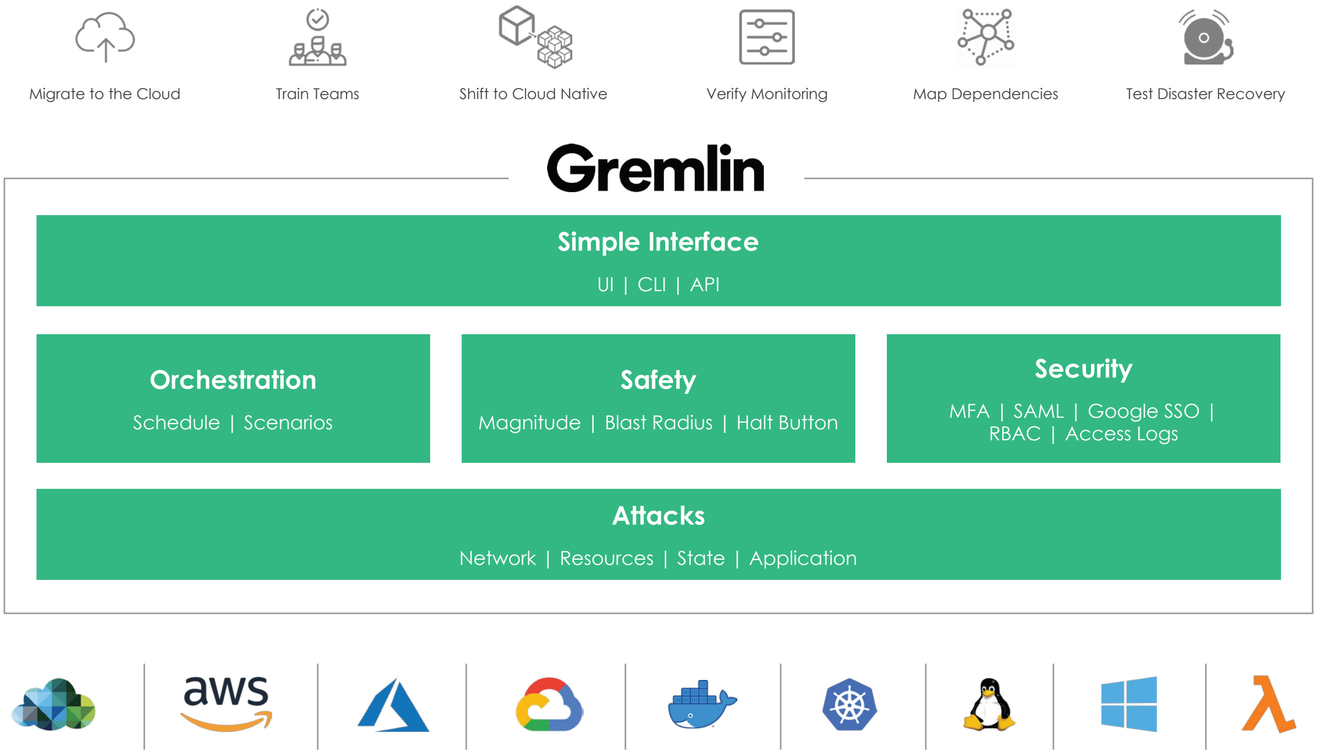
Gremlin is designed to close the reliability gap

Gremlin is the enterprise Chaos Engineering platform, built by the experts who pioneered Chaos Engineering at Amazon and Netflix. The platform offers the most comprehensive set of experiments that replicate real-world failures across an entire application. Leveraging Gremlin’s Chaos Engineering software as a service enables us to build confidence that our applications, infrastructure, and remediation plans were built to withstand real-world high traffic and failure events that can impact performance and uptime. With Gremlin, we can reduce downtime and time to resolution, resulting in increased employee retention, customer satisfaction, and focus on innovation. Backed by years of specialized experience in Chaos Engineering, the platform was designed to be simple, safe, secure, and comprehensive for engineering teams to test their entire stack.

Simple

Gremlin is designed to make it easy to add Chaos Engineering to existing processes without additional complexity.

- Gremlin's **agents are cloud and on-prem agnostic and take minutes to install** in most Linux, Windows, container, Kubernetes, and application environments.
- Control attacks from **an intuitive interface and a well-documented API** to manually or automatically run tests.
- Gremlin includes **Scenarios, which are based on years of on-call experience** to run a set of attacks that replicate common real-world incidents, or chain together unique attack sequences to reproduce failures experienced in the past.
- **Schedule attacks** on specific days and times or schedule attacks to be random to validate changes in production.



Simple Interface

UI | CLI | API

Orchestration

Schedule | Scenarios

Safety

Magnitude | Blast Radius | Halt Button

Security

MFA | SAML | Google SSO | RBAC | Access Logs

Attacks

Network | Resources | State | Application

Safe

Perform experiments knowing there are controls in place to ensure testing is done safely.

- Focus the **magnitude and blast radius of attacks to be precisely targeted**, allowing isolation of dependencies to pinpoint the cause of bugs.
- Gremlin enables us to **manually halt attacks in progress and roll back** the impact if we observe an adverse impact.
- Gremlin includes failsafe measures that ensure **attacks automatically halt immediately if scenarios exceed predefined thresholds**.

Secure

Gremlin has focused on designing the application with enterprise-level security in mind.

- The agent leverages the least permissions necessary and **never has root access**.
- The application offers **MFA, SAML, SSO, and RBAC** to control who can perform experiments.
- **All attacks are tracked with an audit trail** to avoid abuse.
- For validation of our security controls, Gremlin submits to regular third party testing and is **SOC 2 Type 2 certified**.

Comprehensive

Our platform has the unique capabilities to allow teams to test every layer of the application stack, including infrastructure, operating system, application, and end-user experience, without ever switching context.

- To **mimic notoriously unstable networks**, Gremlin can introduce latency, blackhole traffic, drop packets, and fail DNS.
- To **test against state failures**, Gremlin can reboot hosts, kill processes and change the OS clock time.
- Gremlin can **test high load on infrastructure resources** by overloading the CPU, consuming memory, inducing I/O and filling disk.
- **Target containers and Kubernetes** environments leveraging these platforms' primitives.
- **Narrow the impact to a single user, device, or percentage of traffic.** Apply network tests against serverless functions.

	CHAOS MONKEY	Gremlin
Number of attack types	1	12
Intuitive SaaS control plane	✗	✓
Open source	✓	✗
Halt attacks	✗	✓
Blast radius control	✗	✓
Attack containers and use k8s primitives to target	✗	✓
Enterprise support, SLAs, indemnification	✗	✓
Time to set up and run the first attack	Days	Minutes

> *"The team had been struggling with an issue that had caused incidents in production a number of times over the last few months. They had had trouble reproducing it. With Gremlin, we were able to both reproduce it, get the fix and then verify that the fix worked.*

Matthew Simons, Senior Product Development Manager at Workiva

Benefits of Chaos Engineering with Gremlin

Reduce incidents before they impact customers

Outages cost businesses over $700 billion per year[1]. Chaos Engineering with Gremlin helps proactively detect and fix issues in our applications and infrastructures to avoid these costly outages. By performing Chaos Engineering, we can uncover autoscaling configuration issues, unnecessary dependencies, and improperly configured region failover before they become a problem for customers.

Lower detection and resolution time

When our applications are facing an issue, every minute counts. The average company takes 24 hours to resolve an application failure[2]. We can beat that average by running GameDays to prepare teams for failure scenarios and accurately tune our monitoring and alerting when an outage occurs.

> *Diagnosing the SLO issues in 2017 took hours. We used Chaos Engineering to improve Time to Diagnose of the system to less than 5 minutes by testing and tuning our logging, monitoring, and traceability.*
>
> - Gustavo Leiva, Director of Engineering at Backcountry

1. IHS Markit: The Cost of Server, Application, and Network Downtime: North American Enterprise Survey and Calculator 2. IDC: DevOps and the Cost of Downtime: Fortune 1000 Best Practice Metrics Quantified

> “ *Gremlin helps bring value to our customers because it allows us to innovate quickly and safely, so we can provide robust solutions to the needs of our customers and to the delight of our customers.*
>
> **Nate Vogel, Sr. Director of Data Platforms at Charter Communications**

Ship code faster with higher quality

Adding Chaos Engineering into our CI/CD pipeline allows us to test every deploy for reliability. Finding bugs early helps us push higher quality code and rerunning experiments prevents configurations drifting into failure. Fixing these issues early shortens the repair time and leaves more time for developers to focus on innovation.

Launch new products with confidence

Migrations and new product launches can have a lot of unknowns prior to launch. Without recreating real-world failure scenarios, it is impossible to know how new or rearchitected application will handle production traffic. Applying Chaos Engineering with Gremlin prior to launch can prepare applications for common failure scenarios and test for reliability.

> *There are always going to be issues that are rarely seen when developing software, and these issues can have a big impact. We avoided any cracks in our software when VTM GO went live by working with Gremlin.*
>
> - Glenn Heylen, IT Architect at DPG Media

Use Cases

Cloud migration

Moving away from established systems and controls can be scary. In fact, 99% of cloud misconfigurations go unnoticed[3]. Move to the cloud with confidence by testing new environments with common failure scenarios such as DNS failures and network issues. We can make sure we have properly set up our cloud environment to take advantage of autoscaling and replication with tests like region failure and resource usage spikes.

Incident response

Getting teams prepared for routine outages and big events becomes more difficult as the velocity of change increases, and as our teams grow. Playbooks need to be routinely tested to ensure they are up to date and teams need to drill them to ensure a rapid response to incidents that arise. Gremlin will help employees prepare for remediations with real-world failures like database node failures or lost connectivity to test their ability to resolve the issues that arise. The best way to get started is by using our predefined Scenarios or recreate an event faced before and see how teams have improved their time to detection and time to resolution.

Microservices and Kubernetes

Adopting containers, Kubernetes, and serverless adds the benefit of automated deployments, autoscaling, and more efficient use of resources. However, these new infrastructure offerings add complexity and often feel like black boxes. The shortage of Kubernetes expertise compounds the issue. We can apply Chaos Engineering to test policies and train teams to understand how their new systems behave in various failure modes to gain confidence in each pull request. Running resource and state attacks can ensure that autoscaling and pod/host replacement configurations are tuned for the particulars of different applications.

[3] McAfee: Cloud Native: The IaaS Adoption Risk Report

Monitoring and alerting

Monitoring and alerting systems are only as good as their settings. The best way to ensure our tools' capabilities are properly tuned to give actionable information is by using accurately reproduced situations that have happened in the past, rather than using guesses or estimates. Replicate common issues such as runaway processes, unreachable services, database node failure, and monitoring agent failure with the safety of controlled experiments. Confirm that precise alerts go out to the correct team members at the right time.

Dependency mapping and troubleshooting

Minimizing critical dependencies and preparing for failures is important in our applications. A loss in connection or delay in a non-critical service like advertisements should not take down the entire application. If an application loses a connection to the database, it should not lock other services in retry loops. Gremlin makes it easy to ensure the customer experience does not degrade exponentially when network connections are poor. Running network and state attacks against dependencies can help determine if systems can fail gracefully and independently.

Disaster recovery

Region redundancy, storage backups, and archives help us gain confidence that in the event of a total data loss or account lockout, we will be able to recover. Double-check that the restore process will run smoothly by running a region or database loss exercise. Begin with small experiments of lost connections to a single node, and expand to entire region loss. We can refine the process and speed to recovery by making iterative changes and repeating the experiments to test for improvement. We can confirm that during small events and large scale disasters, we can recover quickly and effectively.

[3] McAfee: Cloud Native: The IaaS Adoption Risk Report

About Gremlin

Gremlin builds reliable systems through chaos engineering - the engineering philosophy that safely stress tests systems to proactively identify and fix unknown faults, and prepare teams for failures in the future. Gremlin aims to make the internet more reliable and prevent costly and reputation damaging outages by empowering engineers to safely experiment on complex systems to build knowledge and more resilient software. Customers such as Target, Walmart, Twilio, Siemens, and Under Armour trust Gremlin to improve the reliability of their systems.