Gremlin

How to Convince Your Organization to Adopt Chaos Engineering

How to make a name for yourself championing reliability and SRE in your organization

This playbook is designed to guide an engineer or technical stakeholder within an organization in winning over coworkers and management to the understanding of and ultimately widespread adoption of Chaos Engineering along with some good Site Reliability Engineering (SRE) practices applicable to any organization that cares about reliability. The playbook provides ideas and techniques that can be used to sell the need and benefits internally in your organization as well as guide the initial implementation in a way that will lead to success and growth across the organization.

To do this, you'll need to convince your organization to spend human and financial resources over the course of a year. This is your chance to build the case for why this is both necessary and important. Implementing something new like Chaos Engineering successfully is a good way to get promoted and help the organization succeed, and this guide is here to help you.

In this guide we tell you:

- Why Chaos Engineering matters
- How to identify a use case within your organization
- How to sell the discipline of Chaos Engineering
- How to build the business case
- What the procurement process looks like
- How to build for the long term

It’s worth noting that while we’re focused on Chaos Engineering, many of the principles and concepts presented here will be applicable for anyone looking to make organizational or cultural change, or to bring on a new software tool to your organization.

The first goal is to demonstrate that Chaos Engineering is a critical tool to prove and improve reliability. As we move to microservices, Kubernetes, and other distributed systems, traditional testing is inadequate to ensure reliability. We need ways to do systemic testing, even in large, distributed environments.

The second goal is to get buy-in and cooperation in the procurement process. This includes picking that first project for Chaos Engineering and building it out with a plan for long-term success and eventual expansion within the wider organization. We want to integrate Chaos Engineering into the organization as an ongoing practice that drives continuous value, not just as a tool or a one-off process.

As you move ahead, think about which team in the organization should own Chaos Engineering? Site reliability engineering (SRE)? DevOps? All teams? The ultimate goal is to create and cultivate a culture in the organization where reliability is feature zero. Even though it is a non-functional requirement, it is vital to organization and team success and needs to be seen as such. Consider which team will benefit the most from Chaos Engineering, as this will help provide a starting point.

Why Chaos Engineering Matters

Creating reliable software is a fundamental necessity for modern cloud applications and architectures. As we move to the cloud or rearchitect our systems to be cloud native, our systems are becoming distributed by design and the potential for unplanned failure and unexpected outages increases significantly. Additionally, moving to DevOps further complicated reliability testing.

Traditional quality assurance only covers the application layer of our software stack. And no amount of traditional QA testing or other traditional testing is going to verify whether our application, its various services, or the entire system will respond reliably under any condition, whether "working as designed" or under extreme loads and unusual circumstances.

A failure at any software stack or application layer can disrupt the customer experience. Traditional QA testing methods will not catch any of these potential problem conditions before they actually happen.

Chaos Engineering lets you compare what you think will happen to what actually happens in your systems. You literally "break things on purpose" to learn how to build more resilient systems.

By proactively testing how a system responds under stress, we can identify and fix failures before they end up in the news. Ultimately, the goal of Chaos Engineering is to enhance the stability and resiliency of our systems.

Chaos and Reliability Engineering techniques are quickly gaining traction as essential disciplines to building reliable applications. Many organizations – both big and small – have embraced Chaos Engineering over the last few years.

Chaos Engineering can:

- Test and verify a much wider range of scenarios of both software and infrastructure than any form of traditional testing has been able to accomplish
- Find problems traditional testing can't expose
- Be performed safely and effectively throughout the development lifecycle, from staging to production
- Help teams understand how systems behave in the real world, not just how they break or what bugs they have

Chaos Engineering matters to your organization because fewer incidents means more time to innovate. Reducing the risk of downtime means happier customers and more revenue.

Engineers benefit from Chaos Engineering because it provides more holistic testing of complex applications, which means more reliable systems. More reliable systems means less time fighting fires and getting paged in the middle of the night. Engineers also benefit from an increased understanding of the system and how it works, so when an incident does happen, they are better prepared to get things fixed in a much shorter amount of time.

Gremlin fits into the benefit equation because it is the only SaaS solution for Chaos Engineering today. Gremlin lets you run experiments safely and securely, giving you the benefits without the hassle of having to figure out how to design and implement more time-consuming options like adapting an open source product or writing your own solution. Either option will require dedicated engineer time to get things set up and usable. Or, in just a few minutes you can deploy the Gremlin daemon into your system and be ready to run your first attack.

How to Identify a Use Case Within Your Organization

Chaos Engineering is scientific. Start small. Be methodical. Have a plan and an expected outcome. Compare and use baseline data with data collected during the experiment to either confirm or disprove expectations. Experiments should be designed with precision, limiting the blast radius and being intentional about the magnitude of each attack.

With that foundation, determine the systems and services in your critical path. Make sure monitoring is set up to measure metrics from services that will be impacted directly, as well as services that might be impacted indirectly (even if you don't believe they will be).

A good candidate system or service for your first attack will be:

- Unlikely to cause data loss
- Unlikely to break your application if the tested system or service goes down
- Already set up with redundancy

Some ideas are a proxy, a web cache, or maybe a load-balanced front end server.

Select one service that is well-supported by your team. Discuss with the team what the existing known problems are with the service and which one might be a good candidate as low-hanging fruit for a simple first attack. Set up a GameDay.

Think about past incidents and the losses incurred as a result; time, revenue, customer goodwill. Now think about how Chaos Engineering could have exposed the system weakness(es) that caused one or more of those incidents. Pick one metric to improve, something measurable and clearly related to a specific problem. That is what you want to design your first experiment around.

Your first attack should give you insight into your system, but what you really want is a quick win. Design and implement an attack that demonstrates:

- The usefulness of Chaos Engineering
- The ease of use of Gremlin to safely, securely, and simply perform Chaos Engineering
- The safe manner in which you can control the parameters of the attack including limiting the blast radius and magnitude
- How easy it is to stop an attack at any time

From here, draw a clear line from this experiment to a specific and costly past incident to show how Chaos Engineering would have prevented that incident. Doing this will go a long way toward helping management feel good about taking a chance as well as convincing them to allow you to grow your testing plan with Chaos Engineering. Here is an email template you can use and adapt to help write useful internal reports.

How to Sell the Discipline of Chaos Engineering

One of the most challenging things we can do as a part of an organization is try to convince the organization that change is necessary and beneficial. Organizations of all types tend toward stability and what is known. Building momentum for transformation or addition can be difficult, but it can be done. While challenging, being an agent of change is also one of the most rewarding parts of any job, especially when we are successful.

This section gives advice to help you become more confident and successful in achieving your goal of implementing needed testing improvements in your team and ultimately across your organization and perhaps even the entire company. We can't always guarantee success, but the following will set you up with the greatest opportunity for success possible.

Be a Great Employee

People who do their job well are listened to more consistently. Their suggestions pull more weight. Know your system. Know how it operates. Know your metrics; be able to recite key metrics off the top of your head in a meeting. Learn so much that you can answer important questions about your systems without having to look up the details. All this builds trust in your technical capabilities and authoritative perspective.

Know your coworkers, manager, and the people they work for. Know their pain points, what keeps them up at night. Understand corporate goals and make sure your work contributes to them in a meaningful way. This will help build trust in you as a collaborative person who is beneficial to the organization as a whole.

Do Your Homework

Research Chaos Engineering. Understand and be able to clearly express why it exists, what makes it important, and how Chaos Engineering can specifically help your organization create more reliable software. Find articles and white papers you can share that are specific to your pain points. Outline the benefits and risks for colleagues and especially for your boss: Both the risks of doing Chaos Engineering and the risks of NOT doing it. Be prepared. Learn how to speak the language of those you need to convince and present things as they need to hear them.

Build Trust With Your Colleagues

Know the people in your team and greater organization. Who are the ones you need to win over? What do they like, how do they communicate? Are they informal or formal? Think it through on a personal communication level for *each person* you need to convince. Take some time and apply a little sociology and psychology to your interaction.

Identify stakeholders. Who stands to benefit from Chaos Engineering when you get it implemented? Start teaching each one about it now, informally and casually. Share information as you learn or discover resources. Win them over gradually before you start talking to management.

You are going to need to be persistent. Be ready to work. Often convincing of the need for change or something new is hard and takes time. Any change is difficult, but one that is perceived as causing more work or introducing instability is even more difficult to introduce. This has been the argument against all forms of testing from the beginning. It is also wrong. We must counter the misconceptions and inject trust with truth.

Yes, Chaos Engineering testing requires a bit of new work. However, that new work saves our organization from a ton of problems by helping us find issues before they crop up in production and cause major incidents or outages. We are not introducing permanent instability, we are carefully testing what happens system-wide in a way no other testing can to determine how to best harden our systems to produce reliability even when problems occur.

Build an intuitive case by sharing quotes, articles, and case studies in a low key, relaxed manner over time. Keep demonstrating the benefit that Chaos Engineering can provide to your teammates. They may even decide that implementing Chaos Engineering was their idea, and help take ownership over the process!

Win Over Your Boss

You have won over your coworkers, or at least warmed them up to thinking about the idea of Chaos Engineering and the specific benefits to them, to the project, and to the greater organization. Now you are ready to make a more direct push.

Ask about, learn, and understand the procurement process in your organization. What has to be included in the request? Who signs off on spending for something like this? Know who you need to convince well in advance and make sure you dot the i's and cross the t's. Start thinking and talking about this now. Procurement is covered in greater detail later in this playbook.

How to Build the Business Case

In order to adopt a new discipline like Chaos Engineering that requires software, it's necessary to demonstrate that it improves a part of the business, often by helping the business make more money, save money, or reduce risk / ensure compliance.

For example, frame your Chaos Engineering suggestions in terms of how it will help meet and exceed goals that already exist, such as performance metrics like SLAs and SLOs that tie into an objective of greater engineering velocity so the business is more profitable. Think about real problems your organization struggles with today. Is Chaos Engineering the solution to any of them? Which ones and how?

Start with the problem, not the solution. Here's what's going on today. Here's how CE could prevent or mitigate or help solve it. Use anecdotes, yours or someone else's, to illustrate real frustrations without exaggerating. Appeal to authority - share inspiring tales from others already having success with Chaos Engineering, and not just the FAANG crowd.

Emotion can be compelling, but for some it can be a turn off. Think about when, whether, and how to appeal to emotion. If it fits, talk about how Chaos Engineering builds a feeling of trust in our systems and helps engineers sleep better at night. This is not appropriate for all audiences, but is powerfully effective for some. Again, know the people around you well enough to have a sense of how they prefer to communicate (and be communicated with!).

Every organization is averse to loss. Be sensitive to that loss aversion and make the idea of failure less risky. Talk about what happens if Chaos Engineering doesn't work (little to no negative impact) and compare that with the potential rewards if it does (more reliable systems, less downtime, fewer outages, happier customers).

Show that you have not only considered the pros, but also know the potential cons and arguments against. Have a prepared, gentle, and convincing answer for each of them. Be ready for some people to freak out; offer appropriate comfort to soothe their fears by discussing how to mitigate the risks they perceive.

Demonstrate the ROI

Give numbers whenever possible: Chaos Engineering will save our organization $X over Y months, reduce our incident rate by Z, or help us migrate to the cloud N times faster. Here are some useful thoughts:

- “By 2023, 40% of organizations will implement chaos engineering practices as part of DevOps initiatives, reducing unplanned downtime by 20%.” From *Gartner Predicts 2020: Agile and DevOps Are Key to Digital Transformation, Manjunath Bhat, George Spafford, Joachim Herschmann, Daniel Betts, Jim Scheibmeir, Keith Mann, 5 December 2019*
- Preventing one Sev 1 incident pays for the solution.
- Why CTOs And CIOs Should Care More About The Cost Of Downtime

How to Anticipate Objections

Anticipate objections. Have an answer for anything you can anticipate and be humble enough to say you don't know when a different question arises. Make this a dialogue by really listening and considering what everyone has to say.

The timing is bad

- When you wait, you leave money on the table. To the previous point, the ROI of CE is very high, so finding, fixing and verifying issues sooner earns you more money in the long run.
- See the cost savings from Qualtrics and the Gartner quote above (downtime = lost revenue, productivity, SLA fees, lost reputation)
- "The best time to plant a tree was 20 years ago. The second best time is now." --Traditional Chinese proverb

We're in the middle of a migration

- That's great! New cloud environments present new challenges. Be prepared to handle these by running through common failure scenarios. Ensure that your new systems and your existing team can handle anything thrown at them.
- Migrating to the Cloud Is Chaotic. Embrace It.

Existing system problems make CE scary--we have too much chaos now

- Chaos Engineering can help you prioritize. Which system problem is most likely to cause problems? Run CE attacks to see which one will have the largest impact on customers.
- As you migrate to modern systems (e.g., cloud, Kubernetes), check your dependencies to your old code base while you prepare for migration.
- To the earlier point, the ROI is high, it should be prioritized accordingly.
- Under Armour leverage Gremlin for their migration to K8s, Incremental Reliability Improvement and Updating the Industry's Reliability Practices

We don't have monitoring set up

- CE can help you set up your monitoring and alerting and tune your tools to avoid noise and correctly alert you.
- Chaos Engineering Monitoring & Metrics Guide and Preparing for Disaster

Assemble a Six to Twelve Month Roadmap

Here are some ideas that you might include in your personalized roadmap for your organization. These are steps that have worked well for several Gremlin customers and a sequence of milestones like these is likely to help you get your Chaos Engineering practice off to a good start.

First month:

- Get Gremlin purchased and the daemon installed. Test that the connection works as expected.
- Check that your monitoring metrics are tracking what you want to track.
- Run your first GameDay to test the use case you identified earlier in the How to Identify a Use Case Within Your Organization section.
- Write and distribute a report on the results of your first GameDay.
- Schedule your next two GameDays.

Third month:

- Identify and prioritize two applications that you think are among your weakest or highest risk applications.
    - Use data from the past year to help you prioritize by showing which applications had the highest number of after-hour incidents, which affected the most or most important customers, and/or the cost of downtime.

    - Gather baseline metrics for these two applications and think about which should be prioritized for improvement.
    - Retain this information for future use.

- Set a schedule for the rest of the year: We will run X GameDays by Y date at a cadence of Z GameDays per month.
- Pick an attack where you have already learned something and implemented system changes. Re-run that attack to confirm your solution was effective.
- Write and distribute a report that clearly demonstrates this fact. If it does not confirm but shows problems that continue to exist, prioritize fixing those problems and write a report that shows how Chaos Engineering saved you from trusting a fix that was not valid. Write reports like these at least quarterly.

Sixth month:

- Update or write runbooks containing information about which systems and metrics to check, where to look when problems arise, and how to address those problems. Stay high-level, giving some guidance to engineers while allowing them to focus on specifics.
- Schedule and run your first FireDrill.
    - This is a planned outage simulation where part of your team will run a chaos experiment and monitor (and halt the experiment if things go awry) while the other part of the team knows nothing about the experiment design and will be tasked with solving the problem, just like in a real incident.

    - The goal is to give your team some practice so that when a real incident occurs they know what to do and how to proceed calmly and effectively, solving the problem more quickly.
    - Update your runbooks as needed with information learned during the FireDrill.
    - Write and distribute a report sharing specific things your team learned and how the organization has benefited from this FireDrill.
- Set a schedule for running more FireDrills over the next six months: We will run X FireDrills by Y date at a cadence of Z FireDrills per month.

First year:

- Collect specific metrics showing system reliability improvements that are the direct result of Chaos Engineering activities over the past year. Use these to write an annual Chaos Engineering report showing the business value and ROI, comparing time and money spent to time and money saved. There are excellent examples of these in the Adoption Guide.
- Plan how to grow the practice. Have you been running in production or just test/staging deployments? Are there experiments that you have run repeatedly that are good candidates for automating, perhaps in your CI/CD pipeline, to help prevent regressions? How many applications or systems are you testing? What can you grow incrementally and naturally and who else in the organization can help you do so?

What the Procurement Process Looks Like

In most organizations, the procurement process involves multiple steps and looks something like this:

1. An employee has identified a business case that aligns to a technical initiative in the organization, but pursuing it requires additional tooling, so they make a request.
2. A purchasing agent reviews the request and looks for solutions. This agent is typically not the employee's manager or anyone who knows the employee or the wider context...unless that information is carefully supplied. Make sure you inform the agent of your research, the context for your need, and your preferred vendor with costs. Also, since Gremlin is the only SaaS option for Chaos Engineering, you should make that known.

There are a few things you need to know when entering the procurement process. First, you need to know who makes the procurement decisions and understand their incentives.

Understand the responsibility that your boss assumes when they make a software purchase. The organization expects that your boss, along with you, has done their due diligence, researching available tools and options. Your boss's reputation is on the line as well as yours, so it is important that you both know as much as possible about what you are requesting.

Be concrete when asking. Specific requests are more likely to be approved. Make sure what you are asking for aligns with existing team objectives and expectations and show how what you are asking to do furthers them. Have a solid logical argument, with data, research, and stats to back up the case that Chaos Engineering will benefit your organization's goals. Use data - talk about costs vs rewards, ROI, tie technical and engineering work to business value, such as raising revenue, increasing feature velocity, or lowering cost. Always sell the value.

Don't overwhelm. Present a logical and safe progression that starts with a simple proof of concept, then a safe pilot program and presentation of results before moving on to more significant investment and implementation.

To that end, have two proposals ready: a big request and a small request. Start with the big one. If you get it, fantastic! If not, listen to why and make the small request showing how those reasons don't necessarily apply to the smaller option. The small request will start things and will likely prove so valuable with data *from your organization's systems* that eventually you can expand and get all you wanted with the big request (and maybe more!). Propose the small request as a trial run for a year.

3. The purchasing agent asks questions of potential vendors known to offer solutions specific to the employee's stated need. They will probably do this regardless of whether you supply a preference in #2, but supplying a preference sets a standard they will subconsciously use in their overall evaluation process. The agent's incentive is to get the best price while acquiring a tool that will do the needed job. For the best results, be clear about what you need and precise about specifications.
4. The purchasing agent requests quotes from a subset of vendors who replied to questions. Steps 2-4 may be repeated if quotes don't appeal to the agent. Again, you must set expectations in advance to maximize the potential for a successful outcome.
5. The purchasing department creates a purchase order (PO) to procure the solution chosen by the agent. If you are a part of the process, you are less likely to be surprised here.

Only after these steps are complete will the employee have access to what was requested. The process generally takes 4-8 weeks.

During your proposal preparation time, think about what the rest of this section advises.

Pay attention to context and environment and choose your moments. If the person you need to convince looks frustrated or just got out of a difficult meeting discussing budget issues, that is not the time to bring up new spending and ideas and not the time to request a meeting. Find positive moments and always share in positive ways.

Frame things you say to show that you are thinking of mutual benefit for team and organization success.

You might try to use a retrospective from your organization as a springboard to demonstrate a case where having performed a specific type of CE test earlier could have prevented the failure/outage/problem.

If you sense that it would be useful within the dynamics of your team and organization, set up Gremlin Free and run a simple experiment with your boss and teammates in the room to show ease of use and potential. You can do this on the provided sample system or maybe even on a less-important component of your system.

If you get a hard "no," find out what it would take to change that.

Here is a link to a presentation template that you may find helpful.

Prepare now to justify the expense at the end of the procurement contract. Take careful note of potentially useful details and be ready to show how the tool helped reduce outages, increase reliability, and was a good return on investment.

How to Build for the Long Term

Getting approval and getting started are really just the first steps in a journey of learning how to make your system more and more reliable. Ultimately, the goal is bigger. We want to improve the performance of our team's system(s), but we also want to expand and help other teams across our entire organization.

To do this, we need to set ourselves up for success with that initial project, but we must do so thinking about how that success will lead the organization toward a desire to expand the use of Chaos Engineering in the pilot project, across other services, systems, and teams.

This is the time to shift from ideas and philosophical discussions to implementing a project - talk about practical implementation and results with timeframes, goals, and potential. With your boss, draw up real plans and a solid proposal that is obviously thought through.

One of the first things you want to consider when designing chaos experiments is the metrics you use. Having clearly-defined trackable metrics that can be reported back to management will validate the time and money spent. Crafting an email after each experiment or set of related experiments can help you clearly communicate the benefit received from running them.

A large Gremlin customer recently had a flawless launch of a new product. The executive team was so pleased they said that whatever launch checklist the team used for that successful launch was now the standard. Chaos Engineering was on that checklist and was part of the success story.

Ultimately, you want to expand thoughtfully. Doing this requires buy-in from multiple stakeholders. They must be shown the benefits of Chaos Engineering if adoption rates are to spread across the organization. So, show the benefits. Consistently. Honestly. In most cases, that will be a huge assistance in making Chaos Engineering a normal part of system testing plans, launch checklists, and regular reliability work.

Conclusion

Congratulations, you have successfully navigated one of the most political and perilous processes inside any organization! This is your first step and it is a big one. Take a moment to celebrate and enjoy.

Then, read our Adoption Guide to learn the practical aspects of getting started with Chaos Engineering, perhaps by signing up for a Gremlin Free account and doing some initial testing and research now.