THE ESSENTIALS GUIDE TO SRE

Key Principles and Practices for Production Teams

BLAMELESS

TABLE OF CONTENTS

WHY SITE RELIABILITY ENGINEERING?

In the world of technology, the stakes have never been higher. The move to the cloud and microservices to maximize agility has created unprecedented competitive threats, as companies race to provide the most cutting-edge services. As distributed systems become increasingly complex, the scale of "unknown unknowns" increases. On top of this, customer expectations for reliability are sky-high.

The cost of downtime is catastrophic, with competitors eager to adopt customers with unmet needs. For some companies, this number is considerably higher; for example, **Amazon lost approximately $90 million during their Prime Day outage in 2018**, and the outage only lasted 75 minutes.

Organizations need to prioritize reliability so they can innovate as quickly as possible on top of a strong foundation that won't compromise the customer experience. This will become even more critical as more businesses move toward distributed systems with high reliability requirements.

That's where site reliability engineering (SRE) comes in.

The SRE function is growing quickly (30-70% YoY growth in job listings), but there is not enough skilled talent in the market to meet this demand. It's not only important to get better at hiring SREs, but also adopting the practices and mindsets required for production excellence.

With the shortage of SREs for hire, what can you do to ensure your service's reliability? To answer this question, you'll need a deeper understanding of what SRE actually is.

According to Gartner, the average cost of downtime is

$300,000

per hour.

WHAT IS SRE?

SRE is a practice first introduced by Google in 2003 that seeks to create systems and services that are reliable enough to satisfy customer expectations. Since then, many large organizations such as **LinkedIn** and **Netflix** have adopted SRE best practices. In recent years, SRE has become more widely adopted by many organizations globally, with a focus on reliability and resilience, particularly in response to exponentially growing customer expectations and systems complexity.

SRE is firmly based on a customer-first mentality.

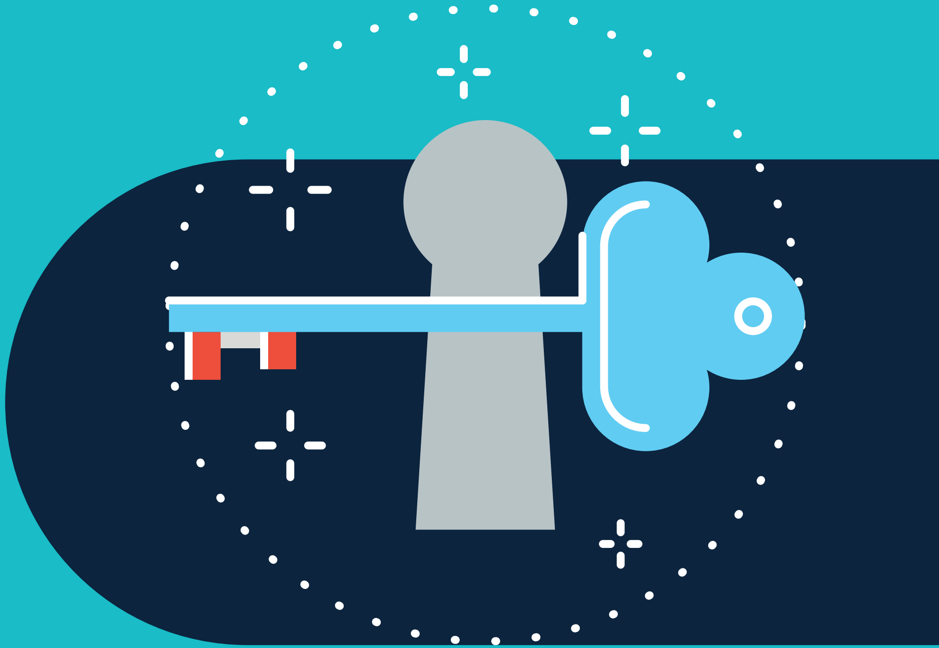
This means that SRE efforts are all tied to customer satisfaction, even if the customers using the service are actually internal users. Each decision is made based on how it will impact customer satisfaction. Teams work together to determine which factors and experiences affect customer happiness, measure them, and set goals. They balance meeting these reliability goals with the innovation velocity required to stay viable in an increasingly competitive digital landscape.

Key Tenets of SRE

SREs and teams that have adopted SRE best practices and several key tenets of SRE.

According to Google, these key tenets include:

- Ensuring a durable focus on engineering
- Pursuing maximum change velocity without violating a service level objective (SLO)
- Monitoring, including alerts, ticketing, and logging
- Emergency response
- Change management
- Demand forecasting and capacity planning
- Provisioning, efficiency and performance

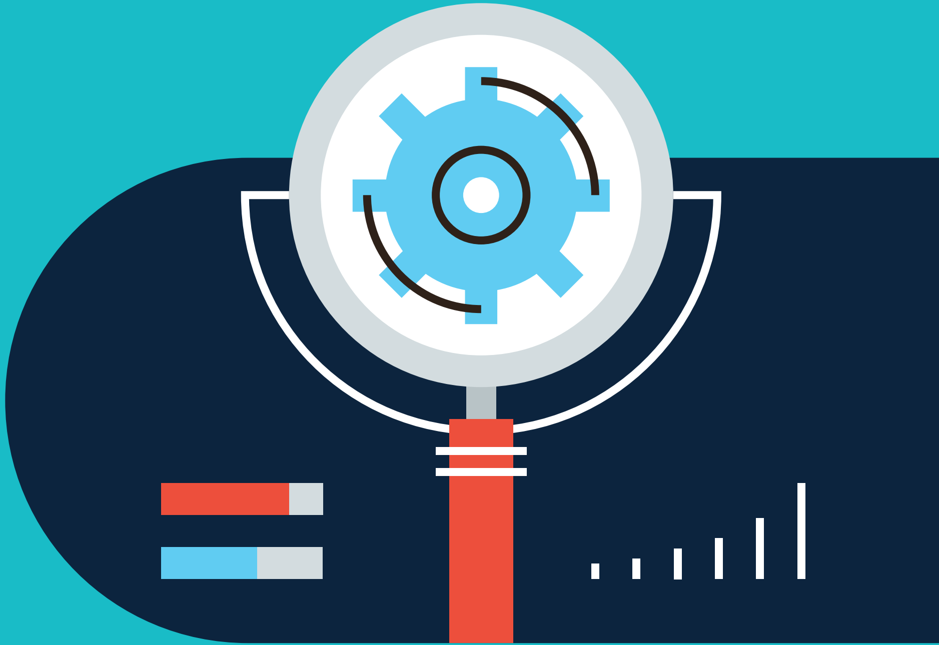


46% of the tenets can be applied out-of-the-box for most software teams in the enterprise, but the rest require customizations or won’t make sense for the vast majority of organizations. The important question to ask yourself is how these tenets fit in with what you’re already doing, and how your teams can improve.

UNDERSTANDING HOW SRE FITS INTO YOUR OPERATIONS MODEL

A common early mistake in adopting SRE best practices is assuming you'll need to rip out and replace your current procedures. This simply isn't true. In fact, SRE can work as a complement to both DevOps and ITIL methodologies.

The trick is to ensure that regardless of your organization's different operating models or toolchains, there is shared visibility, communication, and collaboration across teams. This will allow your disparate teams to stay aligned while using the best practices from each methodology.



HOW SRE WORKS WITH DEVOPS

Think of SRE as the practice that brings life to the DevOps philosophy. The core principles of DevOps and SRE are nearly identical. According to Google's Coursera course on SRE, "class SRE implements DevOps," meaning that SRE is a method that can achieve the goals of DevOps.

Here are the 5 DevOps principles and how SRE achieves them:

1. **Reduce organizational silos**: SRE helps by sharing ownership across developers and production teams, and creating unified infrastructure.
2. **Accept failure as normal:** Blameless postmortems are an SRE best practice that ensures that all incidents are used as learning opportunities. SRE also creates a safe space and guardrails for failure through SLOs and error budgets.
3. **Implement gradual change**: This is done by canarying rollouts to a small subset of customers before allowing all users to interact with new features. Smaller changes are easier and safer to dissect and iterate on.
4. **Leverage tooling and automation**: SREs work to eliminate toil by measuring it and creating automation to do repetitive tasks without needing human intervention. This way, humans can focus on higher-value work.
5. **Measure everything**: SRE specifically focuses on measuring toil and reliability to make sure that both customers and software teams are happy with the service.

With these common principles defined, it's easy to see how SRE and DevOps are the perfect fit, with SRE codifying practices that make it easier to achieve the promises of DevOps. If you're set up with DevOps and want to take the next steps towards SRE, check out our ebook: Bridging the Gap: DevOps to SRE!

HOW SRE WORKS WITH ITIL

In practice, ITIL and SRE can also make for a great combination. The first reason why is simple: every organization wants happy customers, and ITIL and SRE can help different functions work together to make that a reality.

Embedding reliability throughout the software lifecycle can ensure a higher rate of customer happiness. With the newest revision of ITIL, which introduces seven guiding principles, SRE and ITIL align even more closely

The Seven Principles of ITIL 4

1 **Start Where You Are**: Adopting SRE best practices is not one-size-fits-all, and everyone starts somewhere. Taking the first steps and implementing and iterating as you go is what matters most.

2 **Keep it Simple and Practical**: In the Google SRE book's chapter on simplicity, it states "Unlike just about everything else in life, 'boring' is actually a positive attribute when it comes to software! We don't want our programs to be spontaneous and interesting; we want them to stick to the script and predictably accomplish their business goals." Simplicity in both software and business operations streamlines communication, increases velocity, and helps ensure that reliability isn't compromised. Less is more.

3 **Optimize and Automate:** One of the goals of SRE is to automate toil-heavy processes, and free up developer time to focus on innovation instead of unplanned work. This optimizes workflows and allows new features to ship faster.

4 **Progress Iteratively with Feedback:** SREs set alerts for the most important and user-centric metrics.
The metrics, alerts, and SLOs they're tied to are all iterated upon to better satisfy customer needs. SRE retrospectives provide the perfect opportunity to review and revise the effectiveness of practices.

5 **Collaborate and Promote Visibility**: SRE is culturally collaborative. It focuses on a blameless work culture that values learning from failure, and trusting that each team member is doing what they think is best for the organization.

6 **Focus on Value**: Without customers, there is no value in software. Business value is created when customers get what they need from a product. SRE best practices ensure that the product is reliable enough to provide value to the customers by looking at the experiences most important to the customer. Thus, they provide significant value to the organization in helping to drive shared focus.

7 **Think and Work Holistically**: By breaking down silos and focusing on scalability and reliability on a holistic level, SREs are able to provide significant benefits in maturing the organization. Business-wide success is in the hands of every team member, and SREs work to make sure that the company's product, systems, and procedures are resilient enough even when accounting for the many sources of unreliability. For a visual on how SRE, DevOps, and ITIL's best practices can be used in conjunction with each other, we created a handy graph.

ITIL vs DevOps vs SRE

	ITIL	DEVOPS	SRE
Philosophy & Culture	Align IT with business needs to create a symbiotic relationship Command-and-control and process-driven to mitigate risk	Improve teamwork and eliminate silos Aims to create alignment and minimize silos between development and operations Often oriented toward helping teams improve velocity and quality of deploys	Eliminate toil, design for operability Treats operations as a software problem to maximize efficiency Ideal to support distributed services at scale that need to be hyper-reliable
Key Practices & Tooling	Capacity planning Service catalog / CMDB Problem management Change management / advisory board	Capacity planning On-call Microservices CI/CD Infra as code Monitoring and logging Comms & collaboration	Same as DevOps key practices, as well as: Progressive rollouts SLOs & error budgets Observability Chaos engineering
Teamwork	Traditional model of centralized process and visibility. Work is typically queued ('waterfall') Incidents routed through central NOC team	Dev and ops increasingly share the same process and tooling throughout the entire service lifecycle Typically, this means devs go on-call for what they build, but may engage ops for L2 support	SREs often act as consultants to establish reliability-oriented practices Software Eng and SREs' roles converge around shared processes and outcomes
Key Measures	Availability, # incidents, # escalations, etc.	Availability, deployment frequency, etc.	SLOs as well as availability, deployment frequency, etc. Error budgets

Whether you identify as a DevOps or ITIL shop, your organization has something to gain by following the principles of SRE. Let's dive into what exactly these principles entail.

PRINCIPLE NO. 1: **Create a Mindset of Resiliency**

Relisilency isn't something that just happens; it's a result of dedication and hard work. To reach your optimal state of resilience, there are some crucial SRE best practices you should adopt to strengthen your processes.

Incident playbooks

As you know, failure is not an option… because actually, it's inevitable. Things will go wrong, especially with growing systems complexity and reliance on third-party service providers. You'll need to be prepared to make the right decisions fast. There's nothing worse than being called in the wee hours of a Sunday morning to handle a situation where thousands of dollars are going down the drain every second.
Your brain is foggy, and you'll likely need time to adjust to the extreme pressure of a critical incident. In these cases (and really, all cases where an incident is involved), incident playbooks can help guide you through the process and maximize the use of time.

According to Chris Taylor at Taksati Consulting, good incident playbooks help you cover all your bases. They typically include flowcharts and checklists to depict both the big picture and the minute details, a RACI (responsible, accountable, consulted, informed) chart for each step, and a list of environmental influences that are unique to your system. To create your incident playbook, Chris recommends aggregating the following information:

- An inventory of relevant tools
- The right personnel/subject matter experts to engage in response
- Knowing the problem to solve, or the workflow you're trying to document
- Current state (whether this is a new process, or updating an old one) By developing

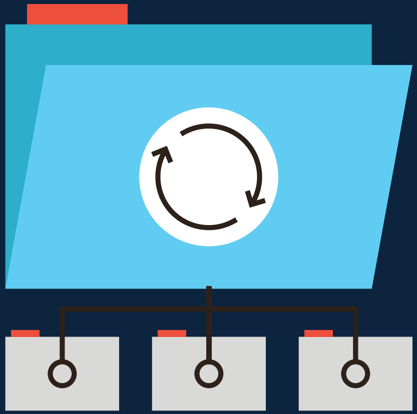
By developing incident playbooks and practicing running through them, you'll be more prepared for the inevitable.

Create a Mindset of Resiliency: Change Management

Change management is often done haphazardly, if at all. This means that organizations are unable to manage the risk of pushing new code, possibly leading to more incidents. Rather than employ ITIL's arduous CAB method, SRE seeks to empower teams to push code according to their own schedule while still managing risk. To do this, SRE uses SLOs and error budgets.

SLOs, or service level objectives, are internal goals for service availability and speed which are set according to customer needs. These SLOs serve as a benchmark for safety. Each month, you have a certain allowable amount of downtime determined by your SLO. You can use this downtime to push new features. If a feature is at risk for exceeding your error budget, which is the space before your SLO is breached, it shouldn't be pushed until the next window. If the feature is low to no risk to your SLO, then you can push it safely and confidently.

Each month teams should aspire to use the entirety of, but not exceed, their error budgets. This way, your organization can optimize for innovation, but do so safely without risking unacceptable levels of customer impact.



Create a Mindset of Resiliency: Capacity Planning

Black Friday outages, scaling, moving to cloud… all of these big events required heightened capacity planning. If you don't have enough load balancers on Black Friday or Cyber Monday, you might be sunk.
Or, if your company is simply growing quickly, you'll need to adopt best practices to make sure that your team has everything it needs to be successful.

There are two types of demand that require additional capacity: organic demand (this is your organization's natural growth) and inorganic demand (this is the growth that happens due to a specific marketing campaign or holiday. To prepare for these events, you'll need to forecast the demand and plan time for preparing more resources.)

Important facets of capacity planning include regular load testing and accurate provisioning. Regular load testing allows you to see how your system is operating under the average strain of daily users.

As Google SRE Stephen Thorne writes, "It's important to know that when you reach boundary conditions (such as CPU starvation or memory limits) things can go catastrophic, so sometimes it's important to know where those limits are."

If your service is struggling to load balance, or the CPU usage is through the roof, you know that you'll need to add capacity in the event of increased demand. That's where provisioning comes in.

Adding capacity in any form can be expensive, so knowing where you need additional resources is key. It's important to routinely plan for inorganic demand so you have time to provision correctly. The process of adding capacity can sometimes be a lengthy effort, especially if it's the case of moving to the cloud. You'll also need to know how many hands you'll need on deck for these high-traffic occasions.

Capacity planning is an important part of having a resilient system because when thinking about the allocation of resources, your team members matter. They need time off for holidays, personal vacations, or other unexpected events. When you fail to plan for time off, you won't have enough hands on deck to handle incidents as they occur. Denying people time off is obviously not the answer, as that will only lead to burnout and churn. So it's important to develop a capacity plan that can accommodate people being, well, people.

Johann Strasser shares four steps you can take to develop a capacity plan that will eliminate staffing insecurity:

1. Establish all necessary processes with the appropriate staff – from top management to team leaders. Decide how often you will need to revise/ revisit this process and make sure that everyone is in agreement on this.
2. Provide for complete and up-to-date project data and prioritize your projects. What projects are the most important, and which can be put on the back burner? Additionally, how long will each project take? You'll need this data to be able to move forward with accurate plans.
3. 3. Identify the capacities across your existing team, as well as your infrastructure and services. Is the team equipped and system architected in a way that minimizes performance regressions and protects efficiency and capacity?
4. 4. Consolidate the requirements (step 2) and the capacities (step 3). Identify underload as well as overload and try to balance them. So, now you've got the people and the process, but how can you learn and improve on your resilience?

For that, you'll need great retrospective practices in place that facilitate real introspection, psychological safety, and forward-looking accountability.

As Steve McGhee, SRE Leader at Google, shares, "Conducting blameless postmortems will enable you to see gaps in your current monitoring as well as operational processes. Armed with better monitoring, you will find it easier and faster to detect, triage, and resolve incidents."

Create a Mindset of Resiliency: Retrospectives Best Practices

When something goes wrong, it’s important to learn from it to prevent the same mistake from happening again. To do this, it’s important to craft and analyze retrospectives (aka postmortems, post-incident reviews, RCA reports, or whatever you like to call them). To have retrospectives worthy of analysis, applying SRE best practices will be key. In fact, retrospectives are a great place to begin your SRE adoption journey

As Steve McGhee, **SRE Leader at Google**, shares, “Conducting blameless postmortems will enable you to see gaps in your current monitoring as well as operational processes. Armed with better monitoring, you will find it easier and faster to detect, triage, and resolve incidents. More effective incident resolution will then free up time and mental bandwidth for more in-depth learning during retrospectives, leading to even better monitoring.

One of the most important elements of a retrospective, and of SRE as a whole, is the notion of being blameless. To learn from retrospectives, there needs to be total transparency. Opening up about mistakes can often be frightening, and requires a psychologically safe space to do so. Positive intent should always be assumed in order to foster the trust that allows for true openness. Blaming team members or defining people as the root cause for failure will only lead to more insecurity, covering up the important truths that retrospectives are meant to uncover.

To craft great retrospectives, there are four other best practices that will ensure your incidents are being used to their full advantage:

- **Use visuals in your retrospectives:** As Steve McGhee says, "A 'what happened' narrative with graphs is the best textbook-let for teaching other engineers how to get better at progressing through future incidents." Graphs provide an engineer with a quickly readable yet in-depth explanation for what was happening during the incident days, weeks, or even years later.
- **Be a historian**: Timelines can be invaluable for parsing through a particularly dense incident. Chat logs can be cluttered, and it's difficult to quickly find what you're looking for. Thus, it's important to have a centralized timeline that gives a clean, clear summary of the events. This also provides the context that helps relevant team members analyze what happened.
- **Tell a story**: An incident is a story. To tell a story well, many components must work together. Without sufficient background knowledge, this story loses depth and context. Without a timeline dictating what happened during an incident, the story loses its plot. Without a plan to rectify outstanding action items, the story loses a resolution.
- **Publish promptly**: Promptness has two main benefits: first, it allows the authors of the retrospective to report on the incident with a clear mind, and second, it soothes affected customers. Best-in-class companies like Google, Uber, and others have internal SLOs for publishing their postmortems within 48 hours.

Creating incident playbooks, utilizing change management and capacity planning, and following retrospective best practices will all contribute to your system's resilience, but that's not all that SRE seeks to do.

"In other words, building a retrospective practice will eventually enable you to identify and tackle classes of issues, including fixing deeply rooted technical debt. With time, you'll be able to practice SRE and make continuous direct improvements to your systems".

PRINCIPLE NO. 2: **Reduce Engineering Problems/ Innovation Blockers**

Happy engineers means happy customers, as engineers won't build the best products possible without support from the organization. There are two major ways that SRE can help brighten engineering's day.

Toil

One of the main focuses of SRE is automation. Toil is a waste of precious engineering time, and by SREs creating frameworks, processes, internal tooling/building tooling to eliminate it, engineers can get back to innovating.

Elimination of Tech Debt

SREs create accountability around retrospective follow-up action items to make sure that old issues aren't buried under new code. SREs also put together frameworks to help developers deliver better performing code, prioritizing what matters most to the customer experience. Pinpointing the tech debt build-up that hurts customer experience is important to guide refactoring initiatives and other practices to reduce tech debt. This establishes a baseline for healthy engineering practices to help minimize future accrual of tech debt.

Additionally, SREs invest in cultural change that prevents more tech debt from accruing in the future, while still making way for innovation.

Jean Hsu wrote about her experience refactoring Medium's codebase, and realized that the most important thing she could do for her team wasn't just to fix spaghetti code; it was to create a culture that fixes technical debt as it goes along, deleting old code as needed. Jean wrote:

"I realized that if I always did this type of work myself, I would be constantly refactoring, and the rest of the team would take away the lesson that I'd clean up after them. Though I did enjoy it myself, I really wanted to foster a longterm culture where engineers felt pride and ownership over this type of work."

SREs are often the cultural drivers for this sort of work, improving the way engineering teams function as a whole rather than simply going from project to project fixing bugs. These changes are long-term initiatives that spark growth and adoption of best practices for the entire organization. With this context in mind, adoption brings positive business benefits for everyone in the organization.

As you can see, SRE could positively impact each engineer's day-to-day productivity. In fact, SRE is not about tooling or job titles, and is rather a more human-centric approach to systems as a whole.

PRINCIPLE NO. 3: **Approach Systems from a Human Perspective**

Resiliency engineering as a practice looks at systems holistically, considering not only infrastructure but also human, process, and cultural factors.

Without adopting the culture and mindset behind SRE, you'll simply have new processes with no uniting value at the center to keep the initiative in place. Focusing on the human approach to systems requires reevaluating your organization's attitude towards three crucial things.

On-Call & Full-Service Ownership Practices

The notion of on-call is important in SRE for several reasons. It establishes clear ownership to ensure software problems are immediately addressed, and inherently incentivizes developers to ship more performant code. But while going on-call is now a fairly common practice, establishing a healthy, balanced process is crucial to prevent burnout.

Nobody can be on-call 24/7, especially when incidents during the on-call period actively disrupt engineers' personal lives. People need uninterrupted time away from work to be at their best, so on-call responsibilities need to be carefully monitored. If someone is waking up at 2 AM every night for a full month, there's something wrong; it's simply unsustainable. Additionally, more than one person should have to carry the burden. The whole development team should be empowered to be on-call so the responsibility becomes a shared one. This also incentivizes developers to ship better code to avoid getting woken up at 2 AM.

SRE best practices encourage a better proactive system, with a robust reactive system in place. Being proactive means fostering a community of constant learning and improvement. When your engineers are better prepared and learning from previous incidents, it's less likely that the same mistakes will be made again. This lowers the amount of incidents occurring as your SRE practice matures. From a reactive perspective, better incident management practices can allow for streamlined communication during an incident, and provide a foundation to treat incidents as "unplanned investments" as they become important learning opportunities.

Retrospectives thus give engineers a place to begin looking when the root cause of an incident is evading them. SRE gives those who hold the pager more agency.

Keeping Burnout at Bay

- Constant firefighting, especially with a tough on-call schedule, can leave engineers feeling burnt out. Over time, burnout leads to high turnover rates, meaning the senior engineering will need to pick up additional slack while new hires are ramped up. This only increases burnout, leading to a vicious cycle of dissatisfied engineers who have little capacity to think about improvements, and new hires who are clueless about where to begin.

- In this situation, the SRE approach would encourage improved visibility into engineering hours, on-call periods, and repeat incidents. Each of these issues directly contributes to burnout, yet many organizations aren't tracking them. By knowing which engineers have spent abnormaally high hours over an extended period of time, team leads can suggest vacation time to curb burnout. Knowing who has been on-call every weekend for the last month allows teams to better manage the rotation so everyone gets a break. Monitoring repeat incidents and incidents of a similar class can give insight into what's burning through engineering hours, as well as whether previous postmortems uncovered improvements or follow-up items that were not taken action on. These are issues that should promptly be fixed in order to give teams a break from firefighting, and more time for strategic work.

Celebrating Failure

Failure will happen, incidents will occur, and SLOs will be breached. These things may be difficult to face, but part of adopting SRE is to acknowledge that they are the norm. Systems are made by humans, and humans are imperfect. What's important is learning from these failures and celebrating the opportunity to grow.

One way to foster this culture is to prioritize psychological safety in the workplace. The power of safety is very obvious, but often overlooked. Industry thought leaders like Gene Kim have been promoting the importance of feeling safe to fail.

He addresses the issue of psychological insecurity in his novel, "The Unicorn Project." Main character Maxine has been shunted from a highly functional team to Project Phoenix, where mistakes are punishable by firing.

Gene writes "She's [Maxine] seen the corrosive effects that a culture of fear creates, where mistakes are routinely punished and scapegoats fired. Punishing failure and 'shooting the messenger' only cause people to hide their mistakes, and eventually, all desire to innovate is completely extinguished."

Fear Is an Innovation Killer, But Failure Is an Innovation Inspiration

Getting the most out of your teams and systems cannot be achieved if blame exists. Blamelessness is at the core of SRE.

To fully adopt this practice, you need to acknowledge that people are not a source of failure. Each team member is simply doing their best with the knowledge at hand, making the decisions they believe are right and in the best interests of the organization. Punishment or blame takes away the desire to try, fix, and continuously learn.

Fear is an innovation killer, but failure is an innovation inspiration. Creating safety and trust within your organization is key to fully realizing and unleashing your team's potential.

BEGIN YOUR SRE JOURNEY TODAY

Any organization can adopt SRE best practices, and it can begin in small increments. The most important change you will make will be the cultural one. As organizations are made of people, any organization can foster continuous learning, blameless culture, and psychological safety so long as its people are committed to a growth mindset. Once these cultural factors are in place, it becomes much easier to implement the practices, processes, and tools that scale that culture of excellence.

And if you need a guiding hand along the way, remember Blameless is here for you.