Blameless

EBOOK

Preventing Burnout with Reliability Practices

JANUARY 2024

Blameless

For any tech organization,

the largest cost center is in engineering salaries. Often far beyond tooling, marketing, or any other service, engineering teams represent the most significant investment your company can make.

Despite this, many organizations don't focus enough on the #1 factor impacting the return on that investment: burnout.

Some amount of fluctuation in staffing is inevitable. People will come and go for many different reasons. They'll have weeks of great performance and periods of slump, beyond anyone's control. Burnout is a factor that can exaggerate these slumps and accelerate departures. Fortunately, unlike many other factors, there are steps we can take to proactively reduce the chance for burnout.

Some symptoms of burnout

- Fatigue
- Stress
- Disinterest in work
- Pessimism about work
- Lack of focus
- Error-prone work
- Lack of curiosity

Some organizational effects of burnout

- Progress deceleration
- Sloppy work that requires fixing
- Low morale
- Frequent employee turnover
- Damage to brand reputation

In this eBook, we'll apply the principles and mindset of Site Reliability Engineering (SRE) to the problem of burnout. SRE is a software engineering discipline that recommends development and operations practices, gameplans around incidents, and cultural values to reduce toil, improve the reliability of your services, and make your customers and engineers happier. Check out our full eBook on the principles of SRE to learn more.

We'll look at some common causes of burnout and how SRE practices can alleviate them, while offering practical implementation suggestions for individual engineers, team leads and lower managers, and engineering division directors.
We hope this eBook sparks a dual journey of reduced burnout and improved reliability. By tackling burnout with this mentality, you'll get both a stronger engineering team and the many benefits of a stable, low-toil, trackable system. Let's strike out on these journeys together!

A quick note before we begin...

These solutions are impactful remedies to some crucial causes of burnout, but they aren't comprehensive. Before you can start tackling burnout in a meaningful way, you have to ensure you have the right foundation. That is:

Competitive compensation
to alleviate external financial stress and to materially prove their work is valued

Comprehensive accommodation
such as vacation days or work from home flexibility, to give them the environment they need to perform their best

A psychologically safe environment
to allow them to air their concerns and maintain a healthy working life

If these foundations aren't met, any other efforts you make to reduce burnout will always be insufficient. As you implement these other practices, take the opportunity to make sure you're providing this baseline to your employees.

Understand “good enough” with SLOs

One common cause of burnout is managers demanding too much from engineers. This often isn’t malicious, but results from managers attempting to reconcile demands made of them from higher management. If your manager is being asked for open-ended improvement on metrics like “uptime” or “deploys”, it can easily translate into a “more is always better” mentality.

Of course, we know that more isn’t always better. Most tasks experience severe diminishing returns after a certain point, as improvement takes more effort and the results are less noticeable. Getting a service from 99.999% uptime to 99.9999% uptime requires a monumental effort, but given that that represents 6 seconds of downtime a week being reduced to 0.6 seconds, end users aren’t likely to care, or even notice.

When engineers are asked to keep improving these metrics, regardless of everything else, burnout is likely to follow. Their work becomes less meaningful, significant, or impactful, while also getting harder and harder.

Service Level Objectives (SLOs) are SRE’s implementation of the idea that “perfect is the enemy of productivity”. An SLO starts with a key metric, one that you’ve determined directly reflects the happiness of users, known as an SLI. SLIs can be built from multiple smaller metrics, weighted based on their importance to critical user experiences. Once the SLI is built, the SLO is set to the level that represents users being happy. For example, you could have an SLI that represents the total time of all the steps involved in searching for a product on your site, and the SLO is set to the time where users would get frustrated and abandon their search due to slowness. To learn more about SLOs and SLIs, check out our full eBook.

The end result is a number where you can confidently say “if we stay below this number, our customers are happy”. This is a very liberating mindset to have! It means that if you’re comfortably below your SLO, you can move on from improving it further and focus on other things. If users are already happy, then further improvements will have diminishing returns and the effort could be better spent elsewhere.

The SLO practice can be applied to other areas to prevent burnout. Let’s look at three examples of how it could be implemented.

Individual Engineer

- When given a new task by your manager, work together to set a reasonable SLO on what “giving this task enough attention” looks like.
- Push back on open-ended goals like “as high as possible”, “as often as possible”, etc.
- Think about what metrics will be incorporated into the SLO. Which are more or less important?
- Make it clear that as long as you’re comfortably within the SLO, you’ll be moving on to other tasks.

Team Manager

- Apply the same questioning when given directions from above: when you’re asked for “improved development velocity”, how much improvement are they looking to see?
- Once you have these more specific requirements, build SLOs with your team that will keep them on track to meet them, but not pointlessly and stressfully exceed them.
- Use a tool to keep track of the status of the SLOs, and work together to come up with policies to switch priorities when they’re at risk of being breached.

Engineering Director

- Think about the greater goals of the organization that engineering is achieving.
- Work cross-departmentally to understand when marketing, customer success, etc. is expecting new projects, and build reasonable timelines backwards.
- Look at longitudinal patterns to see trends in productivity, and don’t overreact to what may just be regression to the mean.
- Build SLOs to encapsulate your expectations for major projects, and don’t push to exceed them pointlessly.

Toil can be a major factor in burnout. The exact definition of what is or isn't toilsome will vary between engineers, but common elements include:

- Repetitive work
- Extremely manual and tedious work
- Tasks that are auxiliary to more meaningful work, e.g. "every time I make a breakthrough on my project, I also have to do these 15 steps"
- Tasks that feel like they could be automated, but frustratingly haven't been

Not only does the nature of toilsome work often lead to unhappy engineers prone to burning out, being bogged down by toil leads to secondary sources of stress. Toilsome tasks are often not part of engineers' core job descriptions, and aren't what's being evaluated as their performance. As they feel more alienated from the work valued by management, burnout becomes more likely.

Fortunately, reducing toil is one of the major goals of SRE. In order to make systems more reliable, faster, and more consistent, SRE advocates building explicit processes and automating wherever possible. This also frees up the mental bandwidth for engineers to tackle more meaningful and impactful problems.

Let's look at how to apply this thinking to toil in three different positions:

Individual Engineer

- Eliminating toil starts with identifying it. Look for areas of your workload that are especially toilsome, and start tracking how much time it takes. This could be done as simply as starting and stopping a timer with different tasks.
- Surface high toil areas to your manager. Make it clear that this is a necessary part of your workflow, even if it isn't what you're being evaluated on, and make sure it's accommodated for in your evaluations.
- Discuss ways that these areas could be sped up through automation and process.
- Make detailed guides for yourself that cover common toilsome tasks. Even if they aren't automated, these guides will reduce the cognitive load of the tasks.

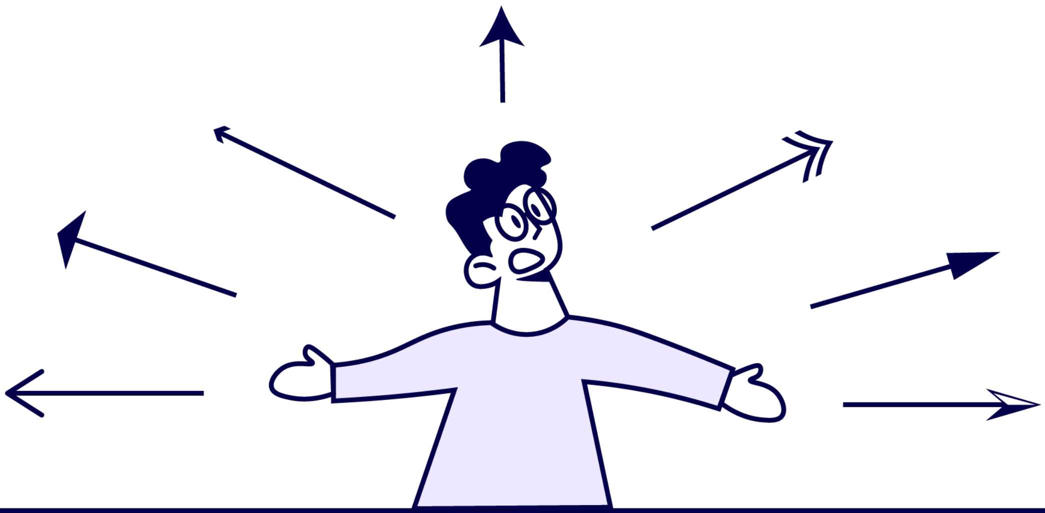
Team Manager

- Work with engineers to identify common toilsome tasks.
- Set aside cycles to explore how these tasks could be automated or eliminated, or, when this is impossible, build clear guides to reduce cognitive load.
- Try working through common processes with engineers, as you may identify steps that could be removed that the engineer is unaware of.
- Encourage breaking down silos and sharing tips between engineers that can remove toil.

Engineering Director

- Get reports across teams on toilsome areas of the system, then make investments to reduce them. If everyone reports that doing code review is a tedious and manual product, find a tool that can automate it.
- Make sure to test these implementations to ensure they're actually speeding up problem areas, and not just adding more overhead.

Smooth out misprioritization with big picture thinking



Do you feel like you're pulled in too many directions? One common cause of burnout is having priorities that suddenly appear or disappear, conflict with each other, jump up and down on your todo list, and generally yank you around. This isn't necessarily the result of mismanagement, but necessary in translating shifting strategic initiatives at the organizational layer to the workload of the individual layer. Regardless, having priorities shift wildly is a source of inefficiency, feeling underappreciated, and stress.

One major cause of rapid priority shift is not working from the right signals. In the modern world of complex microservice architecture, it can be hard to track your system's performance. You may just have a few basic outputs that don't necessarily reflect the overall user experience, which leads to decision-making based on irrelevant signals and noise.

Higher level metrics, like the frequency of incidents of different types, or user satisfaction and churn rate, often have insufficient data or aren't tracked and analyzed over long periods. Short term variance in these metrics can lead to overreaction to correct, even if it's just random oscillation, outlier spikes and valleys, and regressions to the mean.

With SRE, better prioritization and smoother workflows are encouraged through investing in observability and tracking. On one level, this involves investing in APM and other telemetry to get deeper insights to how your system is functioning. On another level, this means building processes to gather data and track more subjective and sociotechnical metrics, such as data around incidents, on-call shifts, and customer feedback.

All of this data is tracked longitudinally, and analyzed to determine baselines, trends, and expected variance. The goal is to have data that lets you detect when meaningful, customer-impacting trends are occurring, and not panic in response to smaller blips. Let's look at what that looks like at three levels.

Individual Engineer

- When priorities rapidly change, advocate for the work you'd been doing. Make sure your progress on previous priorities is recognized, and make explicit what won't happen if you're shifted to new tasks.
- Ask why the priorities need to change, and push back if you have contextual information that suggests a different interpretation.
- Even if they don't change their mind, understanding their motivations should make you feel less alienated from your new tasks.
- Remember that this isn't just a conversation about whether your new directive is sensible and viable – it very well could be – but about the effects of constantly changing direction. Make sure this jarring and exhausting factor is being compensated for.

Team Manager

- Set up observability and tracking into all the primary metrics of your teams' production: the health of services you're responsible for, velocity on team projects, incidents your team tackled, etc.
- Consolidate these metrics into dashboards that can be easily parsed and regularly surface them to upper management
- When allocating work, talk with each team member to get context on how major a focus shift new tasks will be and where they fall within existing priorities. Avoid frequent jarring reprioritization where possible, and acknowledge the stress it causes when necessary

Engineering Director

- When experimenting with new projects or directives, clearly lay out which metrics will determine their success, make sure those metrics are observable, and lay out a reasonable timeline for judging whether it was successful. Don't accelerate changes any faster, as this will translate to trash and burnout for engineers
- When you see some anomaly in data about your system, don't rush to respond. Analyze it in the context of greater system patterns. Is it actually a trend worth responding to, or just a blip of noise?
- Invest in tools to automatically gather more system data, log it, and consolidate it into dashboards. Work with your team leads to determine what reports they should be preparing and set time to review them as a group.

Reduce incident stress with robust incident management

Having to deal with incidents can be a major factor that leads to burnout. We define incidents as anything that pulls you away from your planned work until they're fixed – outages, slowdowns, bugs, and more. Some number of incidents will always be inevitable when you're building a complex and changing system. Engineers often work on-call shifts and expect to deal with incidents as some percentage of their workload. However, the nature of those incidents and how they're resolved can make all the difference when it comes to burnout.

Proper incident management is one of the biggest pillars of SRE. We could write a whole book about it – actually, we wrote two! Check out our complete guide to incident management, parts one and two, to get the full story of how to build up a robust and efficient practice. We'll focus here on priorities to alleviate burnout.

Like we discussed in the previous section, toilsome activities can lead to burnout. When incident management is ad-hoc and disorganized, it is often very toilsome, with many manual tasks and repetitive steps. Build efficient and ideally automatic processes to coordinate responders, assign diagnosis tasks, deploy a solution, and track results. Engineers can focus on just resolving the issue itself, which is much less toilsome and more fulfilling.

Repeat incidents are also a major source of frustration and burnout. Putting out the same fire day after day feels purposeless. To get ahead of repeat incidents, you need to invest in incident learning. Build retrospectives for each incident that summarize the solution, then keep them accessible as guides to make future incidents easier. Dive deep into the causes of incidents to make preventive changes. You want each incident to make your system more resilient to future incidents.

Let's see what people at three levels of management can do to improve incident management.

Individual Engineer

- Advocate for building a complete incident response playbook for you and your peers that go on call for the same services. Focus on how quickly investing the time to build it will pay for itself in speeding up incidents.
- Start recording what steps you take when you fix things in a document that your peers can access and comment on.
- Record what you did on each on-call shift, and how stressful it was subjectively, and way. Surface this to your manager to help have on-call shifts balanced by expected workload, and not just sheer number of hours.

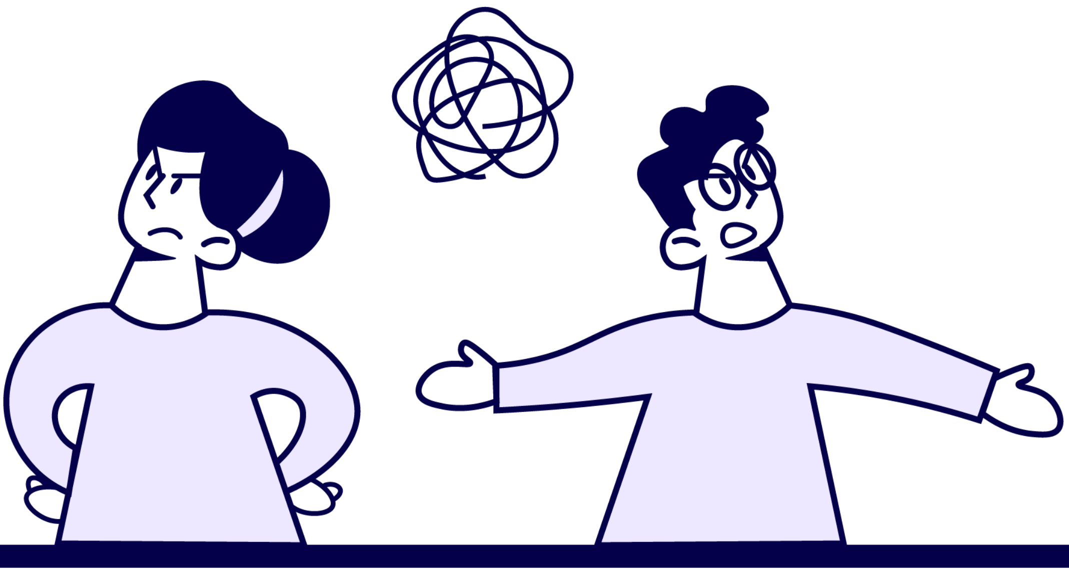
Team Manager

- Talk with your team members to find common types of incidents, and invest the time to build runbooks to handle them, automating where possible
- Spend time with your team to investigate the causes of each incident. When a systemic change could prevent the incident from occurring, build making that fix into your team's workload. Don't treat it as "bonus work", but make it as just as valued and accommodated as any other task.
- Track incident metrics across your team, such as frequency and severity, and surface them to higher management to advocate for more resources when necessary.

Engineering Director

- Invest in an incident management tool to automate and coordinate incident tasks, to make incidents faster to solve, less toilsome, and less frequent
- Create an expectation of incident learning. Work with team leads to build an organization-wide plan for retrospectives and follow-up tasks
- Make sure incident learning and resulting work isn't treated as just an addition to current expectations, but decrease other expectations to accommodate the time this requires
- Track trends of incidents across different service areas, and deploy more resources, like tooling, training, or more engineers, to areas that require it.

Build engineer agency with blamelessness



Burnout can result from an organization's culture and environment just as easily as it can from workload problems. If engineers feel stressed by hostility, ignored when they bring up grievances, and underappreciated in their efforts, they're extremely likely to check out, burnout, or outright leave.

SRE understands the importance of workplace culture in reliability. It prescribes cultural values that are just as essential as the processes. After all, when humans are stressed and afraid, more human error is sure to follow. Adopting these cultural values will also make a huge difference in preventing burnout among your engineers. We'll look at a few that tackle burnout most directly.

The first cultural pillar to adopt is individual agency among engineers. Build a culture where engineers feel encouraged to experiment, test ideas, and explore the limits of projects they're working on. This will result in less alienation from work, as engineers feel more responsible for their work and proud of their results. It will also result in a stronger and more resilient product, as more potential opportunities and challenges are explored.

As you grow in agency, another cultural pillar must be adopted simultaneously: blamelessness. You can tell this one is important to us – we named our whole company after it! Blamelessness means not seeing any individual action as the cause of an event, but looking deeper at systemic causes. For example, if an engineer accidentally deploys code that breaks a service, don't blame the engineer. Instead, look at the process that allowed them to deploy the code without sufficient review.

These two values in concert are the foundation of a culture with low burnout. Engineers feel encouraged to take ownership of their work and push it in directions that are fulfilling. They're confident that even if they make a mistake, it won't be held against them, but instead treated as an opportunity for the system to grow more resilient.

Let's look at how three different positions can help implement these values.

Individual Engineer

- When something breaks, don't cast blame at individuals – even yourself. Think instead about what in the system allowed the issue to occur
- Reassure your teammates that you're working on the same side, trying to tackle a systemic issue, rather than on opposite sides of a blame game
- Think about your projects not just as assignments you have to mindlessly execute, but something where you have agency to explore and experiment to make them more robust and resilient

Team Manager

- When things break, facilitate blameless investigation sessions to discover systemic causes. Don't be content with just attributing it to human error. Come up with ways to prevent that human error from recurring.
- Make it clear to your team members that making honest mistakes won't incur punishment. Encourage experimentation and ownership over avoiding any errors at any cost.
- Check in with your teammates to make sure they're airing any concerns they have

Engineering Director

- Create organization-wide cultural initiatives around individual agency. Celebrate individuals that experiment within their projects at all-hands engineering meetings, even if there were some bumps along the road.
- Make sure guidelines for engineering performance evaluations don't have language that excessively punishes honest mistakes. Complacency, avoidance, uncooperativeness, etc. demand correction much more than "causing" an outage (which has more relevant systemic causes anyways)
- Work with your team leads to build consistent guidelines for blamelessness that apply across the organization. This could include processes for investigating issues blamelessly, language to avoid when discussing incidents, and escalation processes to bring in mediators for complicated issues

Conclusion

We hope this guide encourages you to start your journey to reduce burnout – while making your system more reliable too! Whatever level you're at, you can take steps to make the work experience of you and your team less stressful and more productive.

Software engineering is a difficult job, and it's impossible to prevent all sources of burnout all the time. If you're fortunate enough to not be experiencing burnout now, you never know what other disruptive factors will come into play later. That's why it's important that you're tackling it proactively, building a foundation of policy and culture that will mitigate burnout sources.

At the end of the day, though, we encourage self-compassion to be your foundation in this journey. You won't always get things right the first time. Despite your best efforts, you or a coworker could still burn out. Remember that you're working with the best of intentions, acting out of genuine care for yourself and your organization. If you keep doing your best with this mentality, you'll surely make progress.

Building a low burnout, high resilience organization is a long journey, but it's one of many smaller steps. We congratulate you on taking one of your first steps today, and wish you luck on the road ahead.

Blameless

Make your next steps easy with Blameless

Blameless is an all-in-one incident management and SRE tool that can help you implement many of the processes we've suggested today. Our role-based incident guidance helps you coordinate and resolve incidents quickly. Then, our suite of incident learning tools makes sure you grow as much as you can from each incident with no manual overhead. Track what's important with our SLO manager and reliability patterns to keep from overreacting to noise.

Want to get up to speed on reliability without stressing out engineers? Check out a Blameless demo today!

GET A DEMO!