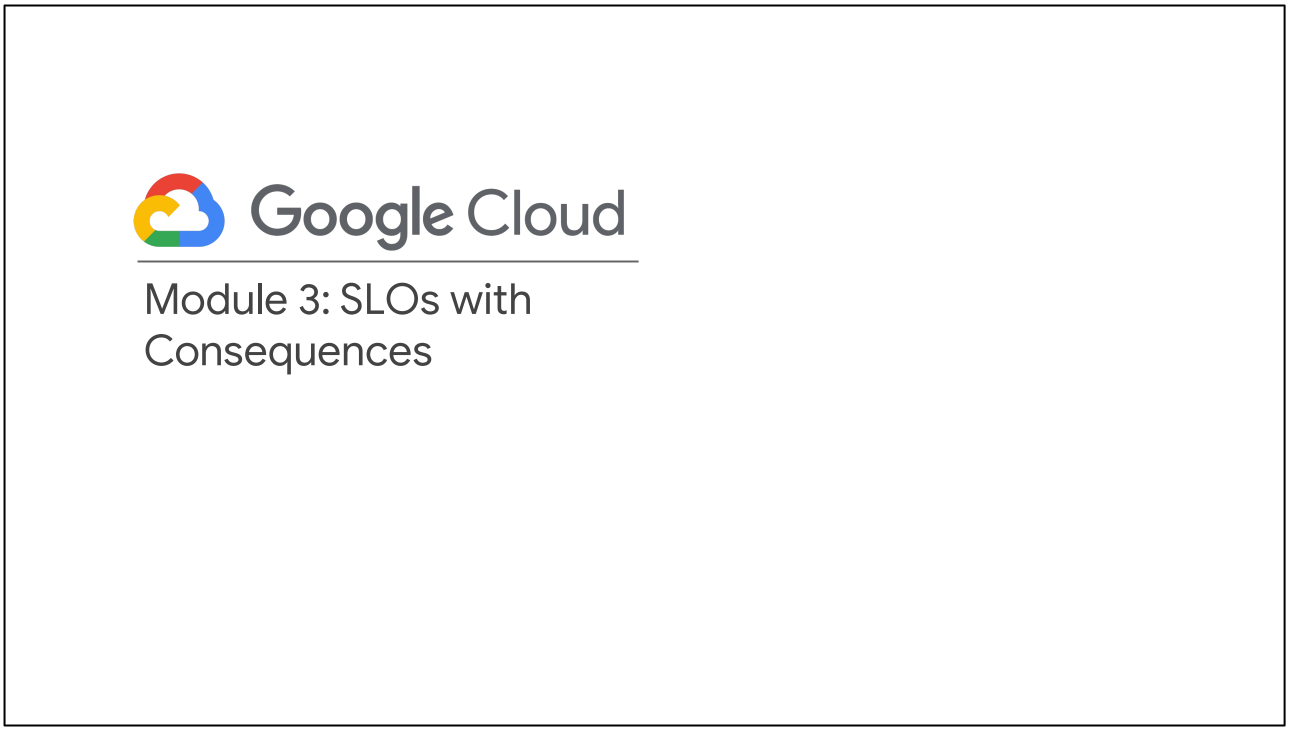
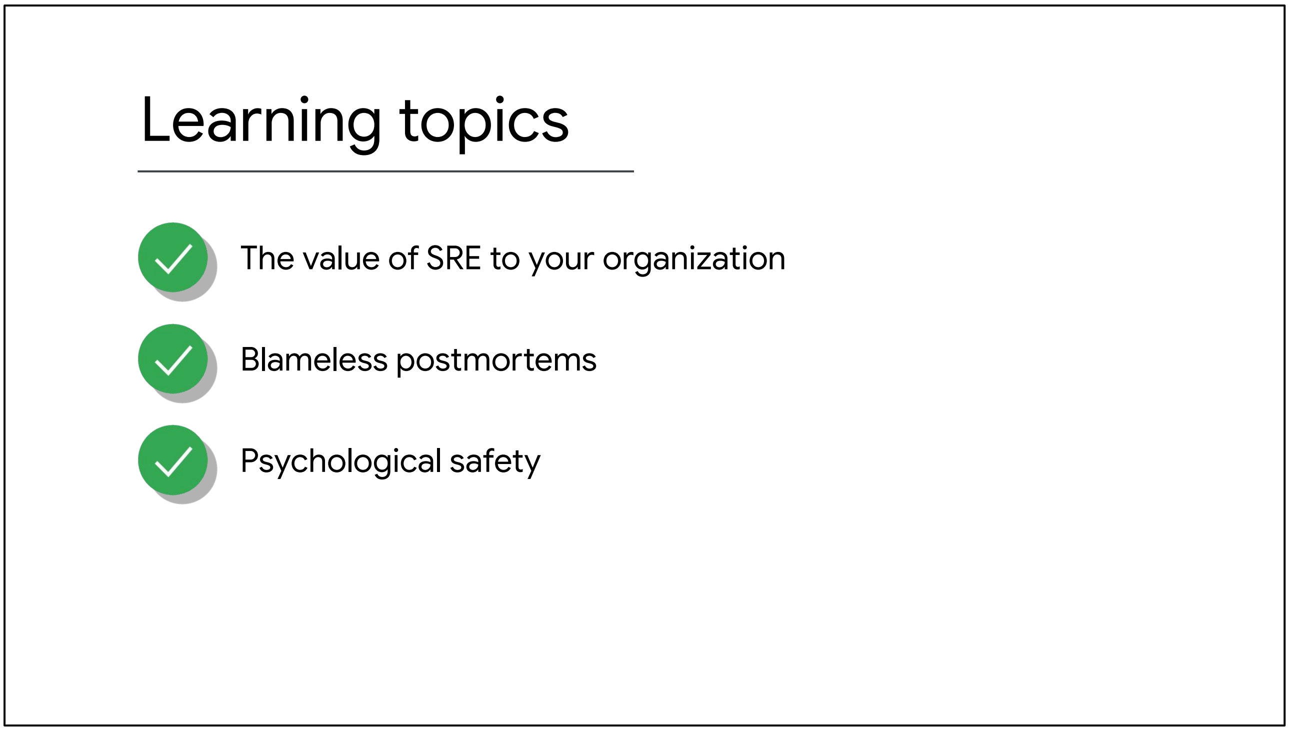


Google Cloud

# Module 3: SLOs with Consequences

If you think about SRE as a journey, the first step is to develop service level objectives, or SLOs, with consequences. The performance of your service relative to SLOs should guide your business decisions.

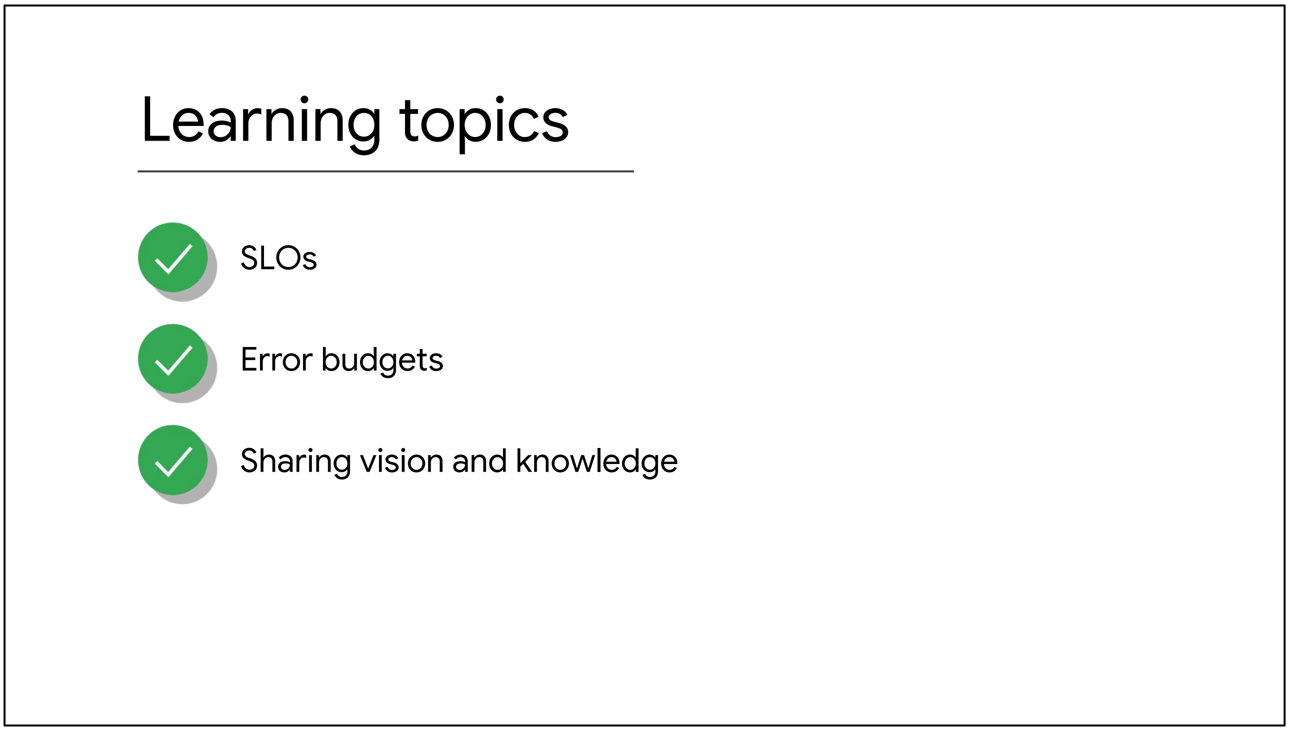


## Learning topics

- ✓ The value of SRE to your organization
- ✓ Blameless postmortems
- ✓ Psychological safety

In this module, we’ll first discuss the value of SRE to your organization. That is, what your IT team and its members will gain from SRE implementation and the role middle managers and top leadership play.

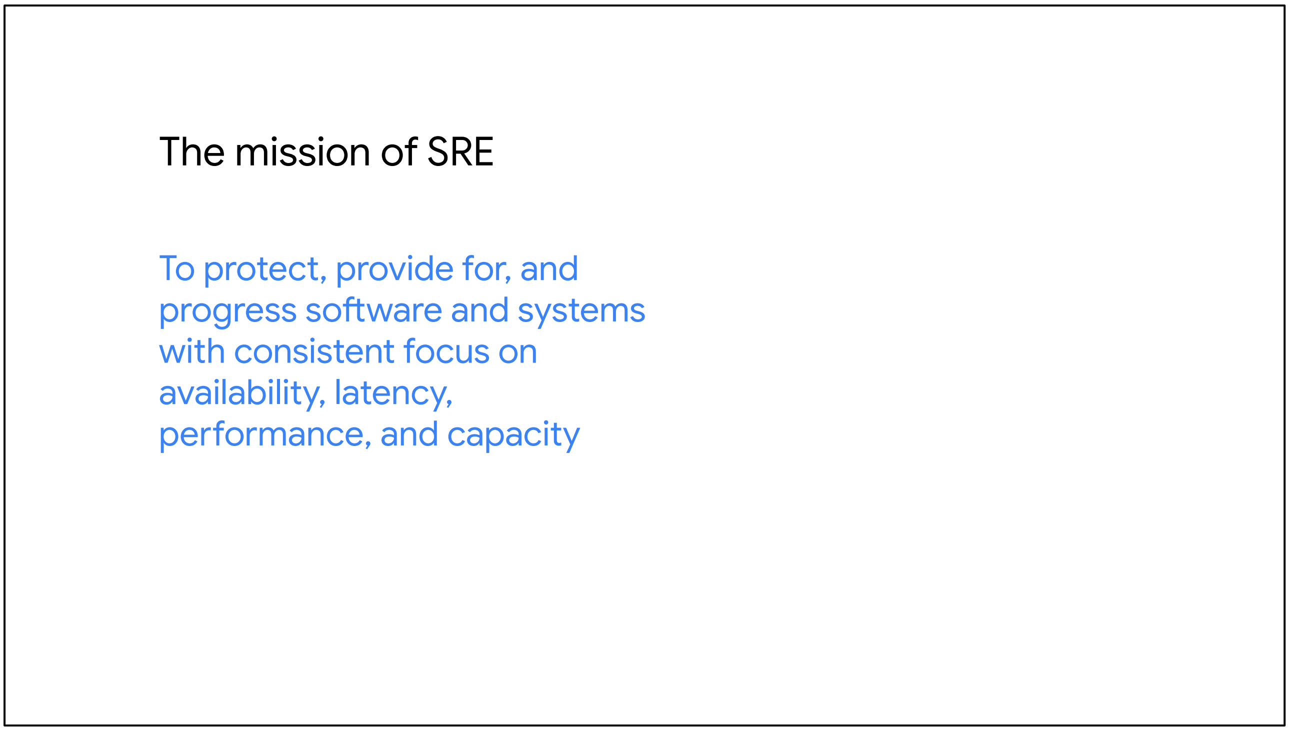
After that, we will cover the first two pillars of DevOps— *Accepting failure as normal* and *Reducing organizational silos,* and some of the SRE technical practices and cultural fundamentals that Google aligns to them. We’ll then discuss post mortems and how teams can, and should, foster blamelessness and psychological safety.



## Learning topics

- ✓ SLOs
- ✓ Error budgets
- ✓ Sharing vision and knowledge

You'll also learn about the practices of SLOs and error budgets, and the importance of sharing vision and knowledge across your team.



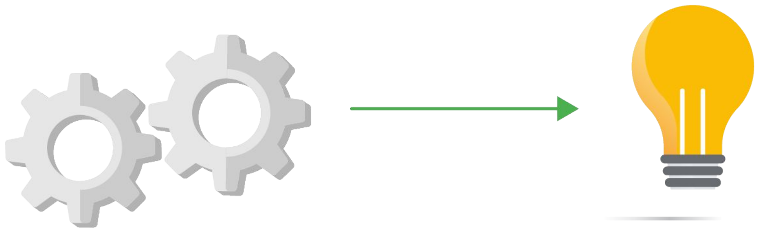
## The mission of SRE

To protect, provide for, and progress software and systems with consistent focus on availability, latency, performance, and capacity

Throughout this course, you'll learn about the key concepts of SRE and how they can be applied in your business. Before we explore each practice, it's important for you to understand how SRE adoption and implementation can provide value to your organization.

The mission of SRE is to protect, provide for, and progress software and systems with consistent focus on availability, latency, performance, and capacity. By adopting SRE in your business, you're looking after both your internal developer teams and the customers that consume your services. SRE practices support engineering teams working at a high velocity to release features. At the same time, they maintain practical goals and measures to prevent failures, so that your end users stay happy.

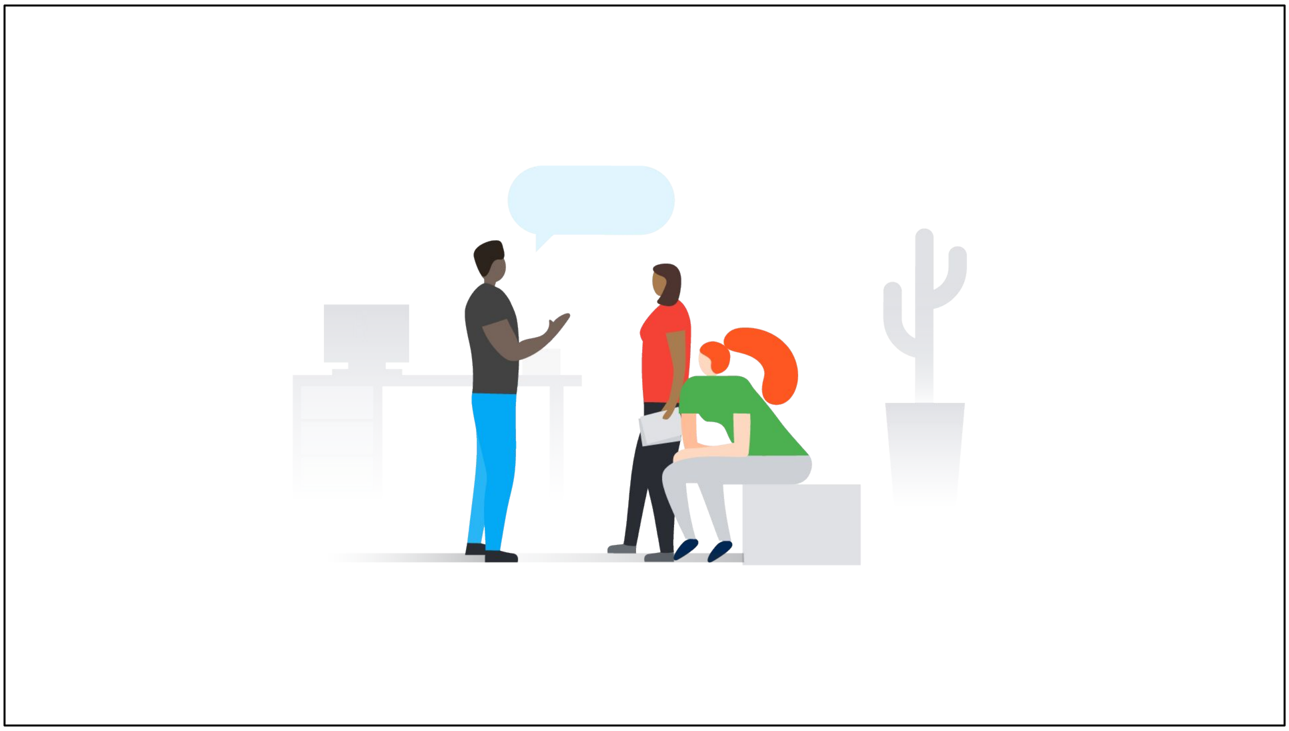
As you learned in the previous module, SRE teams focus on specific practices, such as automation, learning from failure, and reducing the cost of failure. By following principles like these, your teams will be less stressed with mundane, repetitive work. They'll be able to do more valuable work at a quicker pace, while also focusing on the reliability of your services.



Incentivize automation with time for creativity.

Launching and iterating allows teams to fail fast.

Automation, in particular, creates opportunities for upskilling and innovation. If you incentivize your engineers to automate processes where they can, they'll be making time for other creative projects that they otherwise might not have capacity for. SRE also accelerates innovation with the practices of launching and iterating, which allows teams to fail fast, learn from failure, and try again.

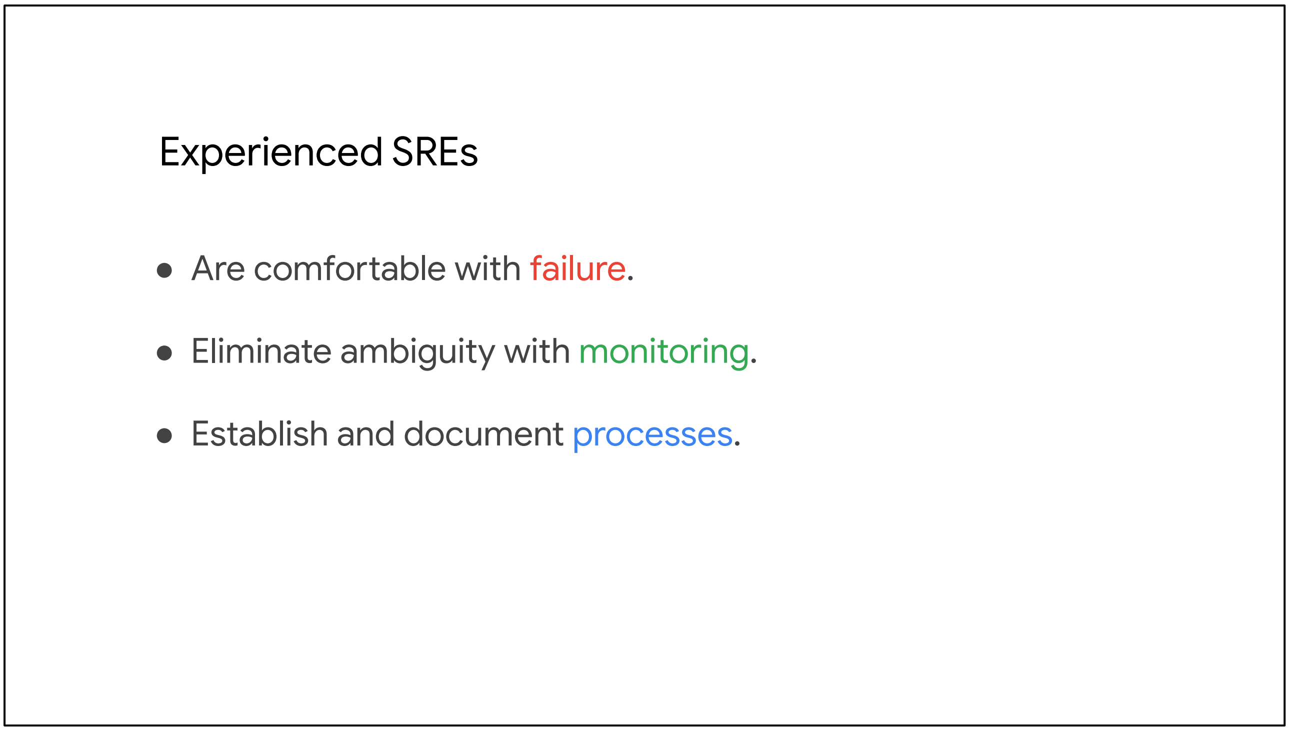


But remember that they will need cultural and organizational support to be successful in the long term. And that's where you come in! Understanding SRE practices and norms will help you build a common language to speak with your IT teams and support your organization's adoption of SRE both in the short and long term.

In the next video, you'll start learning about SRE practices in a bit more detail, starting with blameless postmortems.

When working at a high velocity in development and operations, or really at any speed, mistakes are inevitable. That’s why Google sees “accepting failure as normal” as a pillar of DevOps philosophy.

But how exactly can you implement this philosophy in your organization?

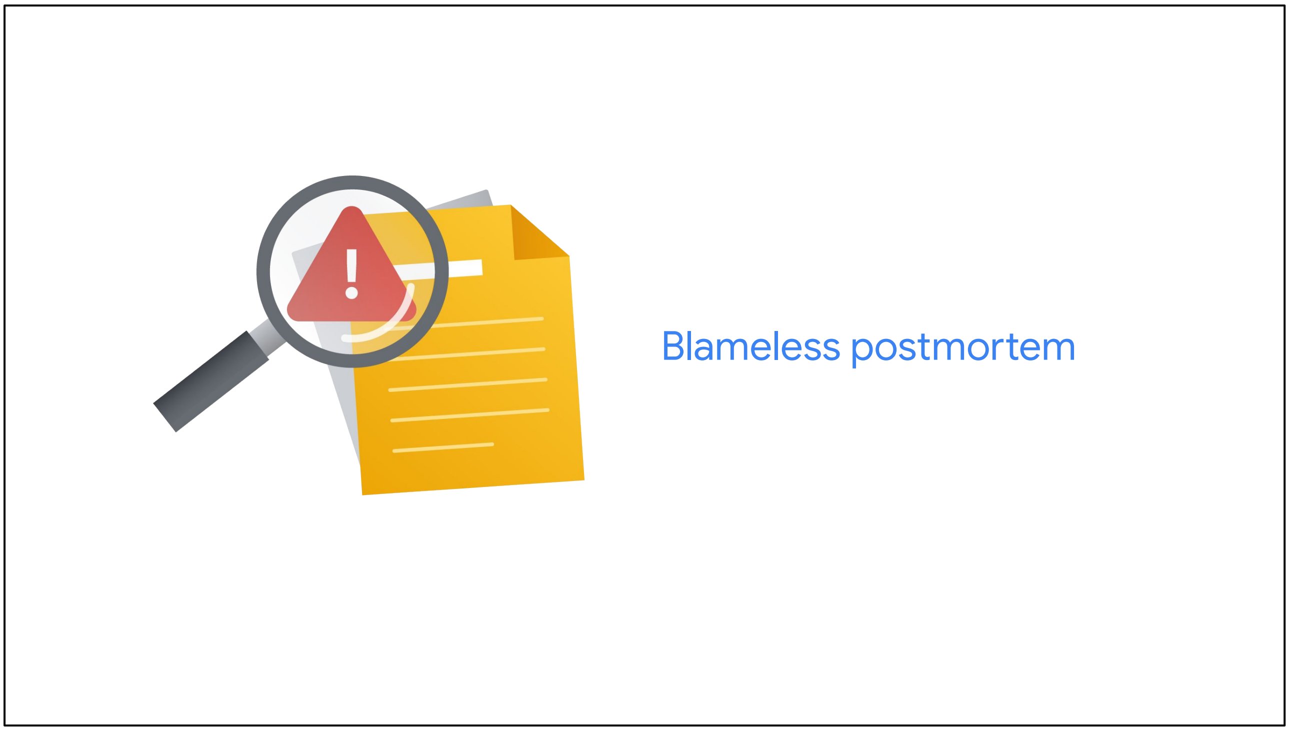


## Experienced SREs

- Are comfortable with failure.
- Eliminate ambiguity with monitoring.
- Establish and document processes.

Experienced Site Reliability Engineers are **comfortable with failure** and know that incidents and outages are going to occur, even if they've taken all the necessary precautions.

Before an outage, SREs seek to **eliminate ambiguity** by building monitoring and observability in the platform and **establishing and documenting processes** for incident response and management, handoffs, and other outage activities. This allows them to confidently focus on the relevant issue during an incident.

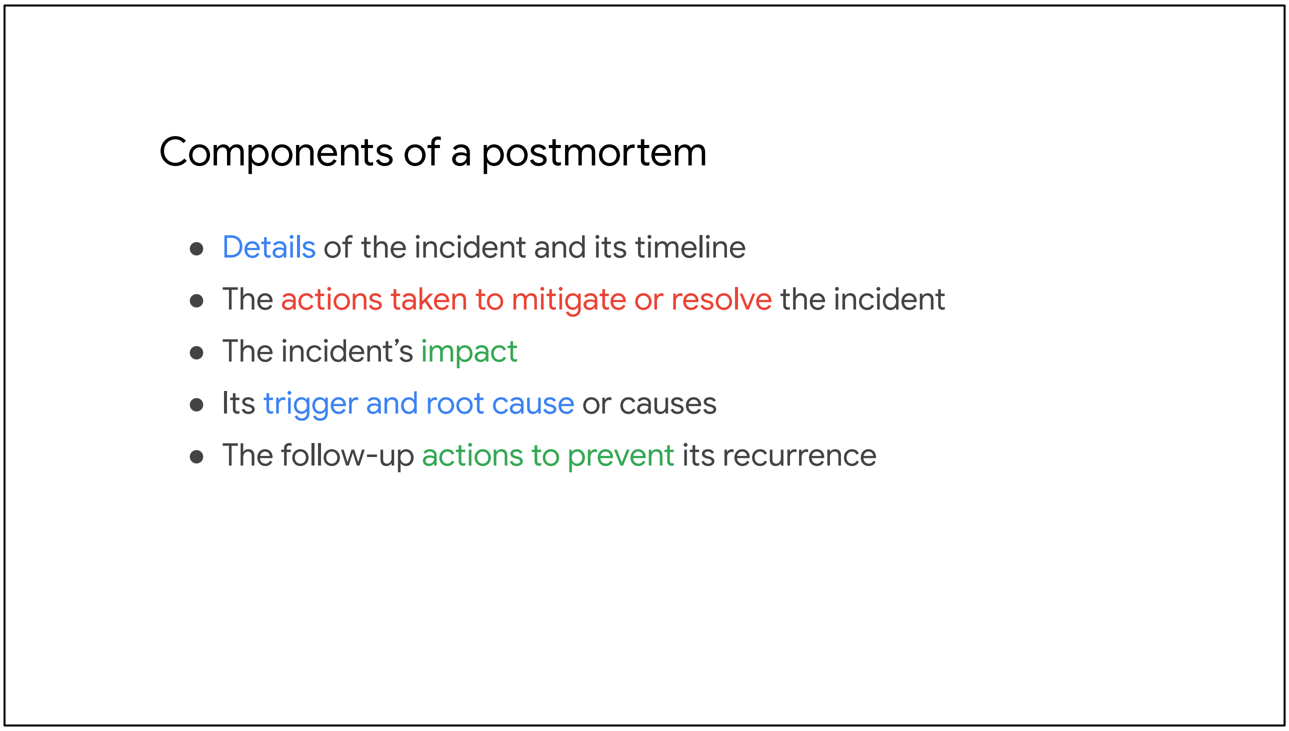


## Blameless postmortem

After an outage, it’s important to understand why an incident occurred and then take steps to make sure it doesn’t happen again in the same way.

SREs do this by documenting and conducting a **blameless postmortem**. Some people also call this a *retrospective*.

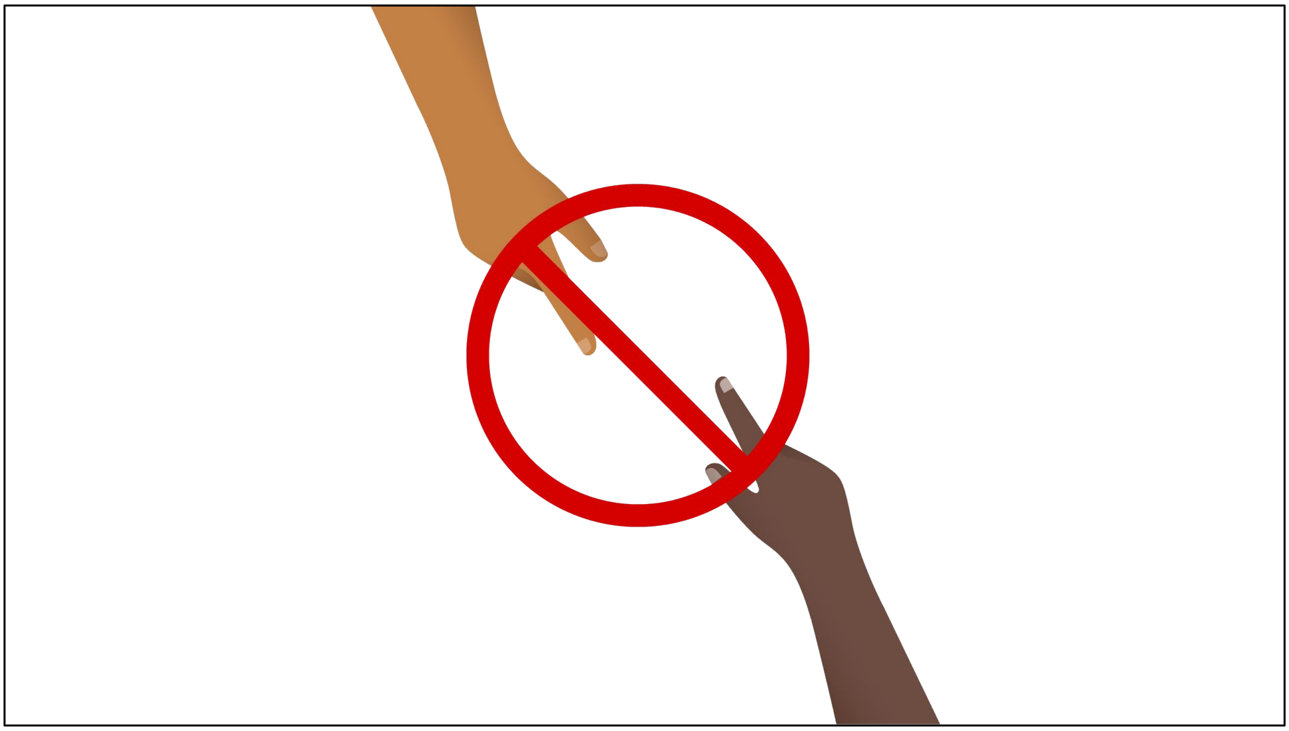
In fast-paced environments where new problems are being addressed constantly, it’s easy to just address one incident and then move on to the next without taking time to actually learn from what happened. To avoid doing this, SREs take a systematic approach to ensure that the team collectively learns from the incident.


## Components of a postmortem

- Details of the incident and its timeline
- The actions taken to mitigate or resolve the incident
- The incident's impact
- Its trigger and root cause or causes
- The follow-up actions to prevent its recurrence

So *what* is the purpose of a postmortem? A postmortem's ultimate deliverable is a written record of an incident that consists of specific parts:

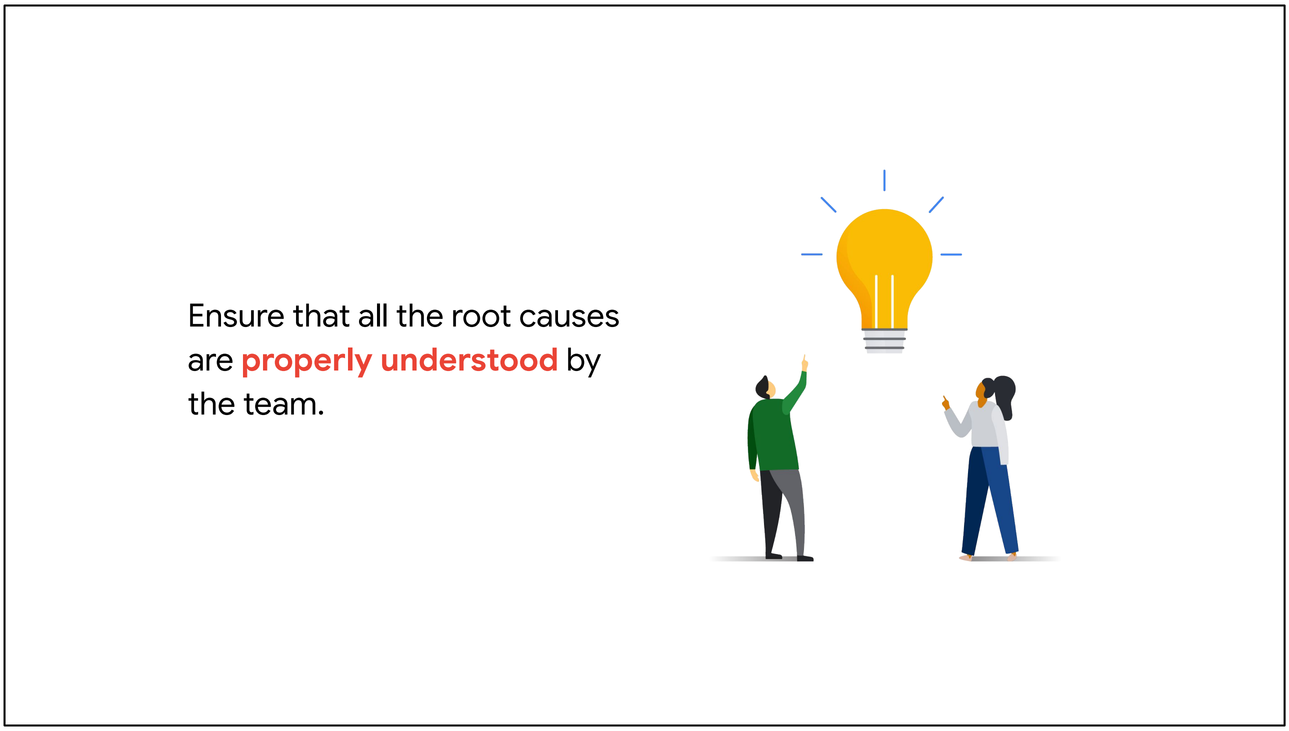
- Details of the incident and its timeline
- The actions taken to mitigate or resolve the incident
- The incident's impact
- Its trigger and root cause or causes
- The follow-up actions to prevent its recurrence



Particularly, a **blameless** postmortem only focuses on the root causes of an incident **without accusing a particular person or team**, or their actions or behavior. Specific people will write and review the postmortem, but **everyone who had a role in the event** will be a part of the postmortem process so you can collect as much information as possible.

Now you might be wondering *why* you should conduct a postmortem, beyond identifying the root cause or causes. If it was just about fixing and preventing the issue, it might seem like it's unnecessary if you've already accomplished that goal.

A postmortem has several specific and important goals:



Ensure that all the root causes are **properly understood** by the team.

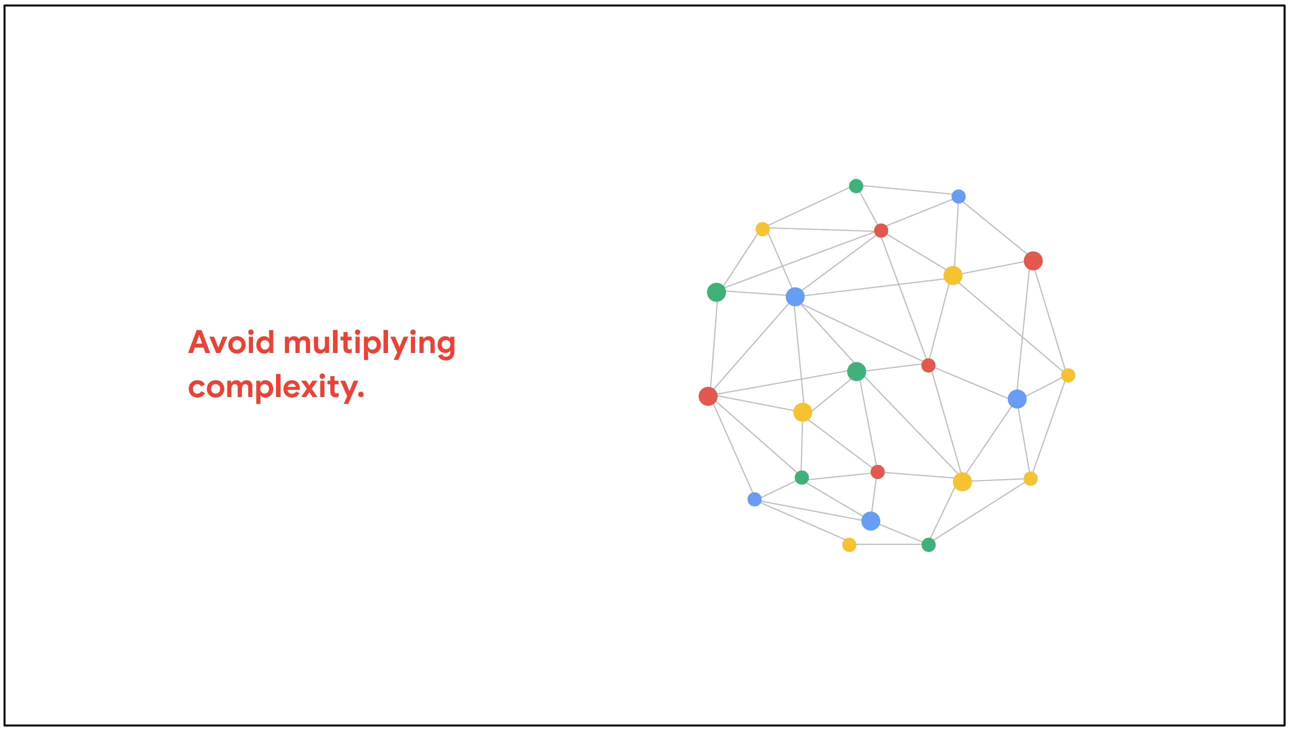
- You want to ensure that all the root causes are **properly understood** by the team.
  - Almost all outages have multiple causes at their root. Many times, each of those causes taken in isolation may not have been enough to cause a failure, but when combined, they lead to an incident. Tactics like “the 5 whys” are used to probe deep into what caused an incident and all of the contributing factors, not just what first appears to be the culprit.

Define or take **effective actions to prevent** the issue from occurring again.

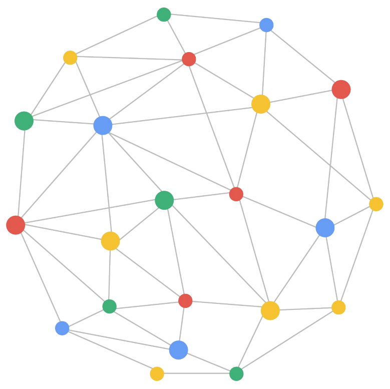
- You want to define or take **effective actions to prevent** the issue from occurring again.
    - At this point you've probably taken some actions to resolve the immediate user impact, but in the long (or even short) term, the outage might happen again. You need to prevent recurrence and prioritize the work to do so.

Reduce the likelihood of stressful outages.

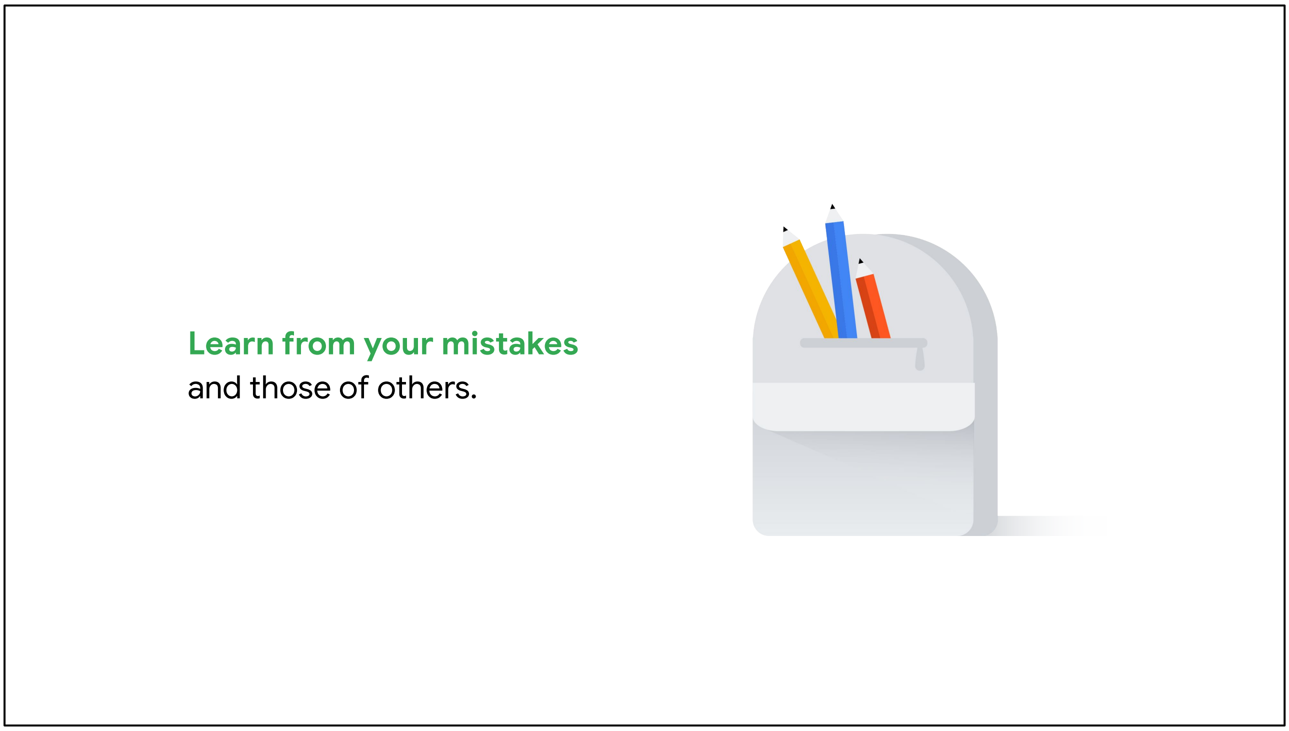
- You want to **reduce the likelihood** of stressful outages.
    - Every outage is a stressor on the team. Google wants its SREs to spend their time improving its systems, not dealing with incidents. Good system hygiene, including outage prevention, is a key quality-of-life factor for your engineers.



## Avoid multiplying complexity.



- You want to **avoid multiplying complexity**.
    - Much of the time, “quick fixes” are involved in solving incidents and preventing their immediate recurrence. Each of those fixes is like a bandaid or patch on the system. If you don’t perform good postmortems and permanently prevent recurrence, over time the fixes will become interdependent and sticky, as each one stacks on the other. This makes the system more complex than necessary and less maintainable and ultimately increases the likelihood of future failure.



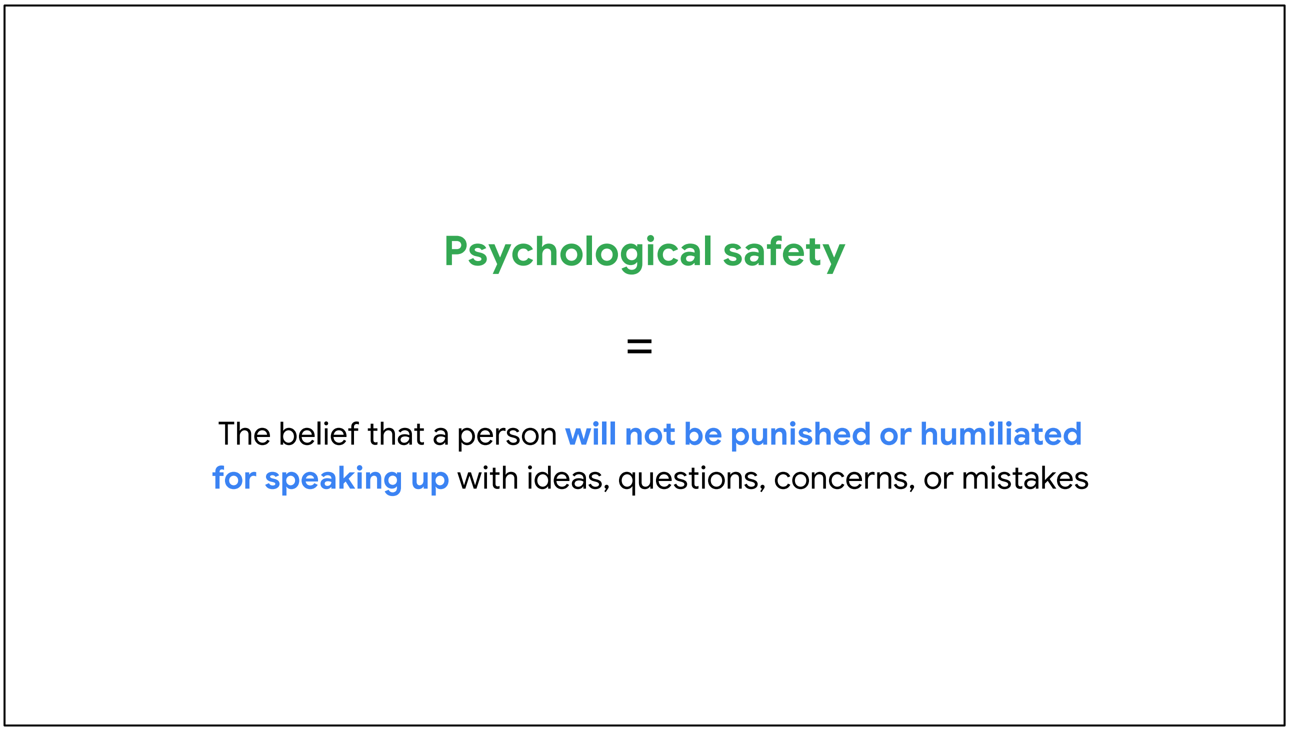
Learn from your mistakes
and those of others.

- You want to **learn from your mistakes** and those of others.
  - Every failure is an opportunity to learn. Good SREs are “constructive pessimists,” and a lot of their instincts come from experiencing past failure.

In addition to creating a documented record for your team to learn from, the practice of writing a postmortem provides additional value to your organization. Focusing on blamelessness helps to increase the effectiveness of your teams. They become 100% focused on preventing a problem from recurring, instead of worrying about being blamed if something goes wrong. It also helps to promote a culture of psychological safety, which we will discuss more in the next video.

SREs believe that failure is normal and that progress is about learning from mistakes and ensuring that they don’t make the same one in the same way again.

As you learned in the previous video, a key SRE practice is conducting a postmortem after an incident or outage occurs. An SRE postmortem is **blameless**, which means the incident is looked at objectively without designating a person or team as the root cause.



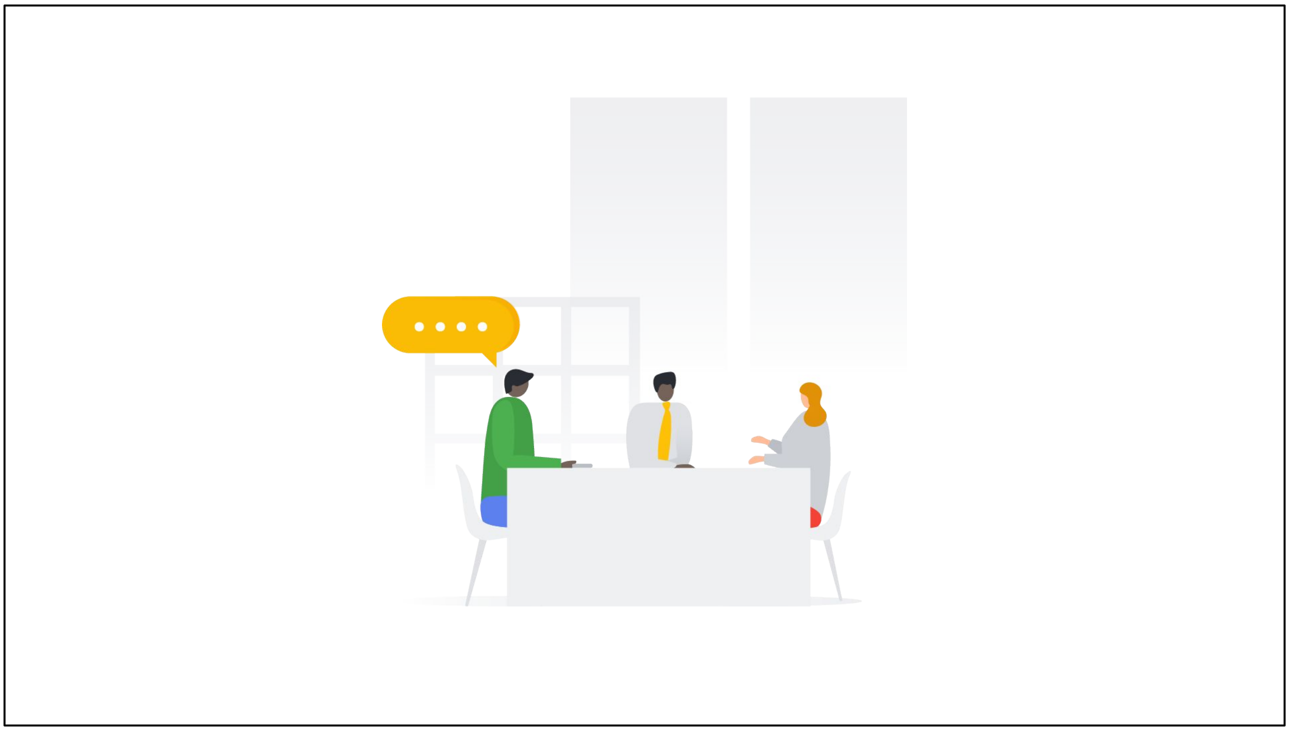
Psychological safety

=

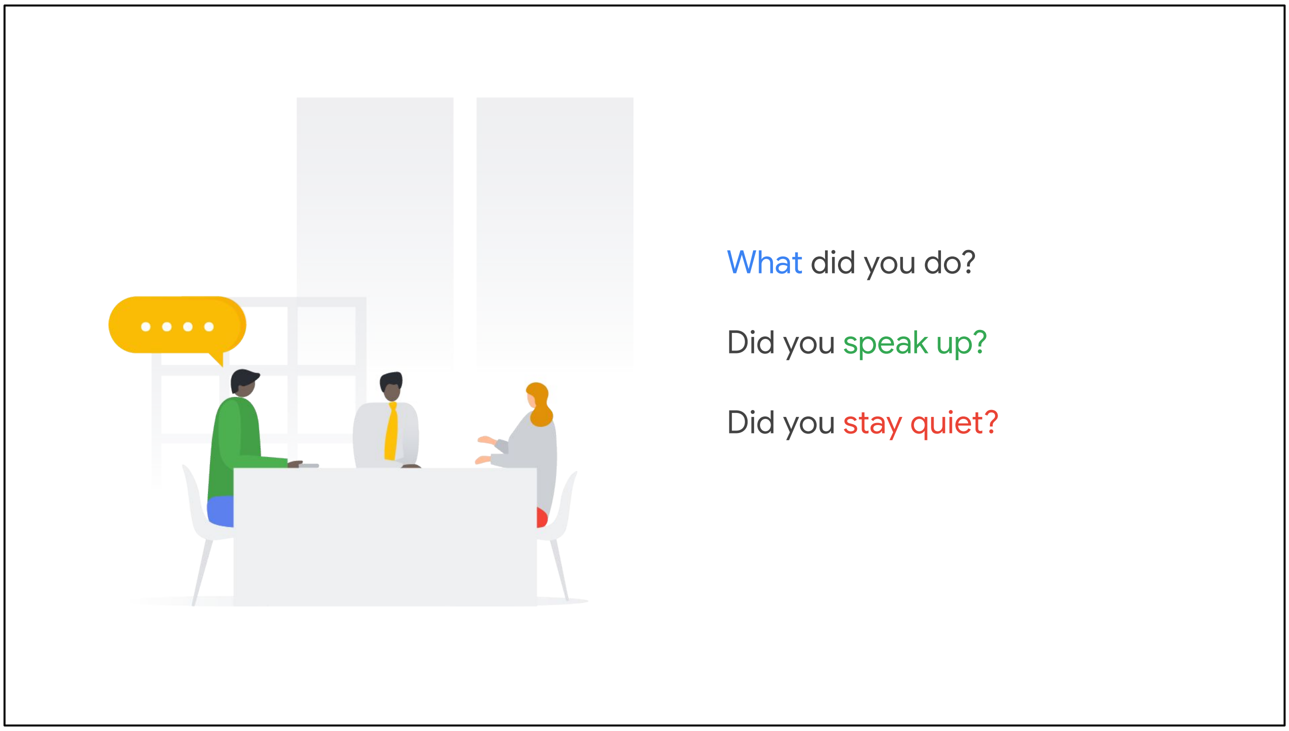
The belief that a person **will not be punished or humiliated for speaking up** with ideas, questions, concerns, or mistakes

Blamelessness can have an immense positive effect on the culture of your organization. Specifically, it creates a culture of **psychological safety** for your teams.

So what is meant by “psychological safety?” Psychological safety is the belief that a person **will not be punished or humiliated for speaking up** with ideas, questions, concerns, or mistakes.



Think back to a time when you had a concern about a task you were asked to perform as a manager or an individual contributor. Maybe there was an approach your manager asked your team to take, and while everyone else agreed, you had some reservations.

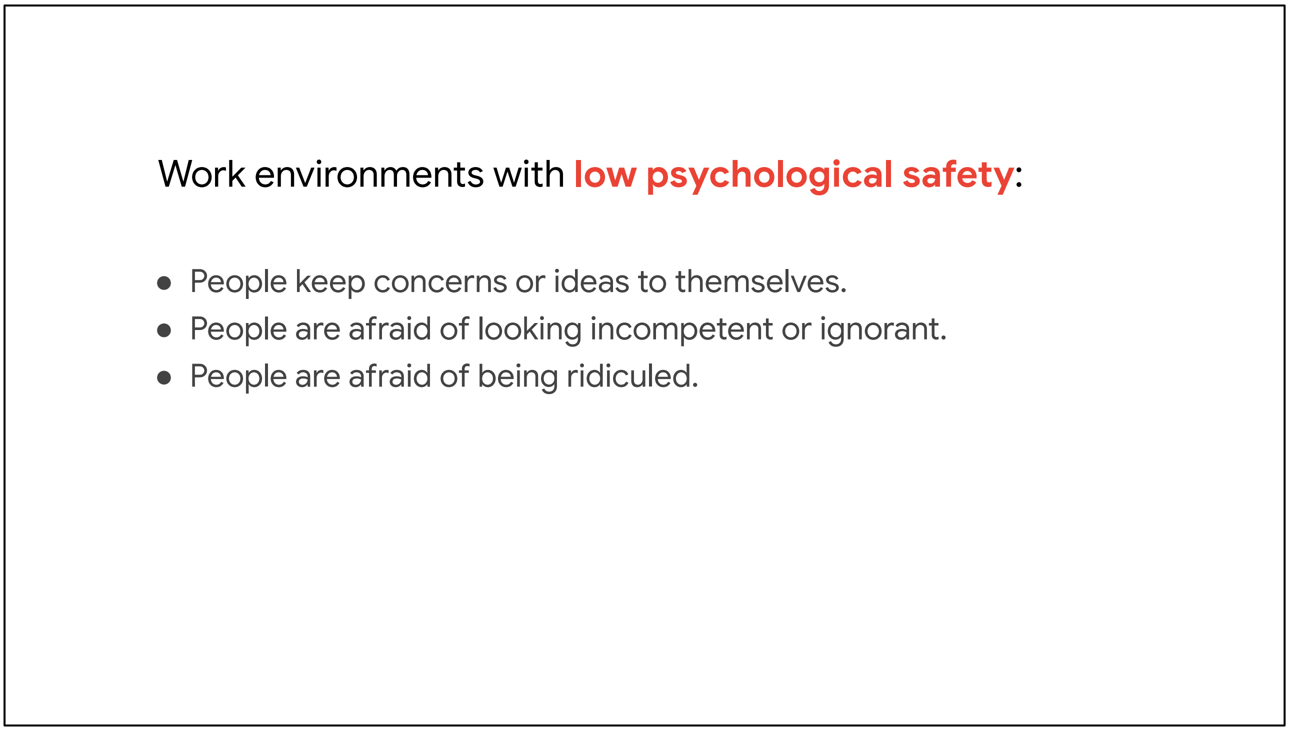


What did you do?

Did you speak up?

Did you stay quiet?

What did you do in that situation? Did you speak up and voice your concerns? Or did you stay quiet? And if you stayed quiet, do you remember why you did?



## Work environments with **low psychological safety**:

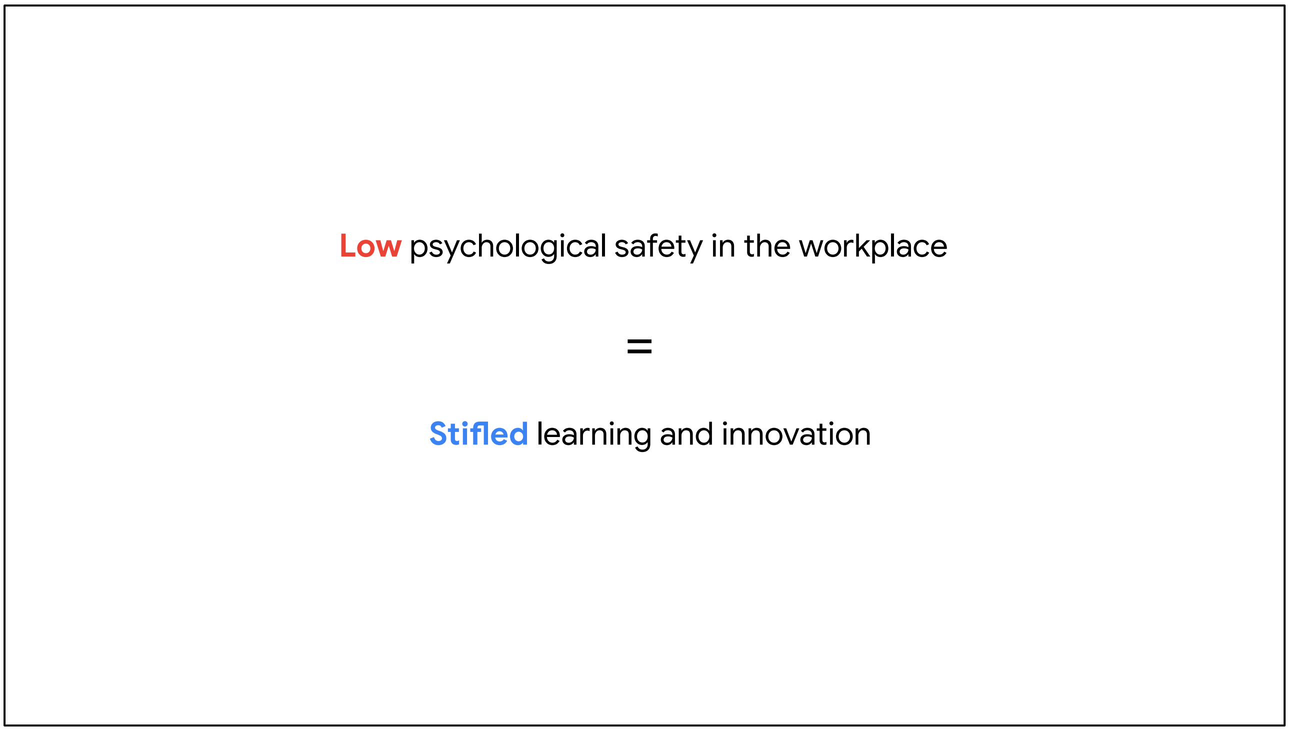
- People keep concerns or ideas to themselves.
- People are afraid of looking incompetent or ignorant.
- People are afraid of being ridiculed.

In work environments with low psychological safety, team members are more likely to keep their concerns or ideas to themselves because they feel that they will look incompetent or ignorant or even be ridiculed for having a differing opinion. This kind of fear can have a lasting impact on your teams.

So what do you think is a **consequence** of low psychological safety among teams?

When you don’t say what you want to say, you’re actually robbing yourself and your team members of small moments of learning. Maybe the concern or question you have has a simple clarification or answer. If you express it, you and others can easily learn what that answer is. But maybe what you have to say or ask isn’t so straightforward. Perhaps it will cause others to think differently, spark a new conversation, or lead to a new idea.

When people are busy managing impressions, they don’t contribute to creating a better organization.

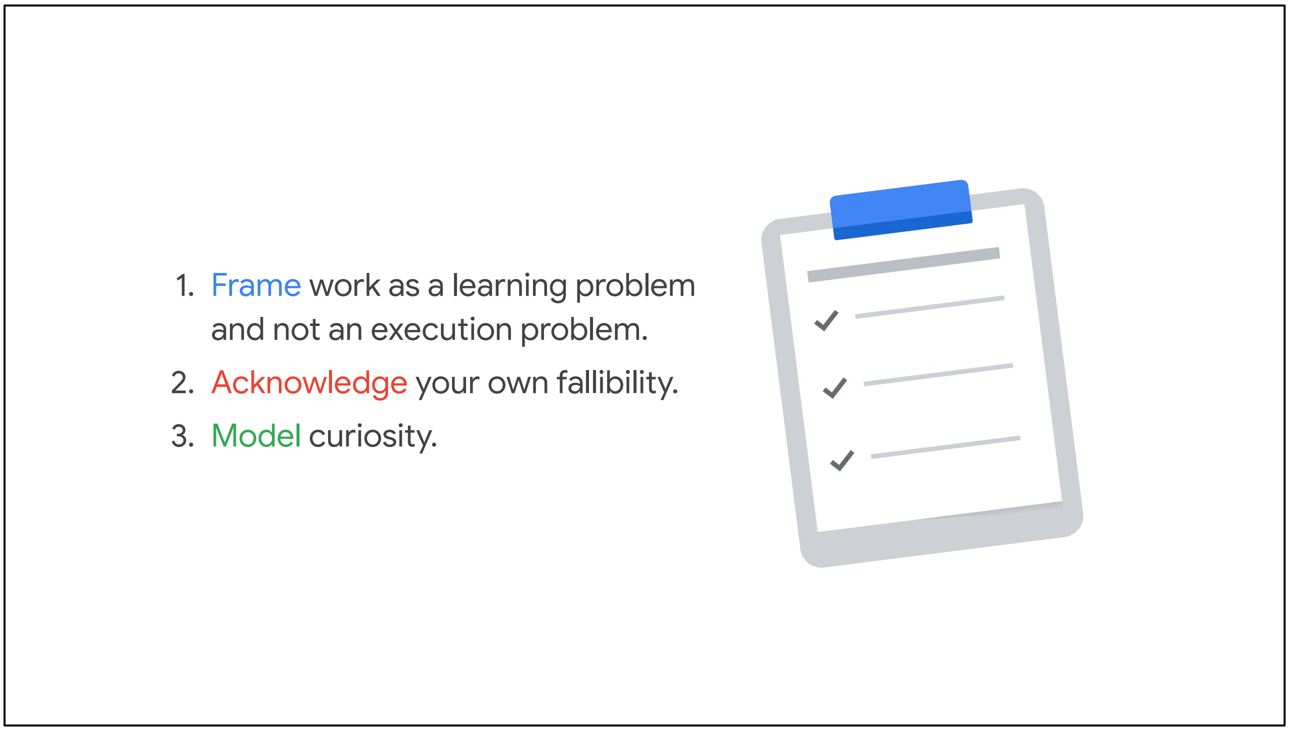

Low psychological safety in the workplace

=

Stifled learning and innovation

In short, low psychological safety in the workplace can stifle **learning** and **innovation**.

Knowing that psychological safety is important in your organization is the first step. So next, how do you **build** it?

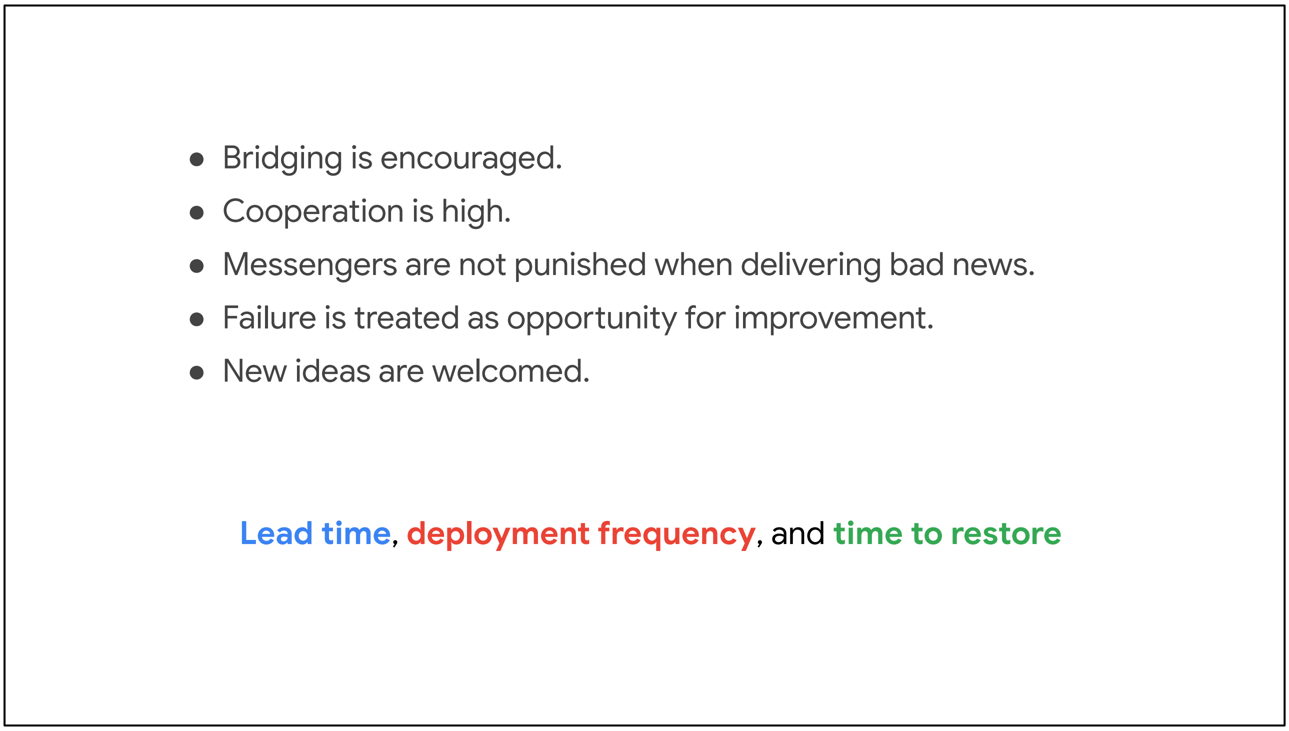


1. Frame work as a learning problem and not an execution problem.
2. Acknowledge your own fallibility.
3. Model curiosity.

As a leader, you can start with three simple steps:

1. Frame work as a learning problem and not an execution problem. Let your people know that you need everyone's voice and ideas.
2. Acknowledge your own fallibility. Let them know that you may miss something you need from them, and encourage them to ask questions.
3. Model curiosity. Make sure that you, yourself, ask a lot of questions!

So how do psychologically safe environments affect software delivery? Well, research has shown they have high impacts on it.



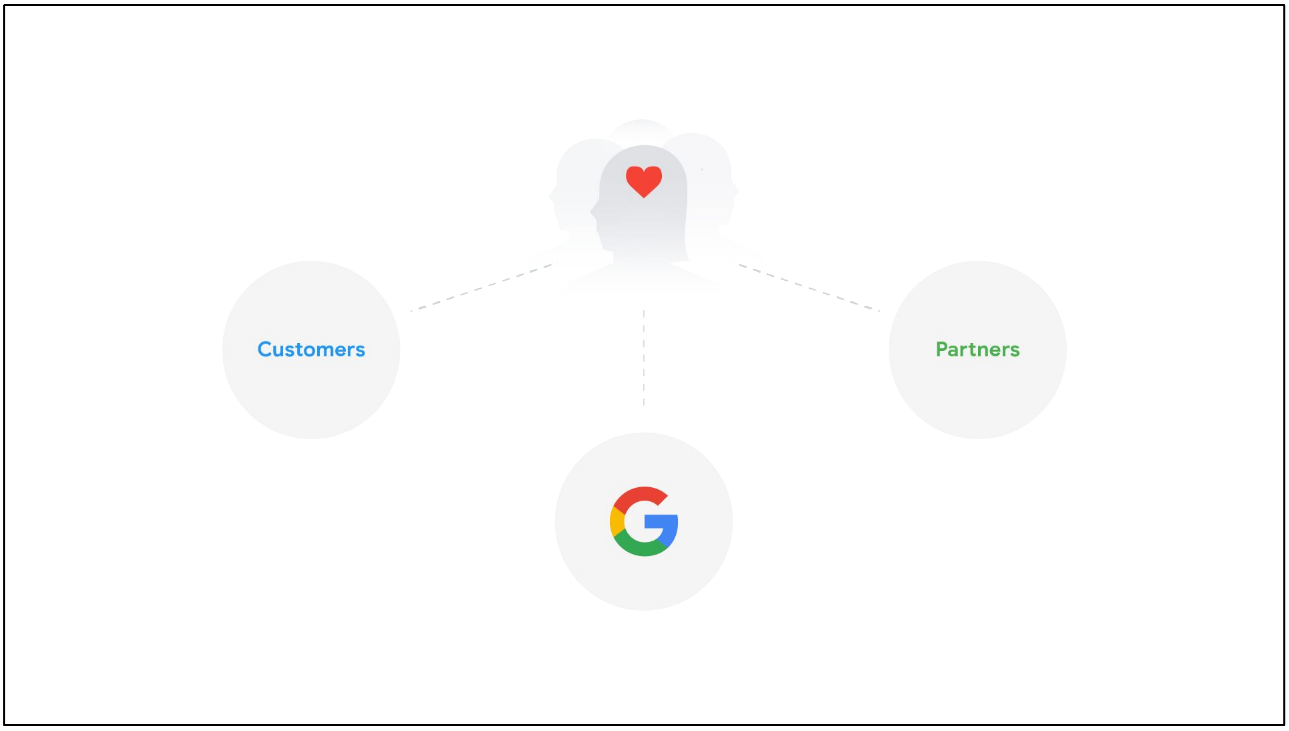
- Bridging is encouraged.
- Cooperation is high.
- Messengers are not punished when delivering bad news.
- Failure is treated as opportunity for improvement.
- New ideas are welcomed.

Lead time, deployment frequency, and time to restore

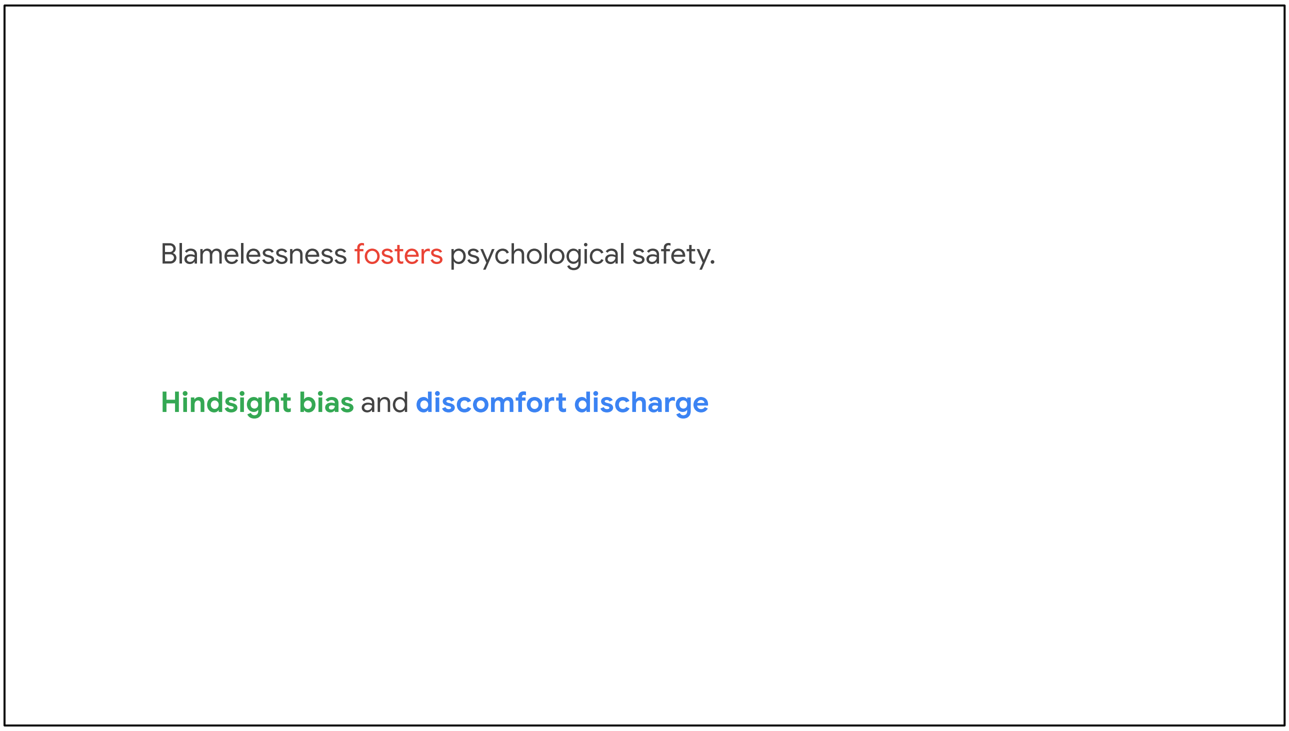
In these work environments:

- Bridging is encouraged.
- There is high cooperation.
- Messengers are not punished when delivering bad news.
- Failure is treated as an opportunity for improvement.
- New ideas are welcomed.

These attributes lead to improvements in **lead time, deployment frequency,** and **time to restore.**



Psychological safety plays a key role in both boosting SRE team dynamics and also directly influencing operational excellence and software delivery. Google has seen it in working with our customers and partners and also within Google itself. Shifting team focus from blaming individuals to analyzing processes is a transformation in how engineering teams operate and has long-term benefits in fostering psychological safety and establishing trust.

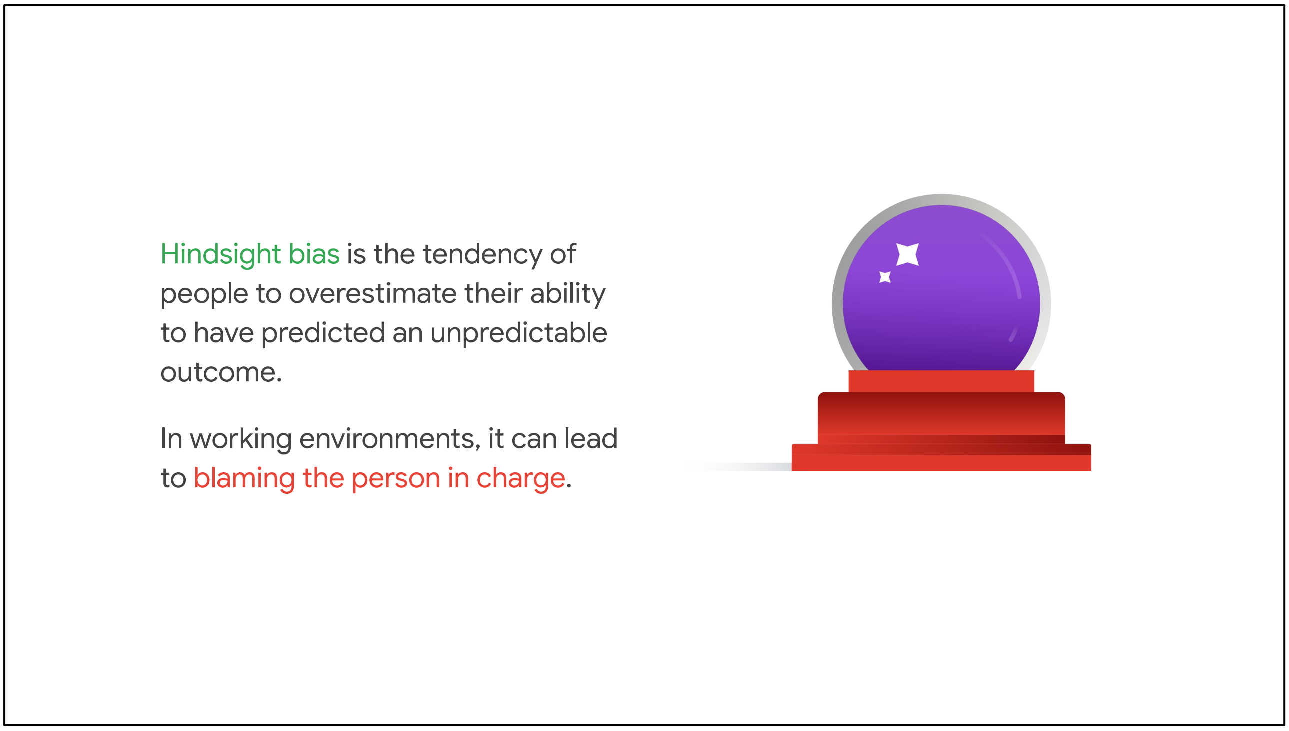


Blamelessness fosters psychological safety.

**Hindsight bias** and **discomfort discharge**

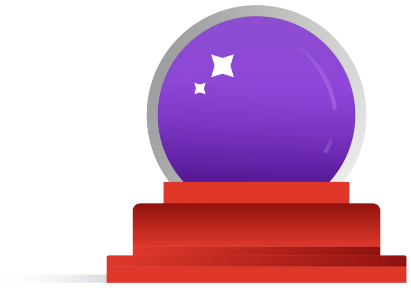
Now, let’s talk a bit about blamelessness. In order to truly employ the practice of postmortems, you need to foster psychological safety in your organization. **Blamelessness is a behavior that fosters psychological safety.**

But *why* do people blame? Research has shown there are two major factors that fuel people’s tendency to blame others: **Hindsight bias** and **discomfort discharge.**



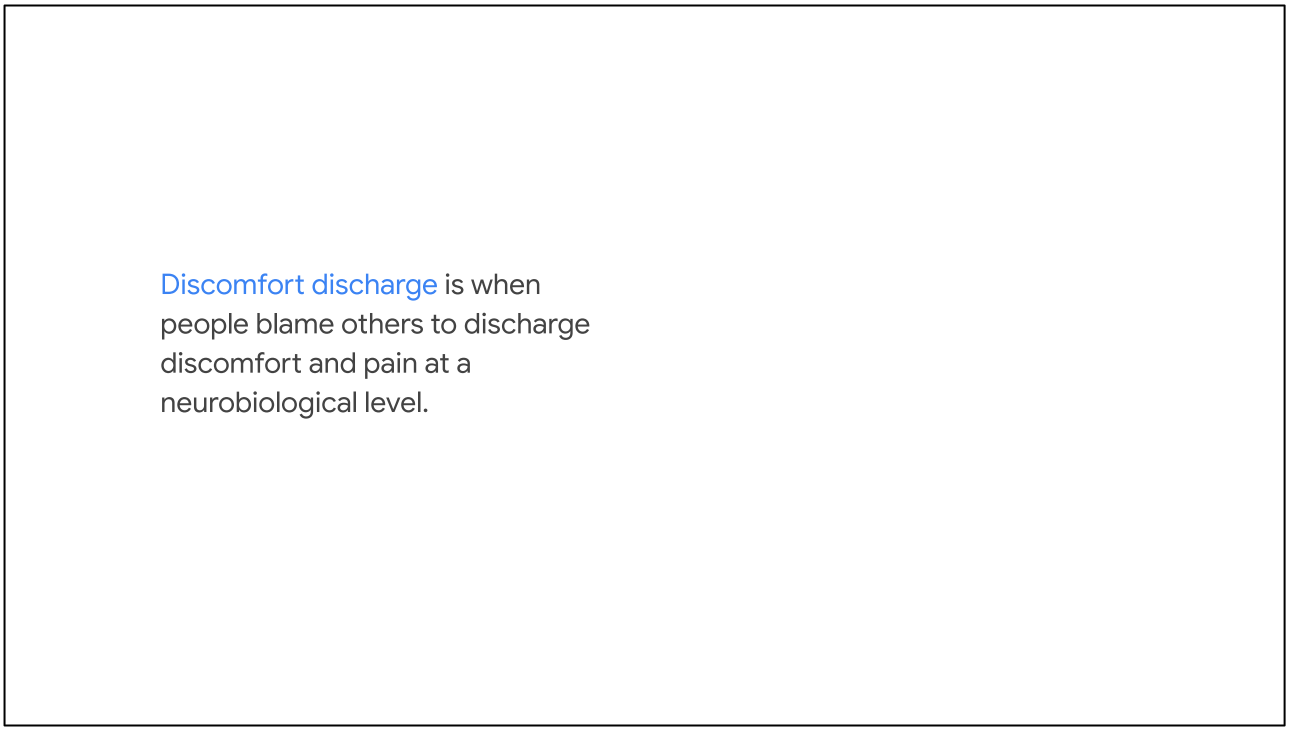
Hindsight bias is the tendency of people to overestimate their ability to have predicted an unpredictable outcome.

In working environments, it can lead to blaming the person in charge.



Hindsight bias is the tendency of people to overestimate their ability to have predicted an outcome, even though it could not have possibly been predicted. People often fail to realize that it's only obvious now that it has already happened. A simple example of hindsight bias is insisting that you knew that the losing team of a sporting event was going to lose all along, just because you originally said you predicted it.

In working environments, however, hindsight bias can often lead to blaming the person in charge, saying they should have seen the obvious and planned for it.



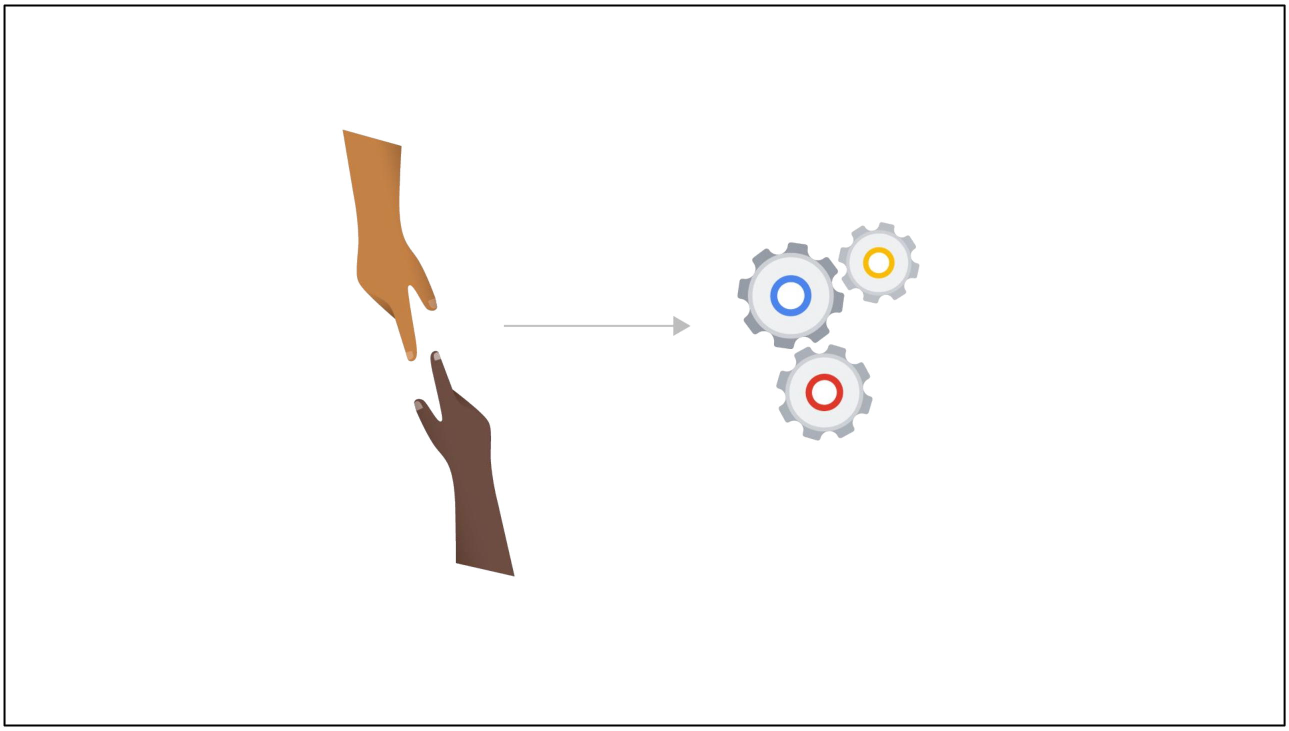
Discomfort discharge is when people blame others to discharge discomfort and pain at a neurobiological level.

The second factor is discomfort discharge, which says that blame exists to discharge discomfort and pain at a neurobiological level. Sociologist Brene Brown claims that we are pretty much wired for blame, because it's a natural way to release discomfort.

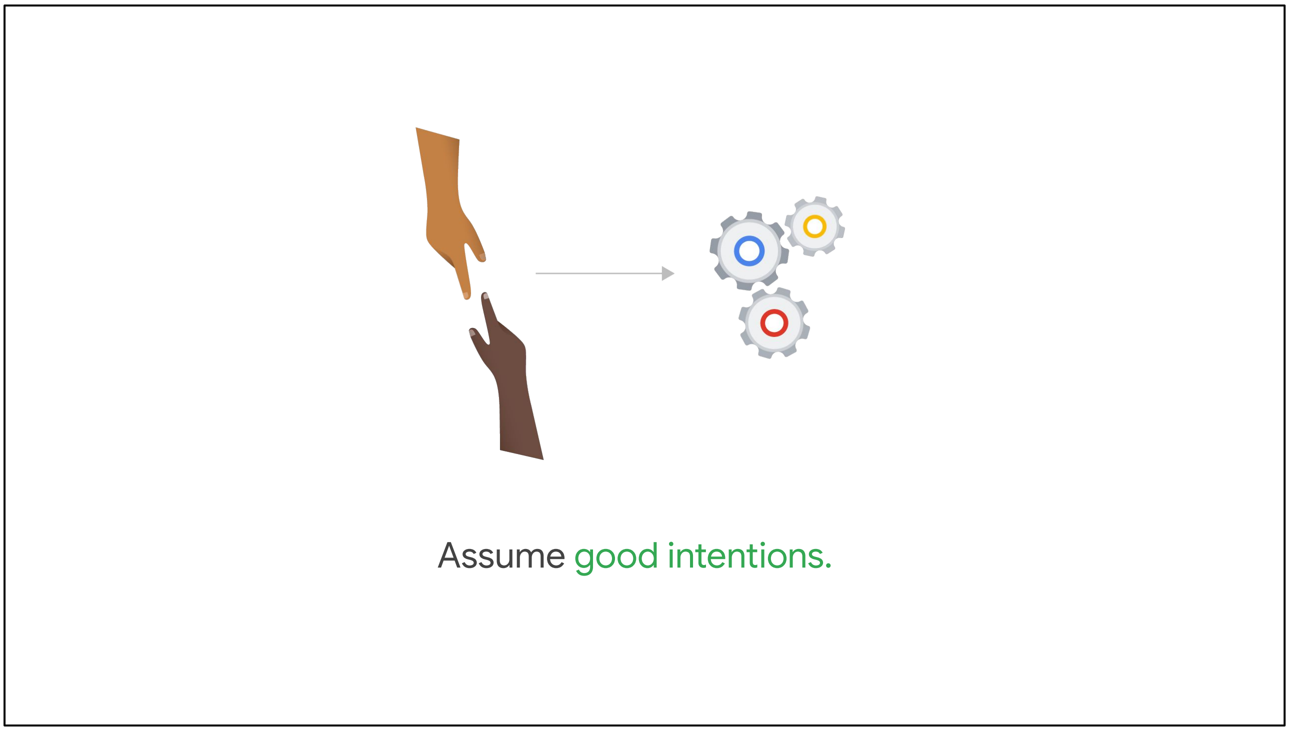
But blaming people hinders the ability to learn from mistakes. People tend to hide information or don't declare incidents because they're scared of punishment. Similarly, people are afraid to ask the questions that may lead to identifying the root causes of an incident if they feel their question may lead to punishment or ridicule for themself or their peers.

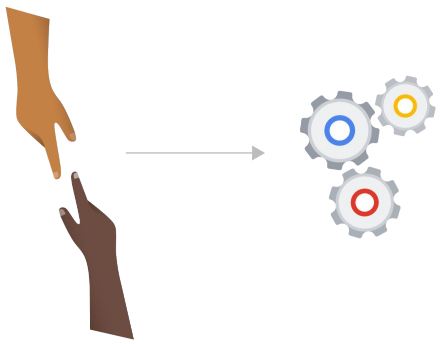
Mistakes are valuable opportunities to learn and improve only if the correct procedural and systematic causes of the mistake are properly identified. In short, blaming people creates environments that are **not** psychologically safe.

So how can you **focus on blamelessness** in your organization and with SRE practices such as postmortems?



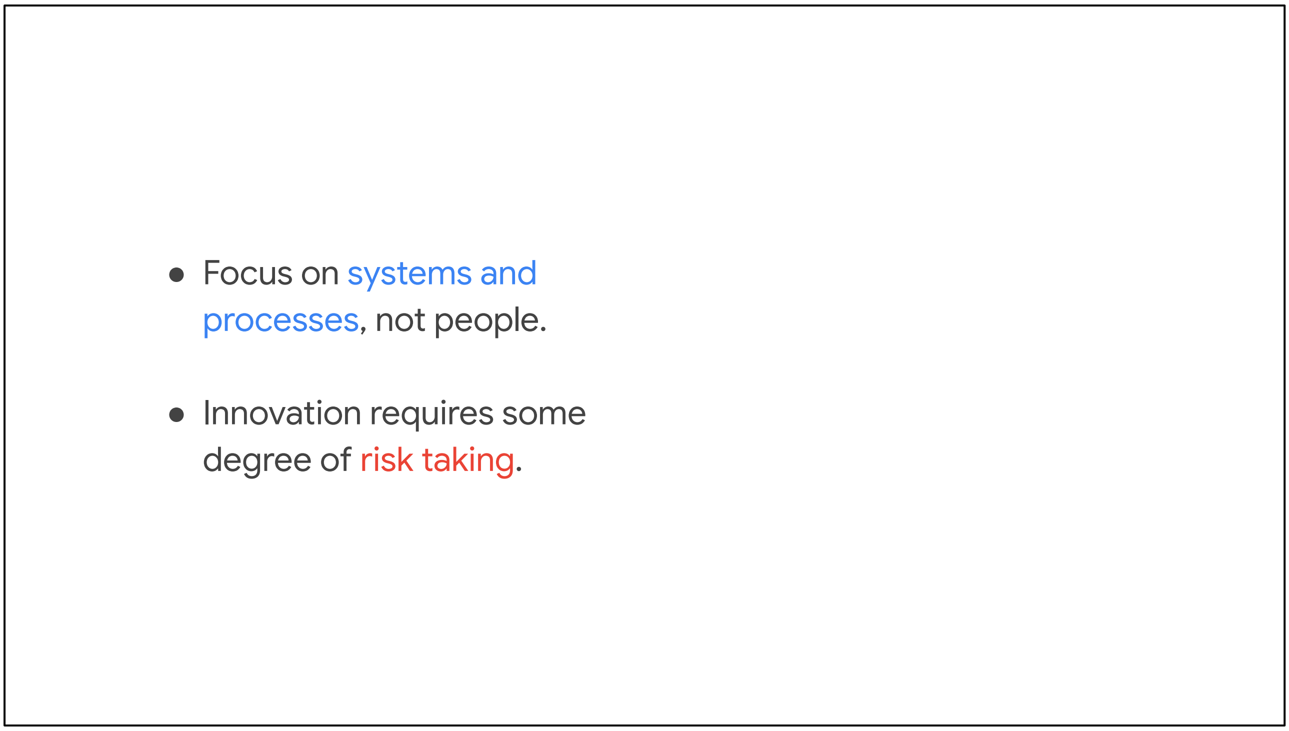
Blamelessness is the notion of **switching responsibility from people to systems and processes**. In “finger pointing” organizations—where the first question a manager asks after an incident is “Who did this?”—employees become much more risk averse and fearful.





Assume good intentions.

It’s best practice to assume that individuals have positive intentions and make decisions based on the best information available. Investigating the source of misleading information is much more beneficial to the organization than assigning blame.

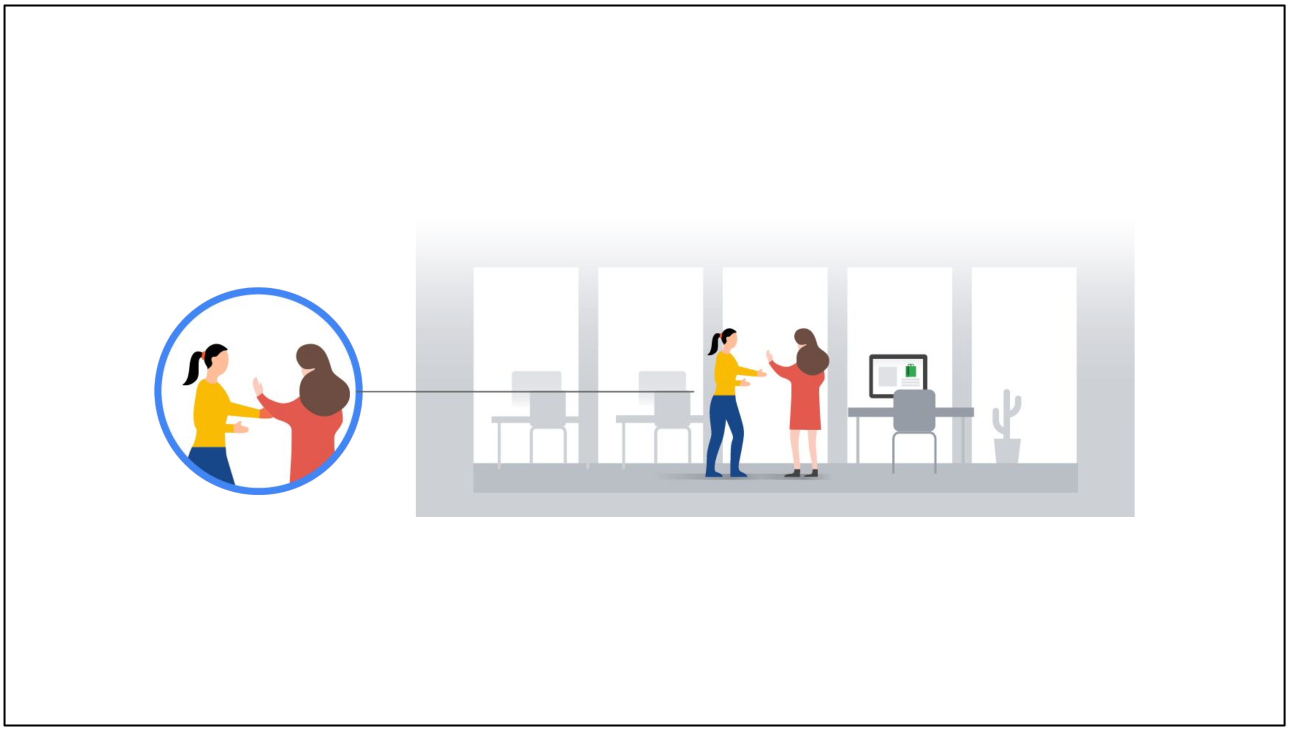


- Focus on systems and processes, not people.
- Innovation requires some degree of risk taking.

In short, don’t focus on people. Focus on **systems and processes** to better support people making the right choices when designing and maintaining complex systems.

It’s also important to mention that aside from the ability to learn from previous mistakes, blaming people instead of systems or processes has a negative impact on the organization’s ability to innovate and improve, as **innovation inherently requires some degree of risk taking**. No new product, service, or process has 100% probability of success. So no one wants to propose improvements if they’re going to be blamed for it if it fails.

Google’s Professional Services team has taken this topic to several of its customers who, like you, are interested in developing SRE culture within their business. They were able to help expose certain aspects of team culture that needed attention and offer SRE culture-focused training to create a plan for improvement.



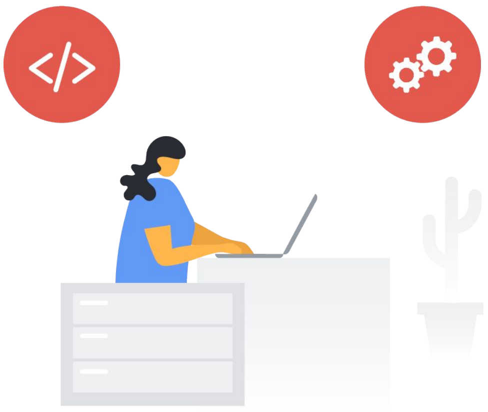
A top online retailer in the Netherlands learned about psychological safety for the first time during one of Google's SRE workshops. They were very enthusiastic about the concept but found it very difficult to implement. Due to their then-current blaming culture, they considered “identifying the person to blame” as a solution to their incidents.

Google introduced them to SRE philosophy, and through intense internal training coupled with performance management changes, they are now on the right path to create a psychologically safe environment.

You can see how blamelessness and psychological safety are important organizational aspects to foster and develop in your business, especially if you want to implement SRE. The success of shifting your IT teams to SRE is highly dependent on your commitment to these as cultural norms.

As you learned in the last module, DevOps philosophy emerged in part due to the siloed nature of software development and operations. Developers generally aren’t very familiar with the systems their code runs on, and operators have to deal with code that may be unstable. And that code just keeps coming over to them at a high velocity from agile developers.

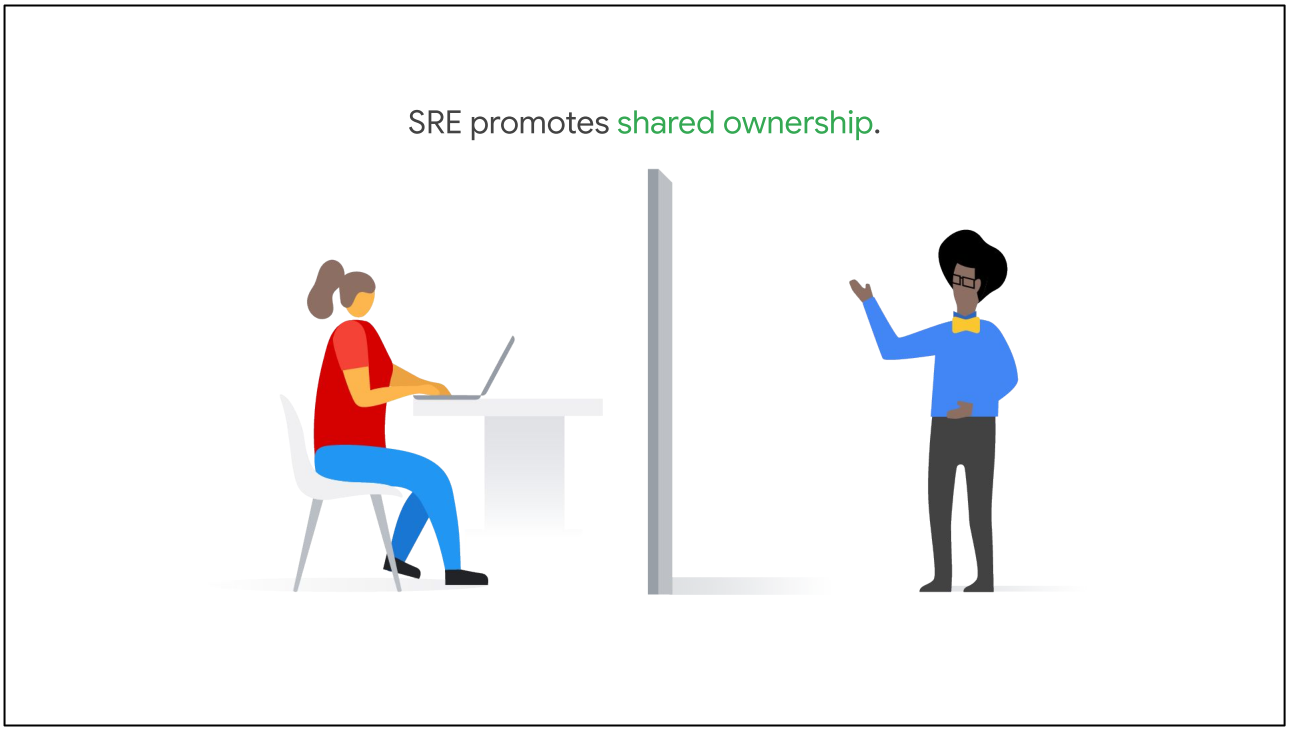
To be an effective team that isn’t stuck in manual break-fix mentality, the walls between the business, development, and operations teams need to be knocked down.



40% to 90% of total software engineering costs are incurred **after** launch.

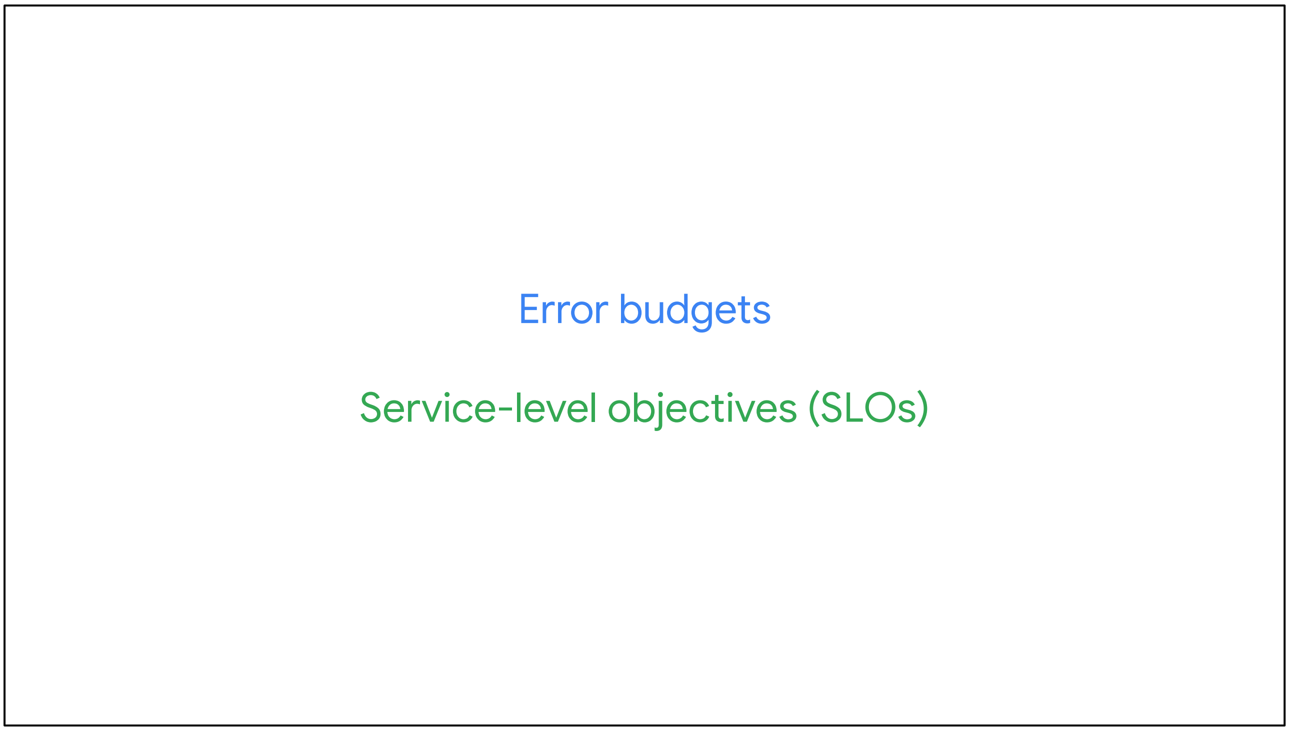
Software engineering as a discipline focuses on designing and building rather than operating and maintaining, despite estimates that 40% to 90% of the total costs are incurred after launch.

If the majority of the total cost of ownership of software is maintaining it after it’s in production, and developers aren’t working on this, then who is?



SRE promotes shared ownership.

SRE employs practices that effectively break down these silos and promote shared ownership between development and operations teams. More importantly, these fundamentals help teams maintain reliability of their services.



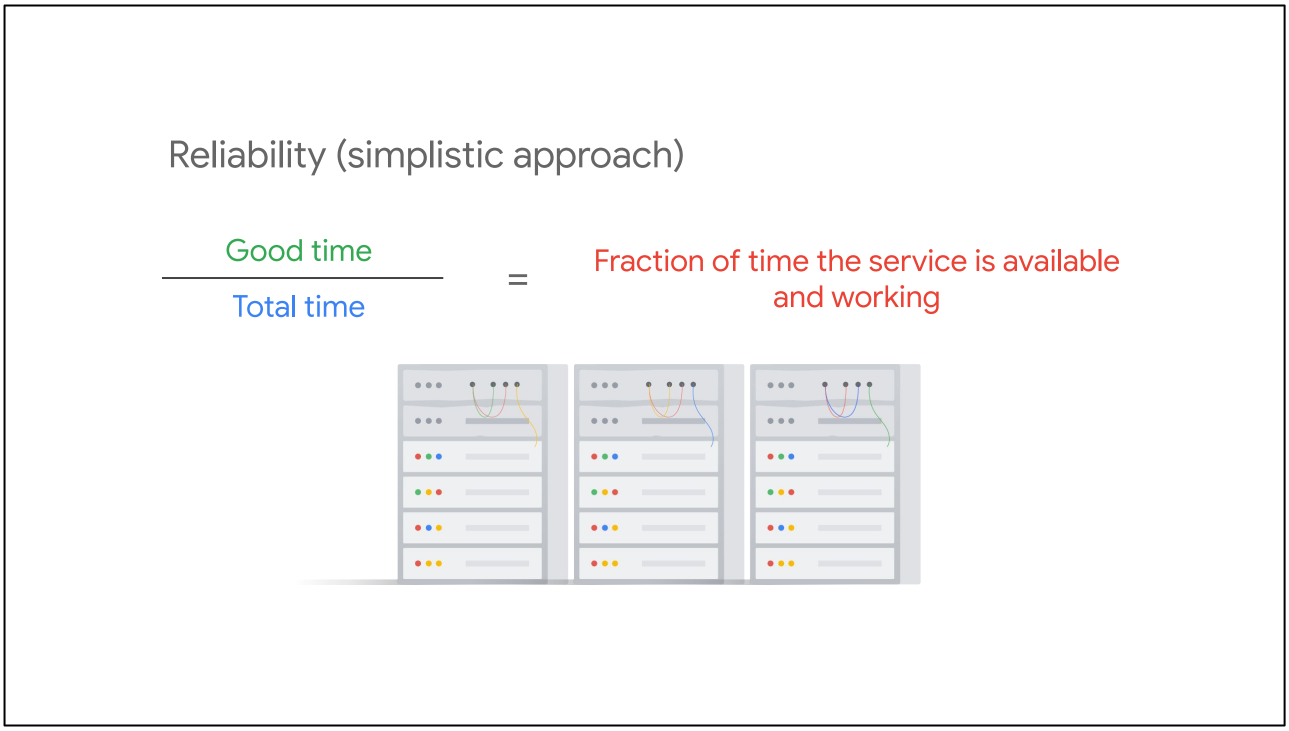
Error budgets

Service-level objectives (SLOs)

There are several practices that support this, but in this course you’ll learn about two specific ones: **error budgets** and s**ervice-level objectives,** or **SLOs.**

SRE gets everyone to agree on *how* to measure service reliability, and *what* to do if you fall short, from individual contributors all the way up to VPs, thus ensuring that responsibility for reliability is shared.

In order to agree on a target for reliability, you'll first have to define what **reliable** means, and how you're determining it.

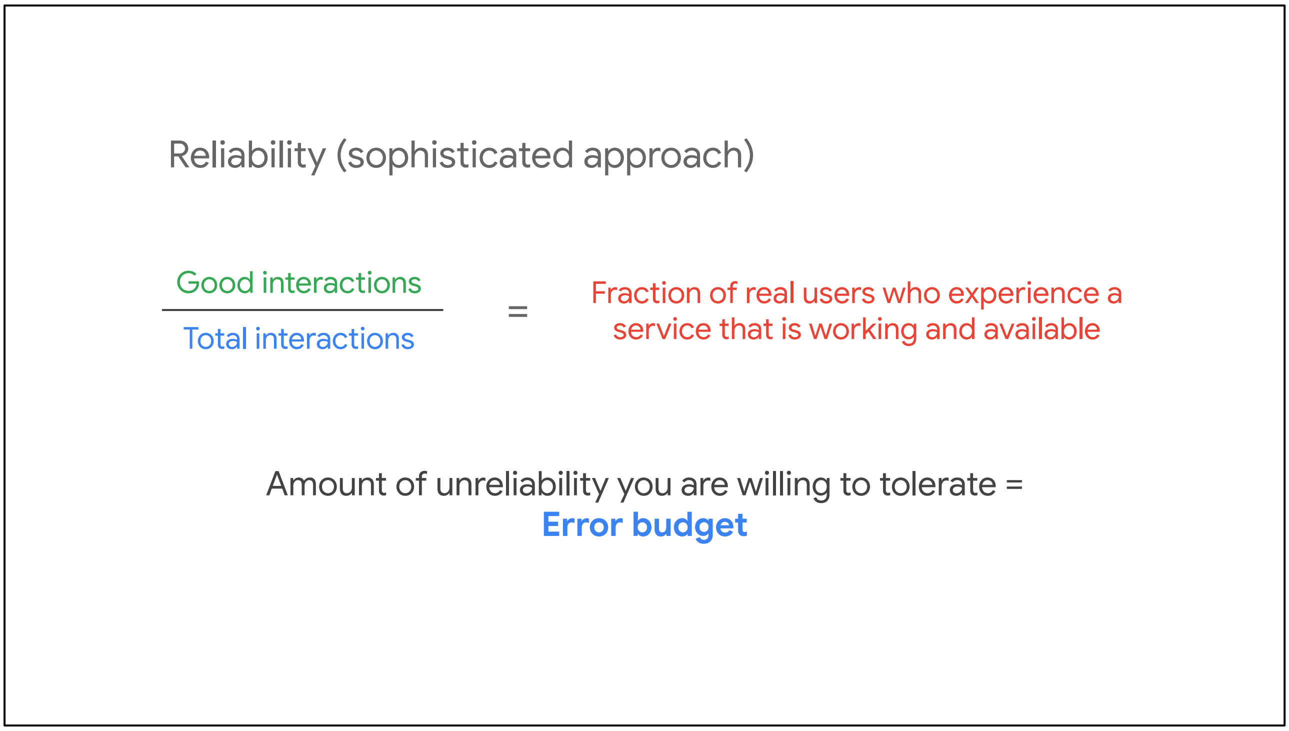


## Reliability (simplistic approach)

$$\frac{\text{Good time}}{\text{Total time}} = \text{Fraction of time the service is available and working}$$

A simplistic way to think about reliability is that it equals your service's "good" time divided by its total time, which gives a numerical fraction of time that the service is available and working.

While this is relatively easy to measure and understand, it doesn't work very well for distributed, complex systems. For example, how do you measure "good" time of a server that currently doesn't receive requests? Or what if one of three servers is down. Does that mean your service is down, or is it up?



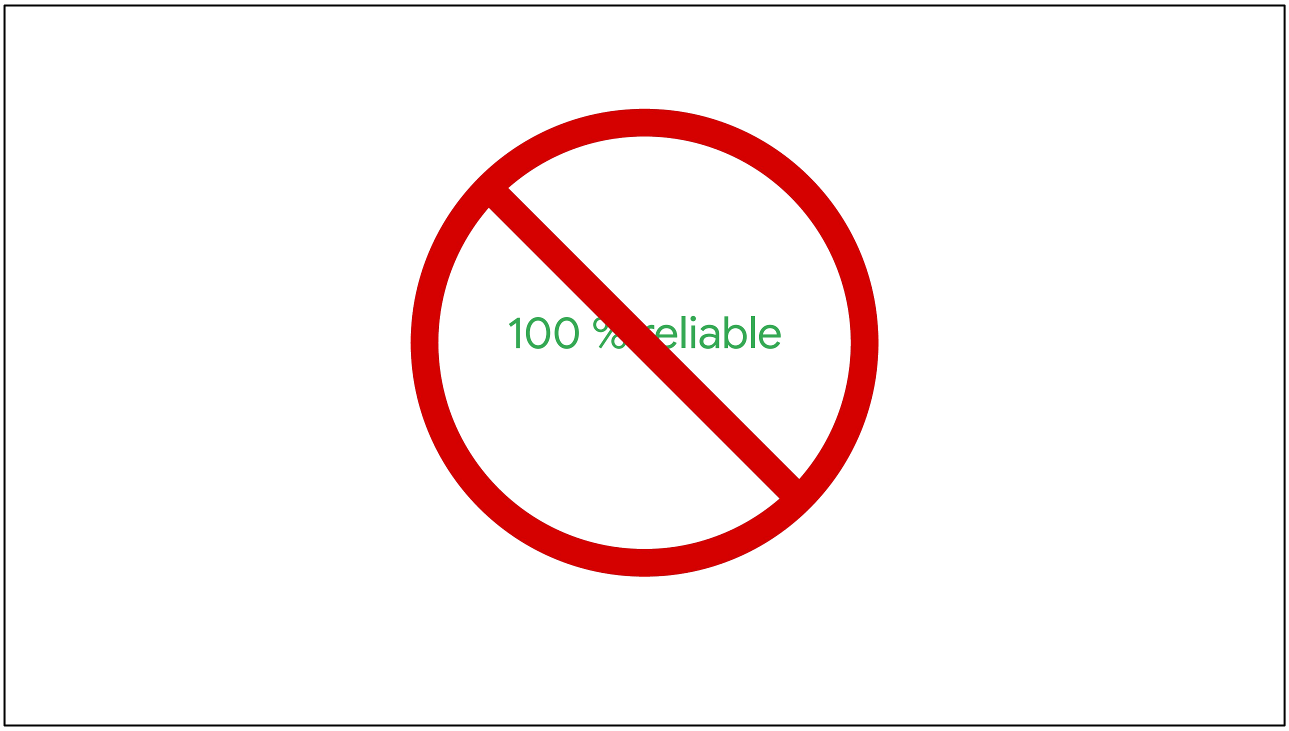
## Reliability (sophisticated approach)

$$\frac{\text{Good interactions}}{\text{Total interactions}} = \text{Fraction of real users who experience a service that is working and available}$$

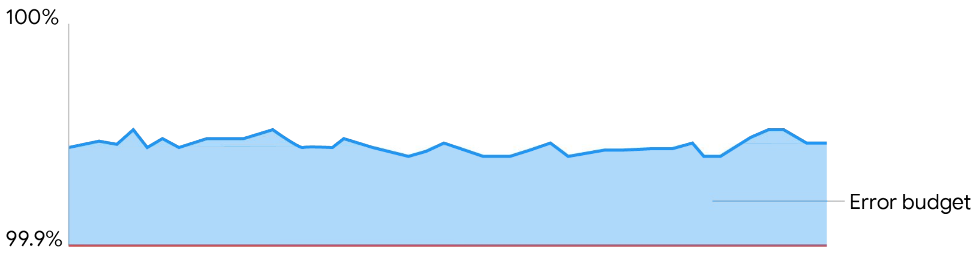
Amount of unreliability you are willing to tolerate =
**Error budget**

The more sophisticated approach is to define reliability as the number of "good" interactions divided by the number of total interactions. This leaves you with a numerical fraction of real users who experience a service that is available and working.

In essence, service reliability, is about determining the amount of reliability you are trying to reach and the amount of downtime you are willing to tolerate. That amount of unreliability you are willing to tolerate is your **error budget**.

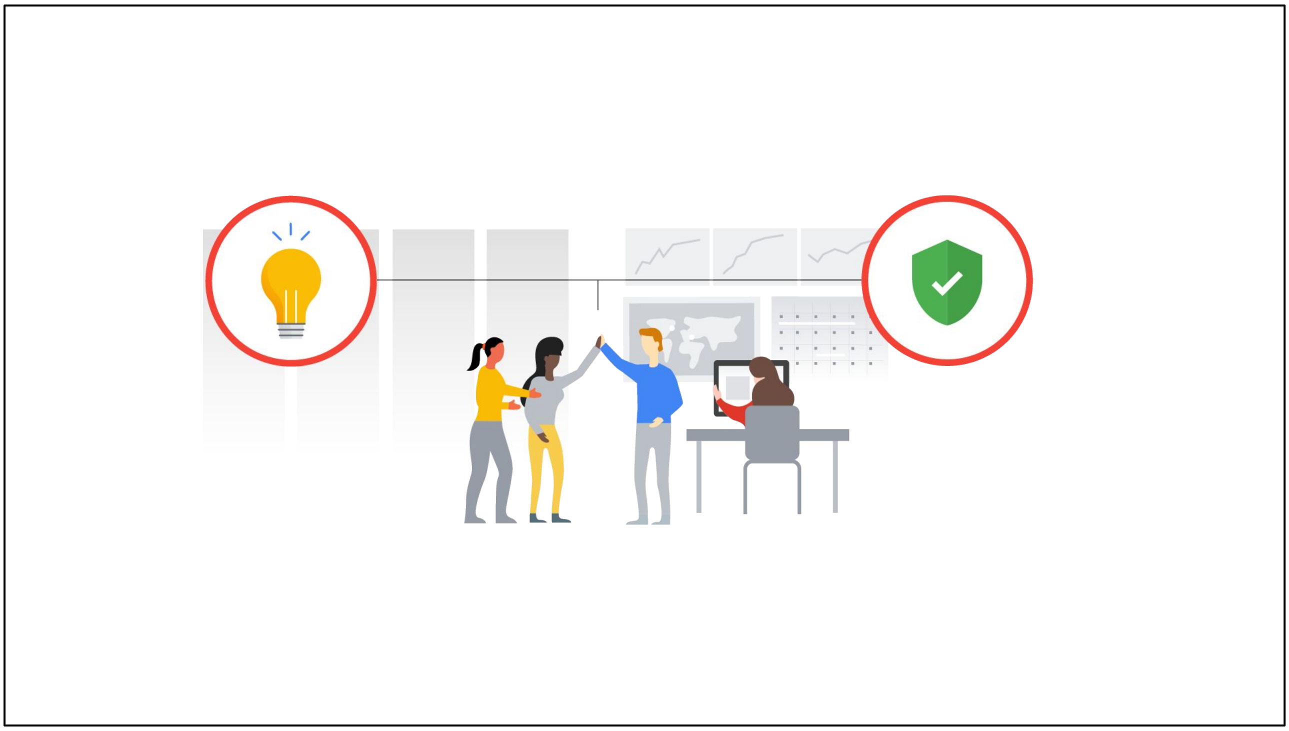


If you’re thinking that 100% reliability is what your goal should be, you’d be wrong. If you spend all your time on targeting 100% reliability, you’ll slow the release of new service features, which is what helps drive your business. This is where the error budget comes in.

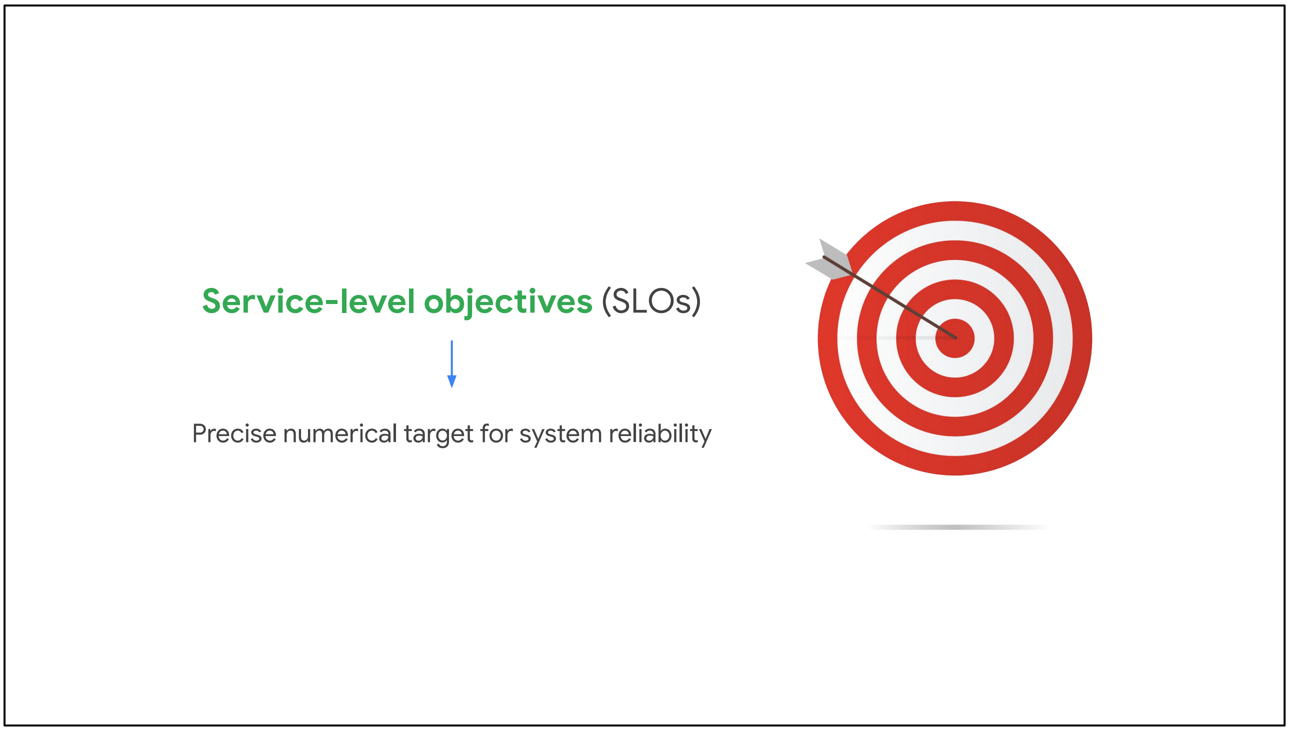


Error budgets help prioritize engineering work.

As long as the measured uptime is above your target—meaning, as long as you have error budget remaining—you can push new feature releases. Alternatively, you can spend this remaining budget on something else, such as expected system changes, inevitable failures in hardware and networks, planned downtime, or risky experiments. Essentially, an error budget is an agreement that **helps prioritize engineering work.**



Error budgets create a common incentive for developers and SREs to find the right balance between innovation and reliability. Developer teams can manage the error budget on their own, knowing that unrealistic reliability targets will slow down innovation. It **creates a shared responsibility** between teams for system uptime, as infrastructure failures take away developers' error budget.

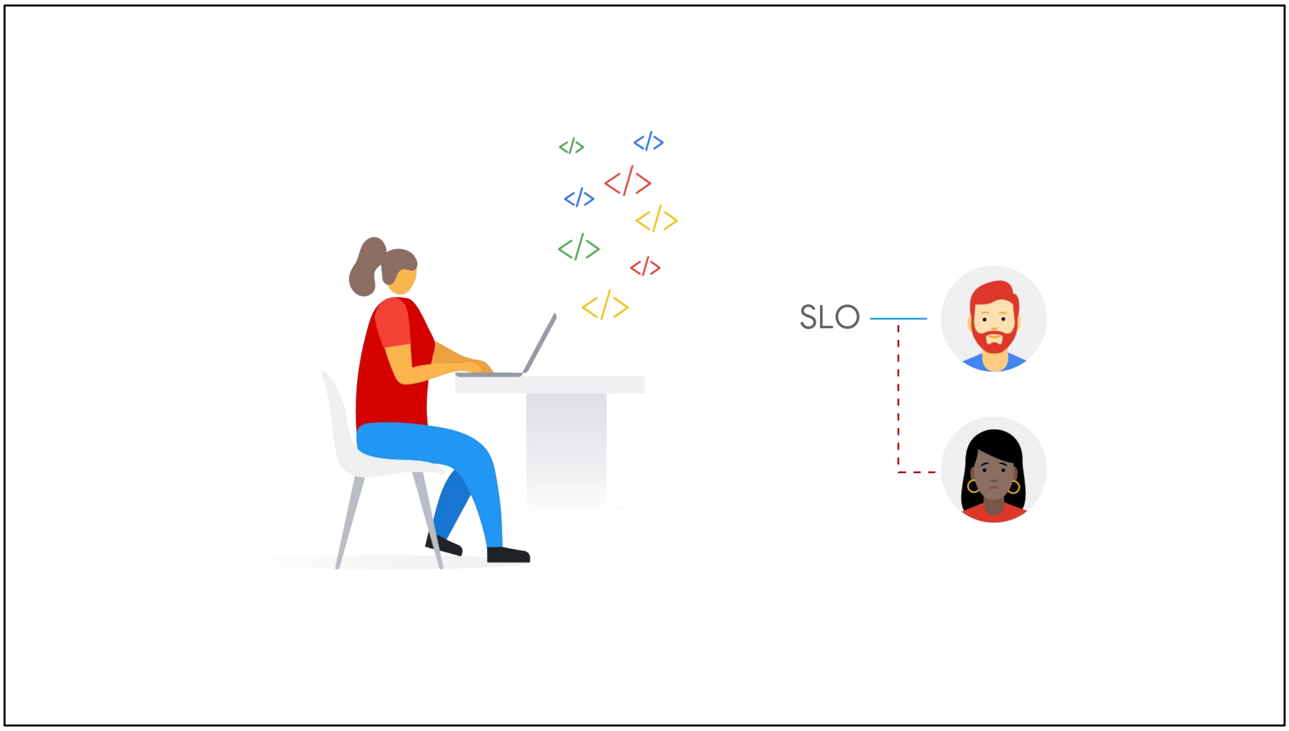


**Service-level objectives** (SLOs)

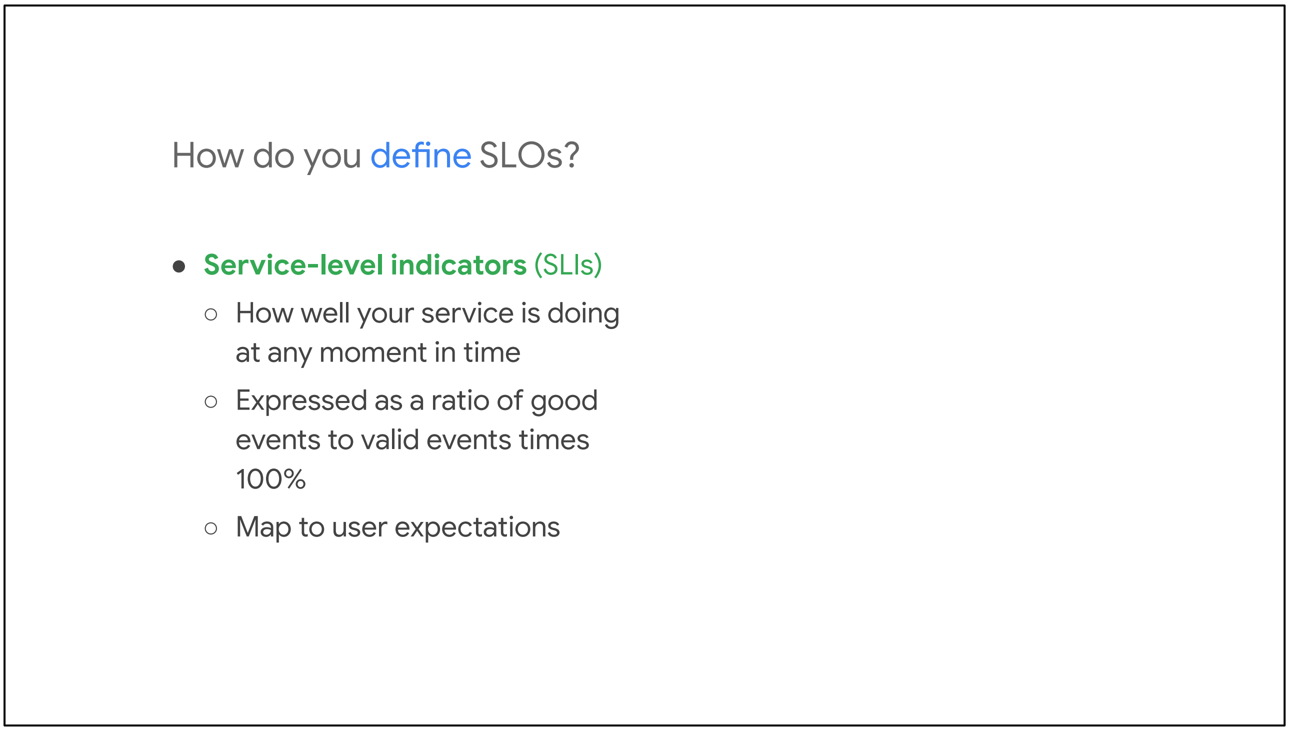
↓

Precise numerical target for system reliability

Another SRE practice that ties to error budgets and creates shared responsibility between dev and ops teams are service level objectives, or **SLOs**. SLOs are precise numerical targets for system reliability. These targets are agreed upon between stakeholders, thereby sharing ownership of reliability, and hopefully mitigating any future confusion or conflict.



Since you want your dev teams to work at a high velocity, SLOs can help them determine how fast is too fast. Measuring SLO performance gives a real-time indication of the reliability cost of new features. If everyone agrees that the SLO represents the point at which you are no longer meeting the expectations of your users, then broadly speaking, being well within SLO is a signal that you can move faster without causing those users pain.

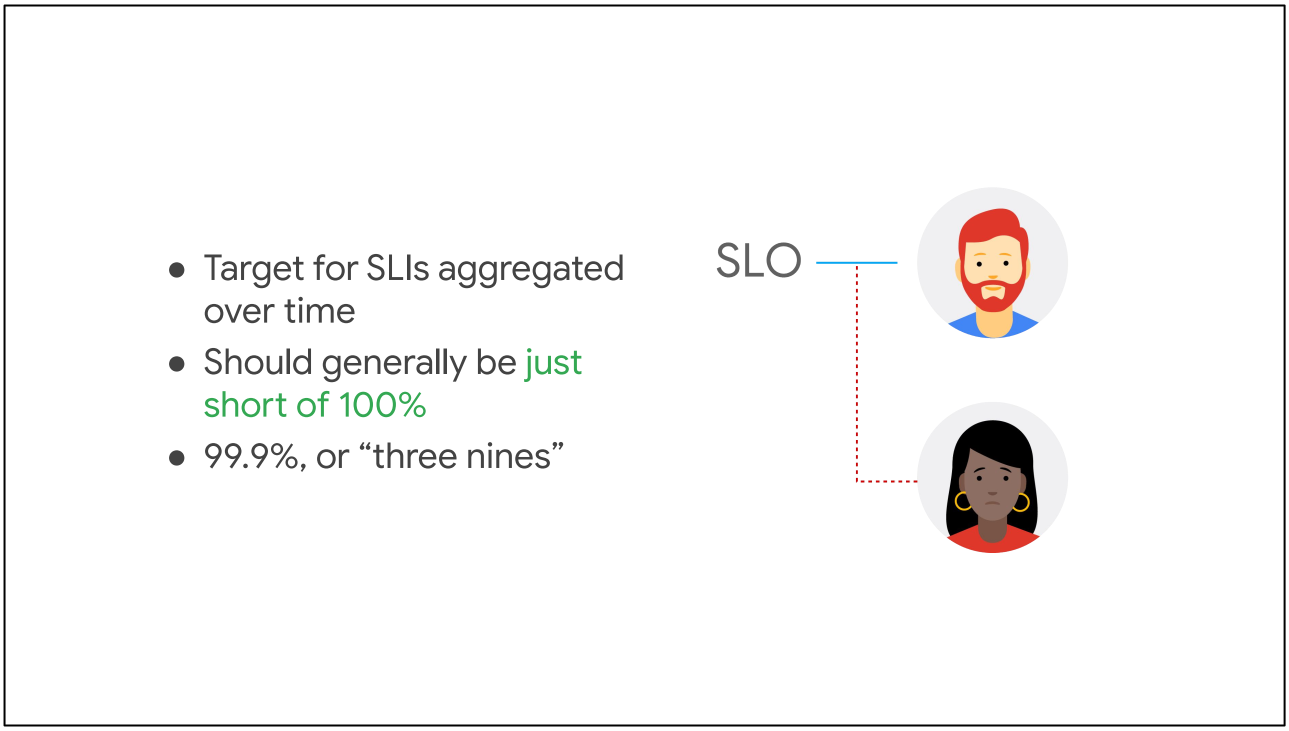

## How do you define SLOs?

- **Service-level indicators** (SLIs)
    - How well your service is doing at any moment in time
    - Expressed as a ratio of good events to valid events times 100%
    - Map to user expectations

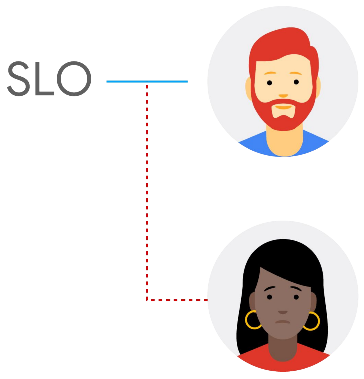
***How* can you define SLOs for your service?**

Well, you first need to understand service level indicators, or **SLIs**. An SLI tells you at any moment in time **how well your service is doing**. It's a quantifiable measure of service reliability. We recommend that SLIs are **expressed as the number of good events divided by the number of all valid events, times 100%**. This gives you a range of 0% to 100%, where zero equals nothing works and 100 equals nothing's broken.

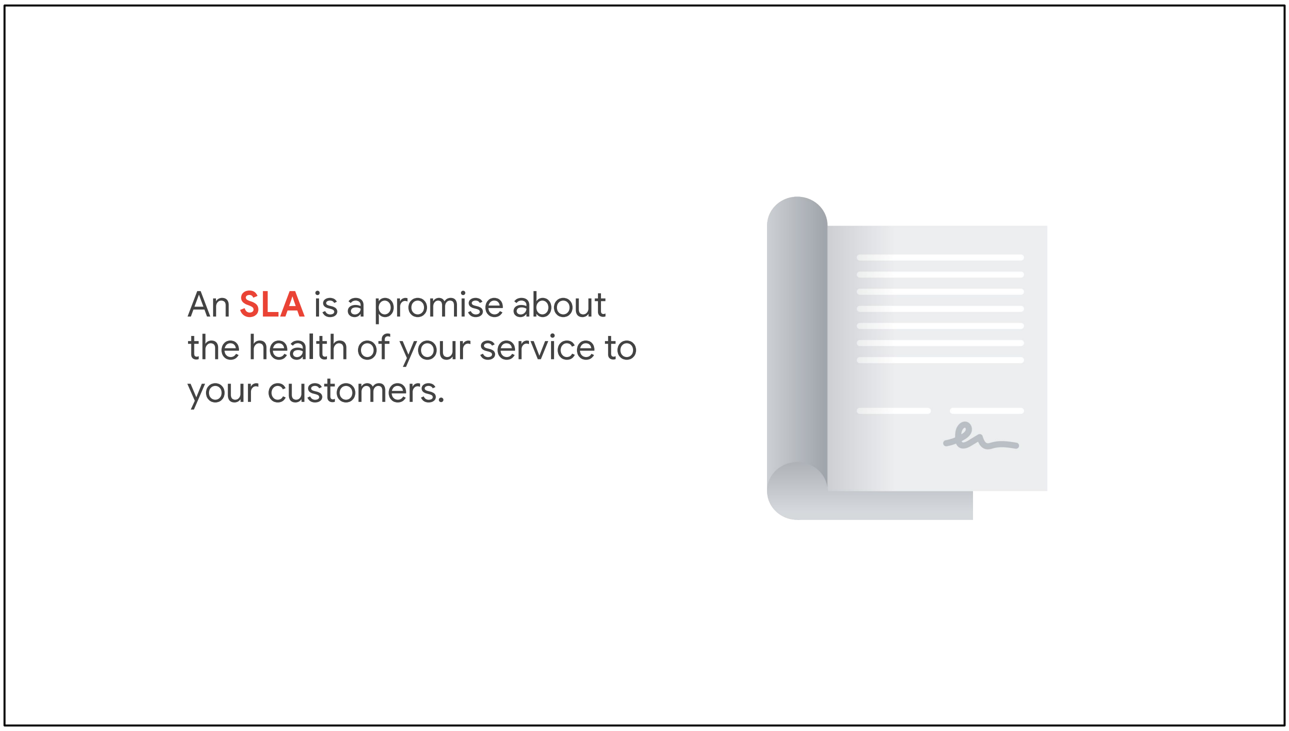
SLIs should **map to user expectations**, such as response time latency and quality, data processing correctness and freshness, or storage latency and throughput.



- Target for SLIs aggregated over time
- Should generally be just short of 100%
- 99.9%, or “three nines”



And so, an SLO is your **target for SLIs aggregated over time**. Assuming you’ve expressed your SLIs as a percentage between 0 and 100, your SLOs should generally **be just short of 100%**, like 99.9%, or **“three nines.”** SLOs draw the line between happy and unhappy customers. A typical customer should be happy if you barely meet your SLO. Anything below your SLO will result in an unhappy customer whose expectations for reliable service are not being met.

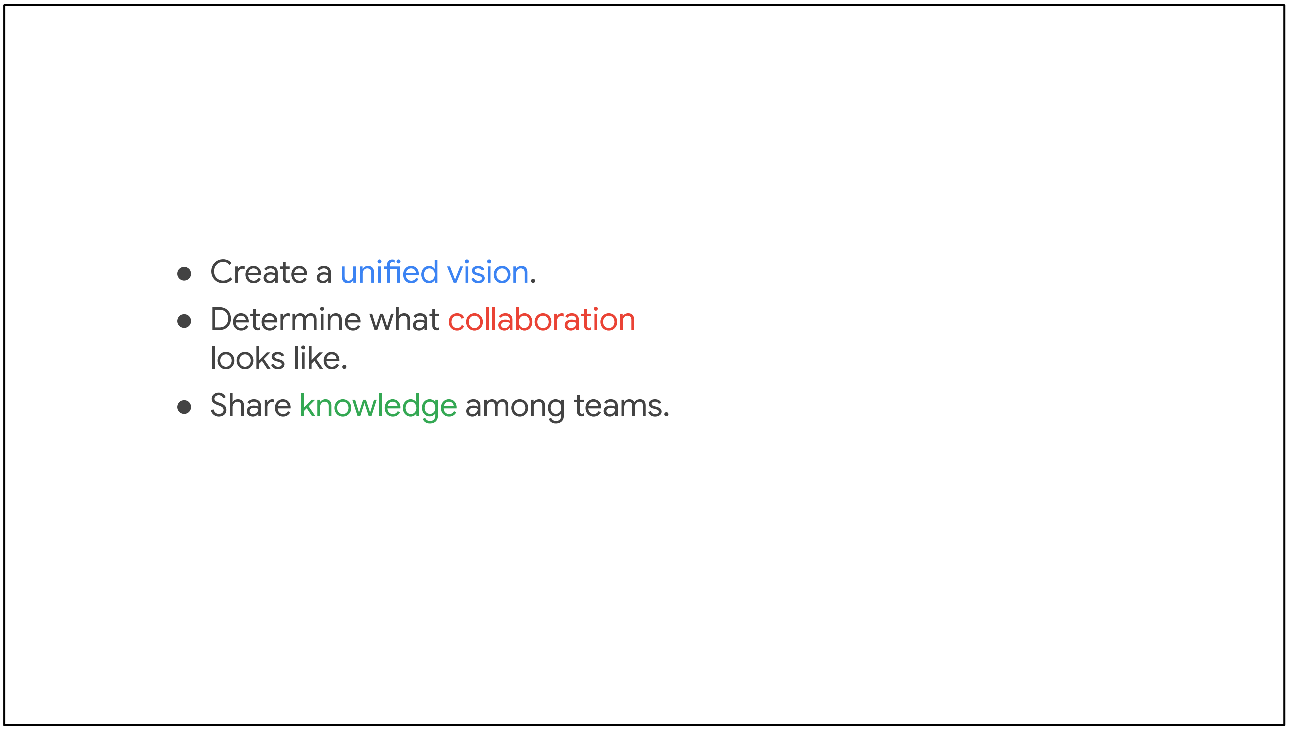


An **SLA** is a promise about the health of your service to your customers.

Lastly, an SLA, which you are likely familiar with already, is **the promise you make about your service's health to your customers**. It defines what your business is willing to do if you fail to meet your SLOs; for example, refunding money.

In the next video we'll discuss the SRE cultural practices that align with the practices of SLOs and error budgets.

The practices of defining SLOs and error budgets help to reduce and break down silos in organizations focused on SRE. It’s also important to understand and build certain cultural practices in your business to help support these technical practices.

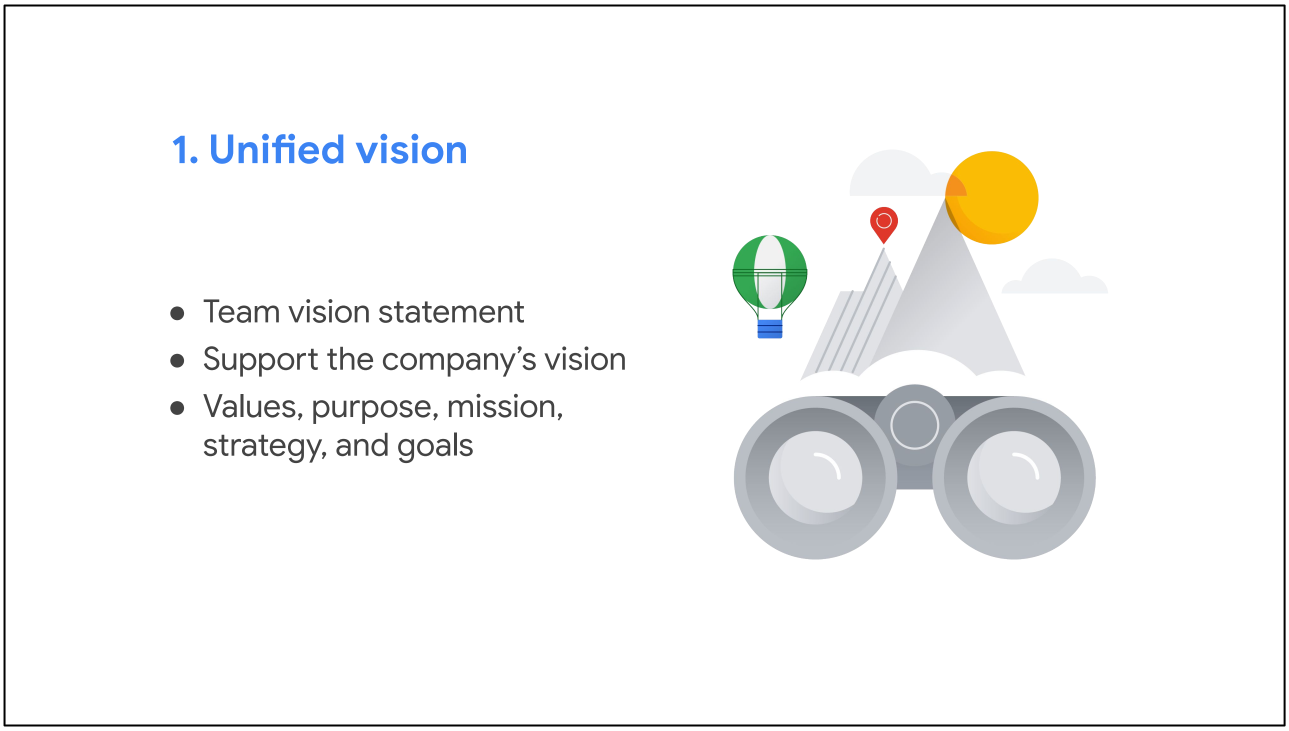


- Create a unified vision.
- Determine what collaboration looks like.
- Share knowledge among teams.

Specifically, organizations developing SRE culture should focus on:

- Creating a unified vision
- Determining what collaboration looks like
- Sharing knowledge among teams

Let’s take a closer look at each of these.



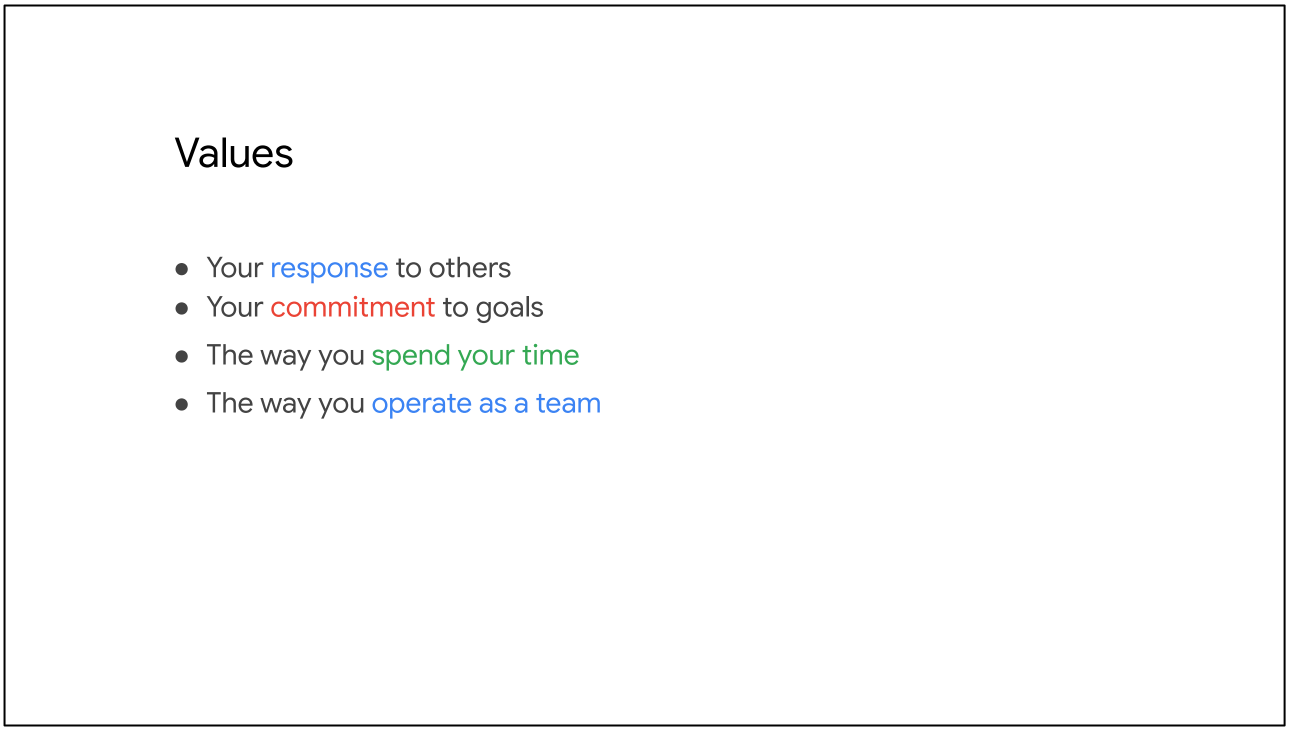
## 1. Unified vision

- Team vision statement
- Support the company’s vision
- Values, purpose, mission, strategy, and goals

**Unified vision**

All companies have a **vision statement** that serves as their guide for the work they do. To give a sense of direction, your IT team’s vision should **support the company’s vision.**

A team’s vision is everything about what drives its work, and includes **core values, purpose, mission, strategy, and goals.** What is your company’s vision? How about your team’s vision? Does it encompass all these things? If you’re unsure, think about your team vision statement as you learn about each aspect that should be included in it.

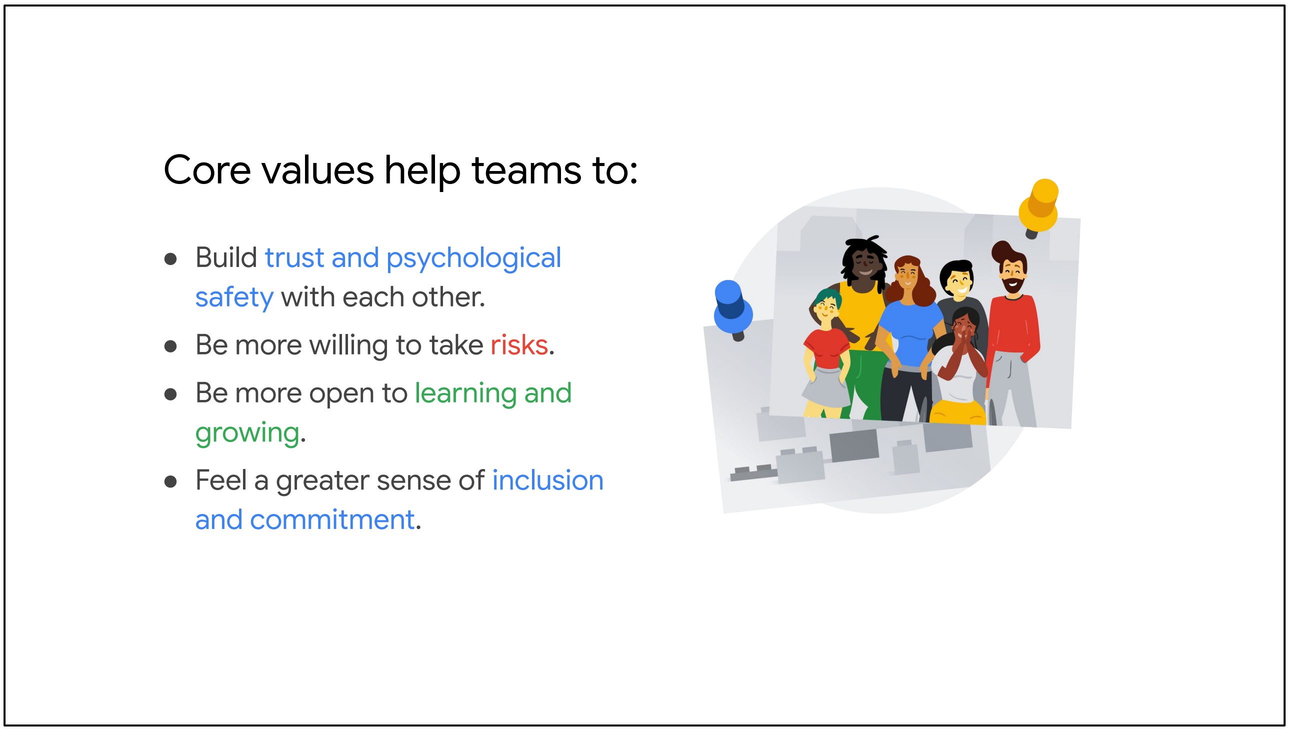


## Values

- Your response to others
- Your commitment to goals
- The way you spend your time
- The way you operate as a team

First, **values** refer to how you can achieve your vision, and what guides behaviors. Values can be expressed by:

- Your response to others
- Your commitments to personal and organizational goals
- The way you spend your time
- The way you operate as a team

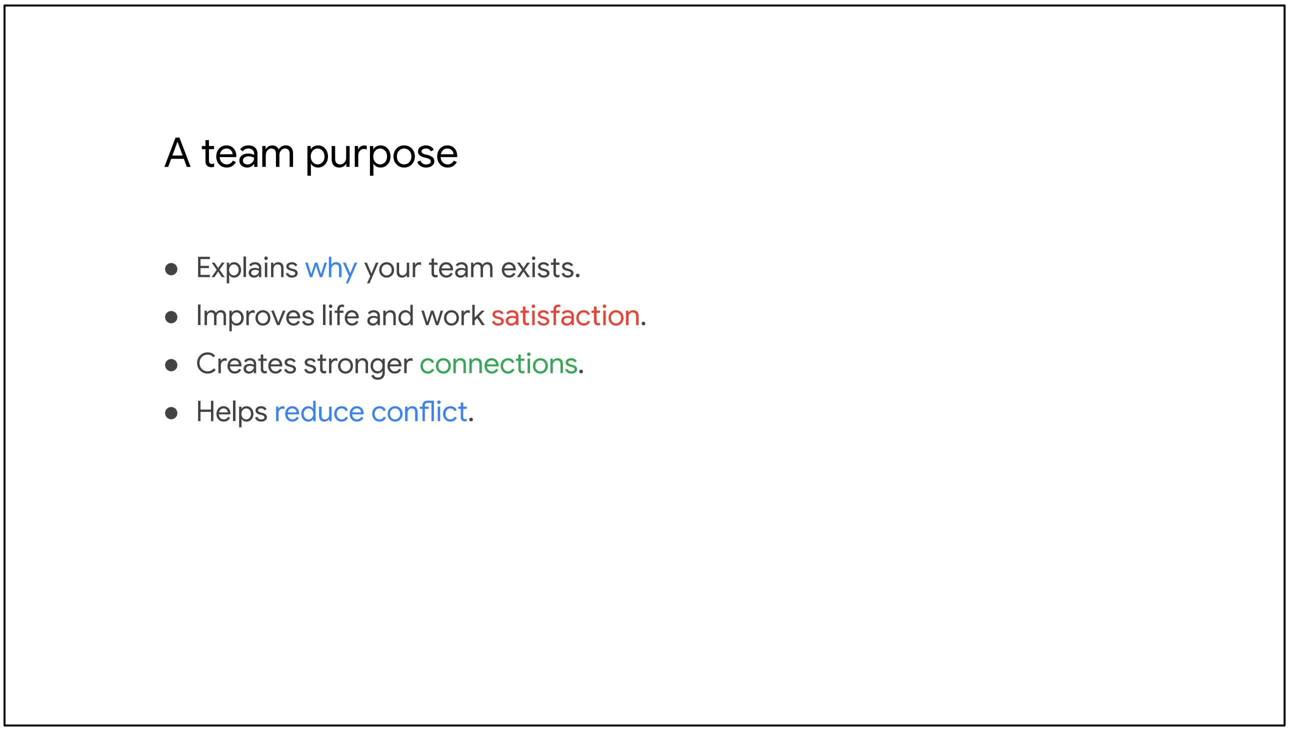


## Core values help teams to:

- Build trust and psychological safety with each other.
- Be more willing to take risks.
- Be more open to learning and growing.
- Feel a greater sense of inclusion and commitment.

By developing core values, you can help your team:

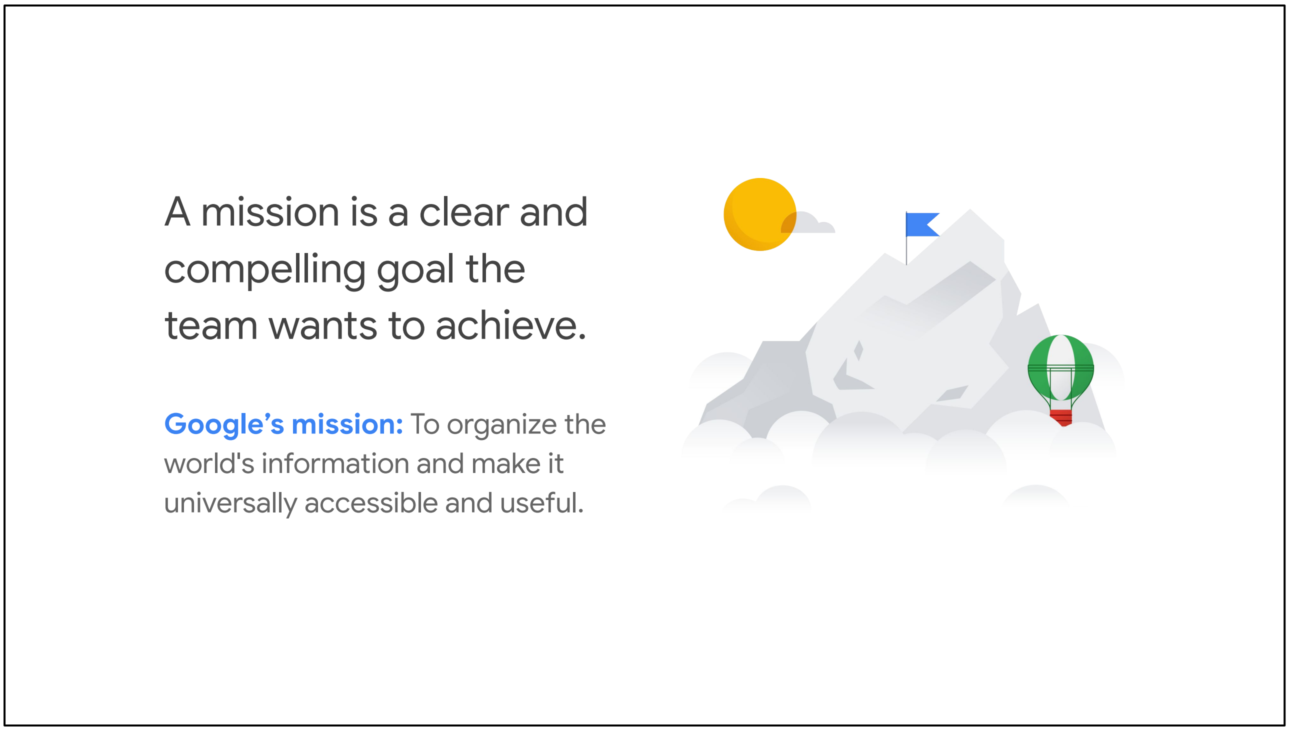
- Build trust and psychological safety with each other
- Be more willing to take risks
- Be more open to learning and growing
- Feel a greater sense of inclusion and commitment



## A team purpose

- Explains why your team exists.
- Improves life and work satisfaction.
- Creates stronger connections.
- Helps reduce conflict.

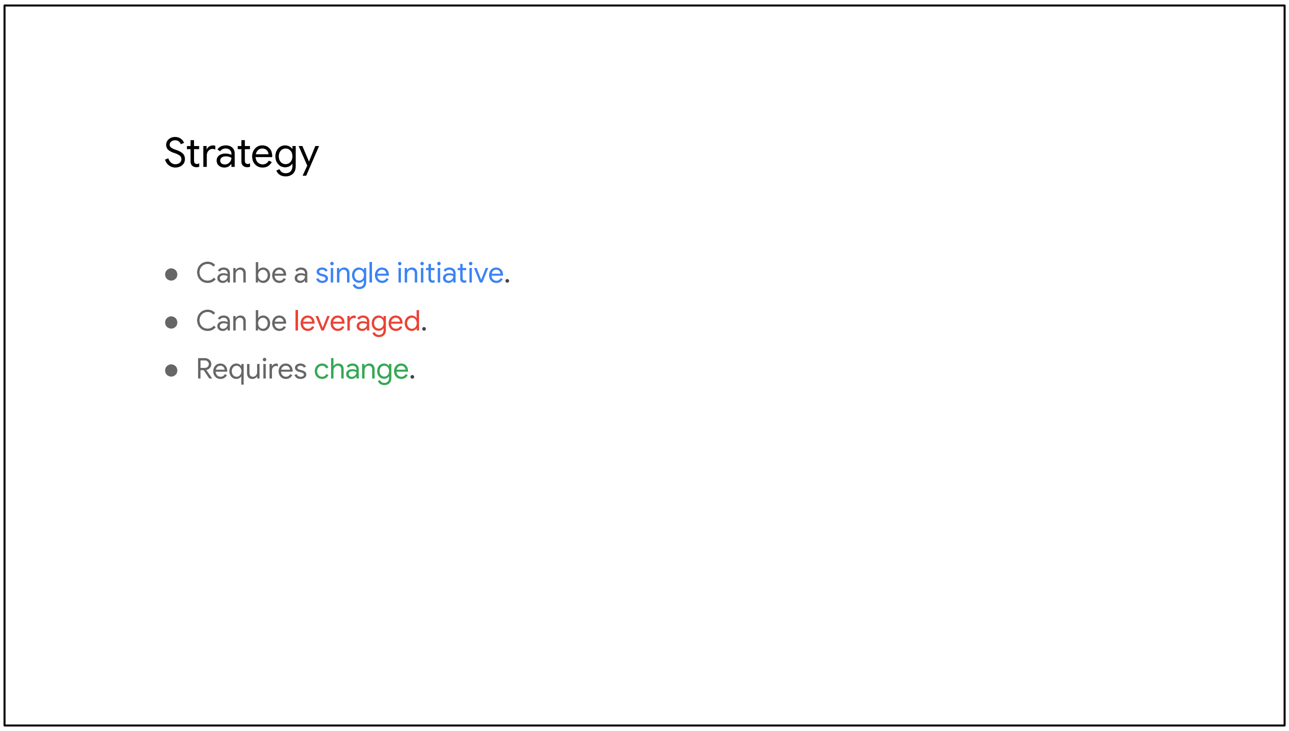
A team's **purpose** refers to why it exists. Internal research by Google has shown that teams that have a purpose and meaning to their work have higher life and work satisfaction, stronger intra-team connections, and less conflict.



A mission is a clear and compelling goal the team wants to achieve.

**Google's mission:** To organize the world's information and make it universally accessible and useful.

Team **mission** articulates a clear and compelling goal that the team strives to achieve. For example, Google's mission statement is to organize the world's information and make it universally accessible and useful. The Google Cloud Change and Culture team's mission statement is to bring Google's unique people, culture, and change management expertise to Google Cloud Enterprise customers, helping them to successfully embed new behaviors and new ways of working via adoption of Google Cloud technology.



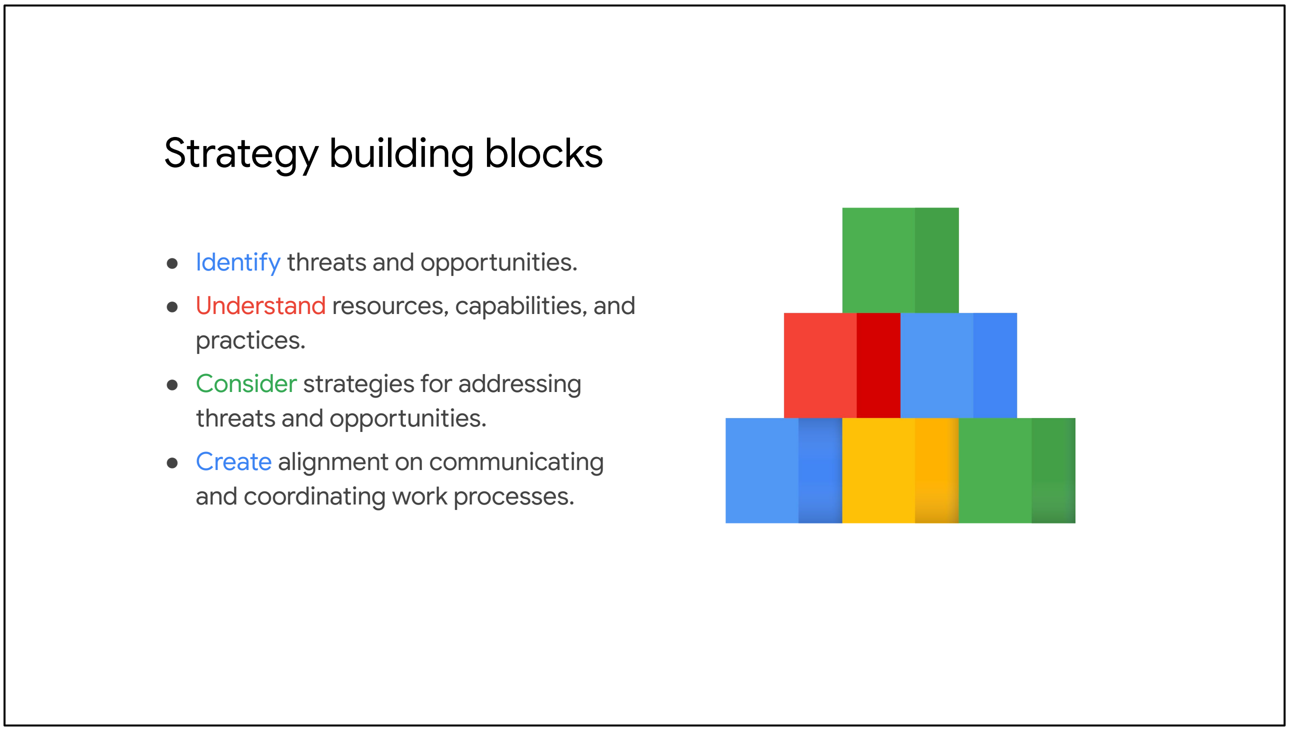
## Strategy

- Can be a single initiative.
- Can be leveraged.
- Requires change.

Your team's **strategy** is how your team will realize its mission.

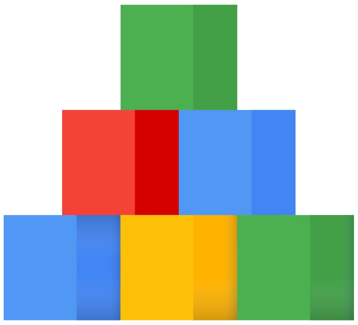
Strategic action takes on many forms.

- It can be a single initiative designed to meet a specific future goal.
- It can be leveraged: a single project can meet multiple future goals.
- It requires change: a change in investment of resources and people, and a change in habits and how work gets done.



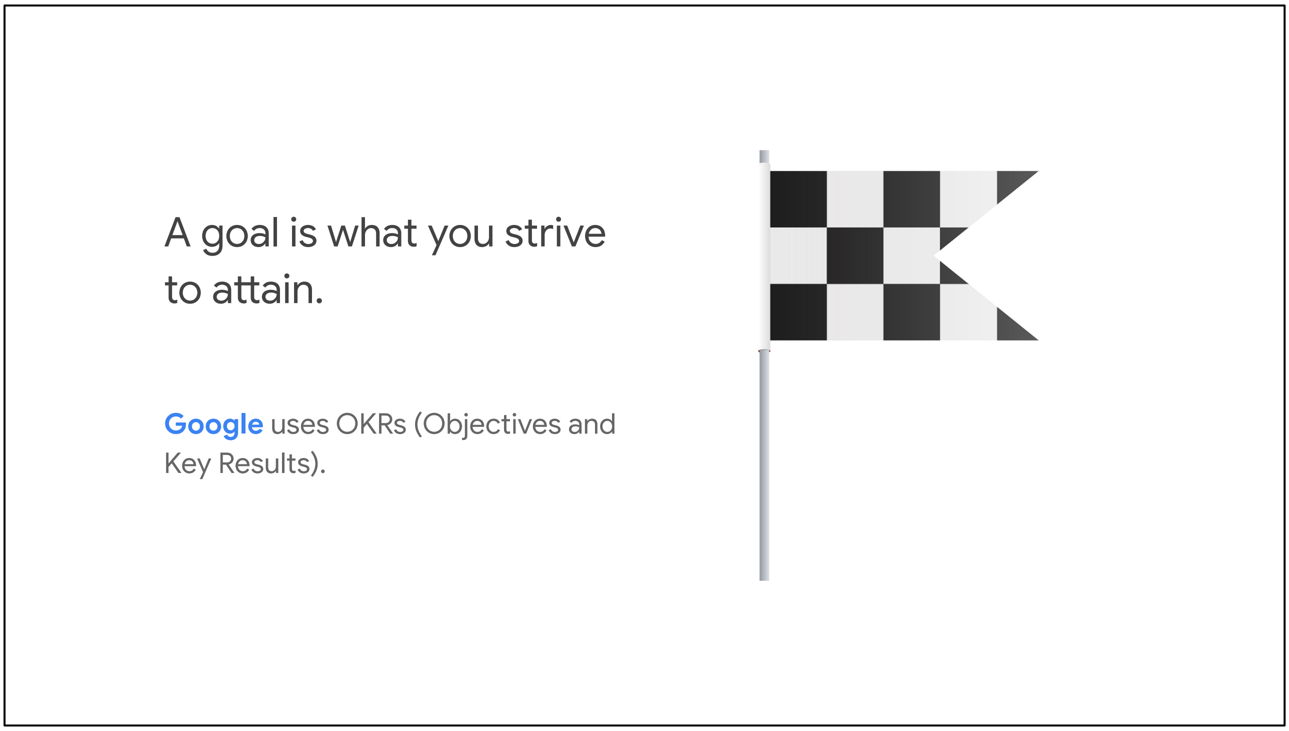
## Strategy building blocks

- Identify threats and opportunities.
- Understand resources, capabilities, and practices.
- Consider strategies for addressing threats and opportunities.
- Create alignment on communicating and coordinating work processes.



Some basic building blocks of strategy are to:

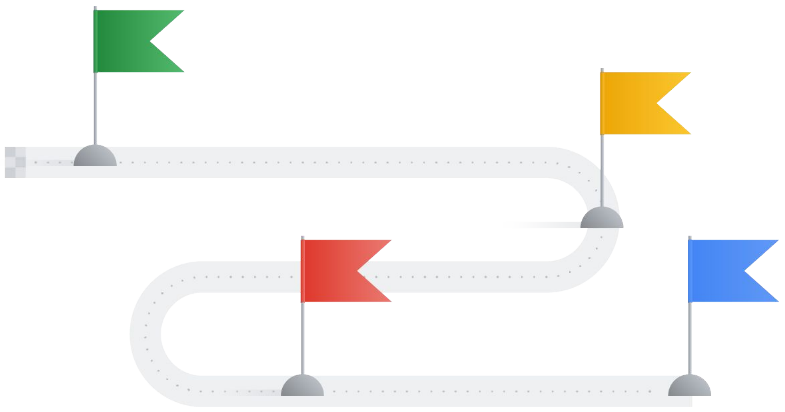
- Look outside to identify threats and opportunities
- Look inside at resources, capabilities, and practices
- Consider strategies for addressing threats and opportunities
- Create alignment on communicating and coordinating work processes



A goal is what you strive to attain.

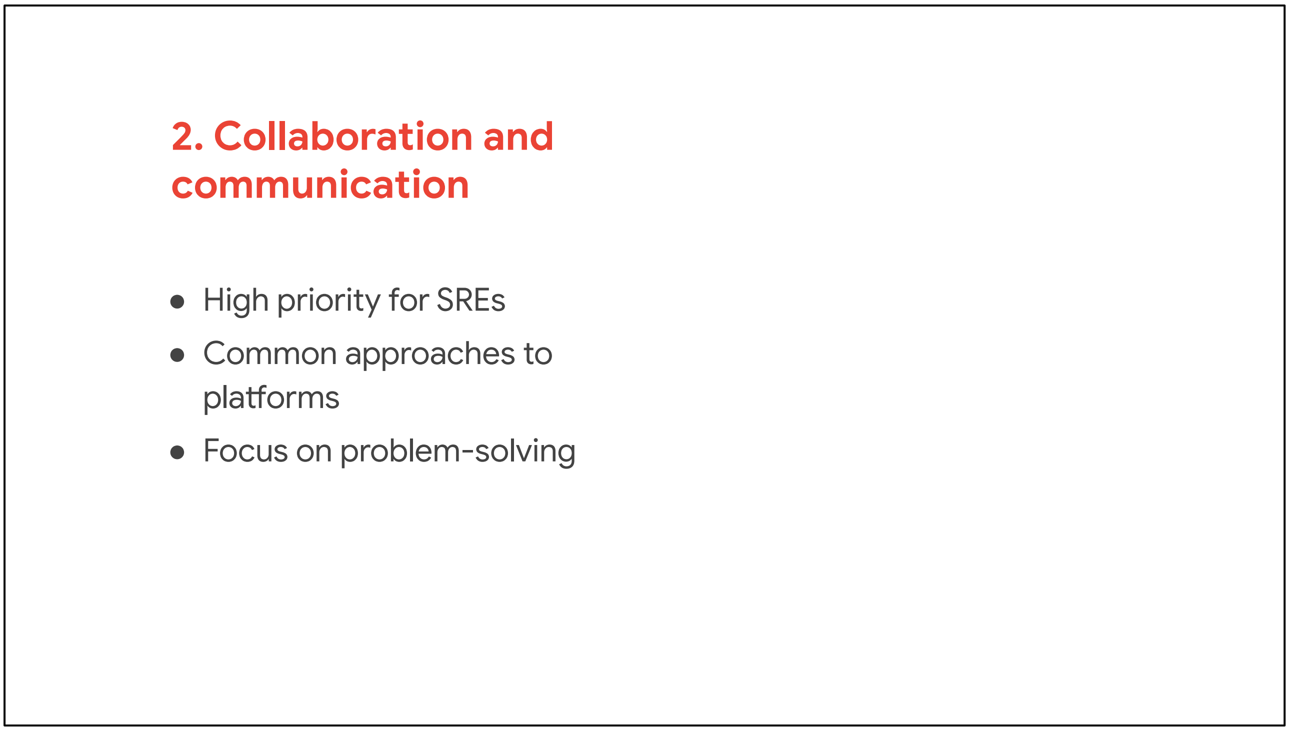
Google uses OKRs (Objectives and Key Results).

Lastly, setting specific **goals** aligns your team on what they are striving to attain. At Google, we use OKRs, which are Objectives and Key Results, to set ambitious goals and track our progress.



Try new things, prioritize work, and learn from success and failure.

In practice, using OKRs is different from other goal-setting techniques. The idea behind OKRs is to set very ambitious goals. When used this way, OKRs can enable teams to focus on the big bets and accomplish more than the team thought was possible, even if they don't fully attain their intended goal. OKRs can encourage people to **try new things, prioritize work, and learn from both success and failure**. And while the team may not reach every OKR, it gives them something to strive for together.



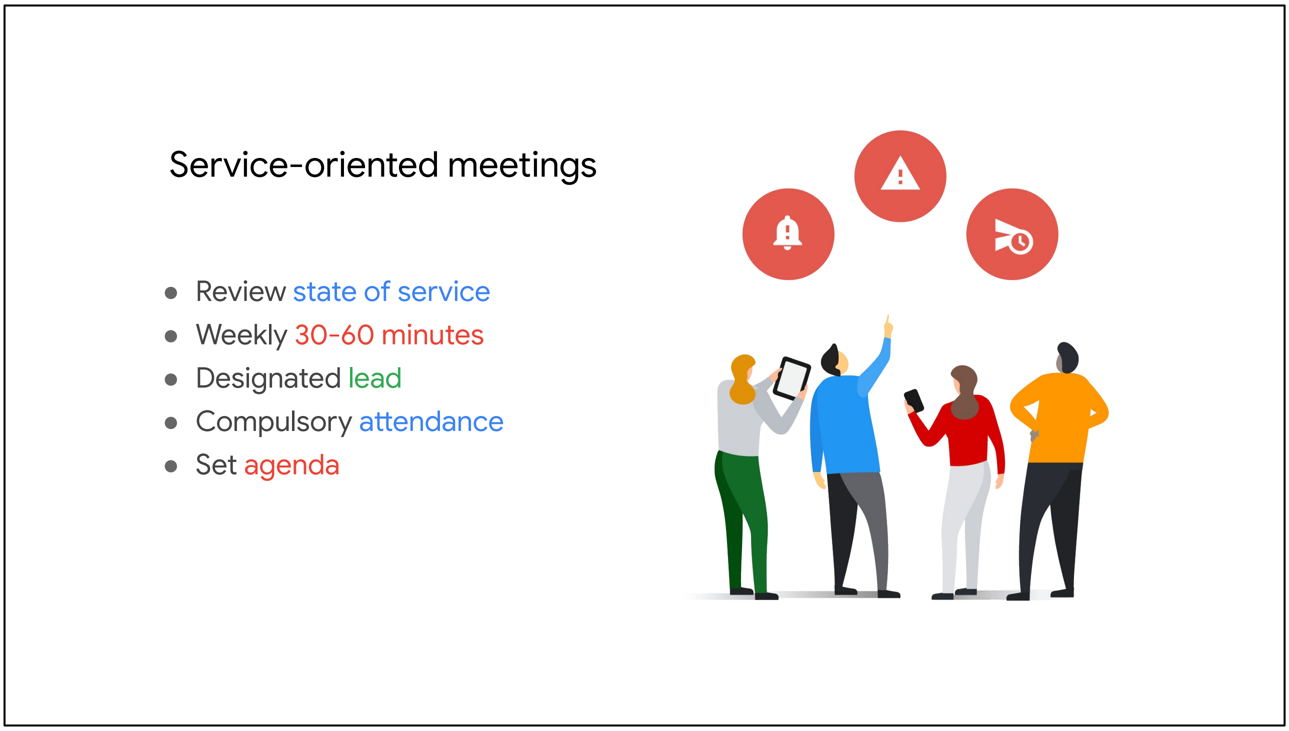
## 2. Collaboration and communication

- High priority for SREs
- Common approaches to platforms
- Focus on problem-solving

Next, let’s talk about **collaboration and communication**.

Given the globally distributed nature of SRE teams, effective communication has always been a **high priority in SRE**. Collaboration between SRE teams has its challenges, but has potentially great rewards, including **common approaches to platforms** for solving problems, which lets teams **focus on solving more difficult problems**.

Let’s look at some examples of how SRE teams can communicate effectively.

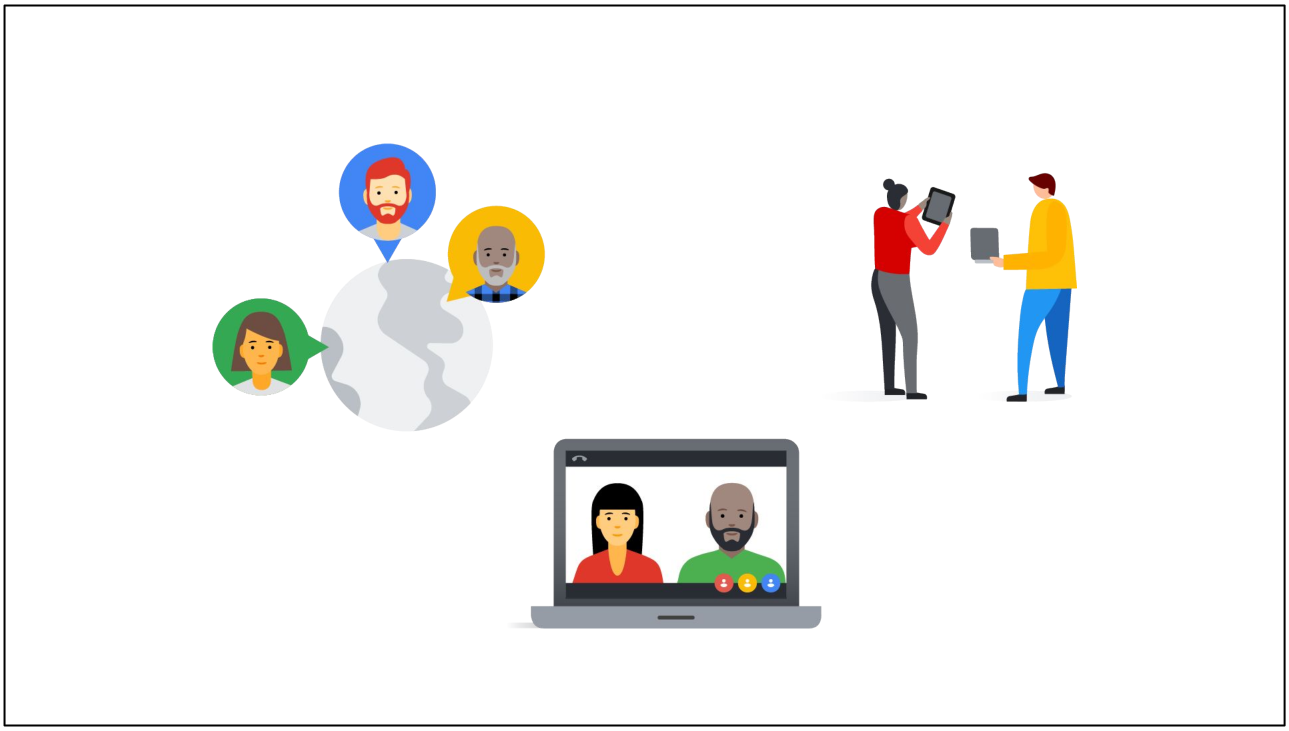


## Service-oriented meetings

- Review state of service
- Weekly 30-60 minutes
- Designated lead
- Compulsory attendance
- Set agenda

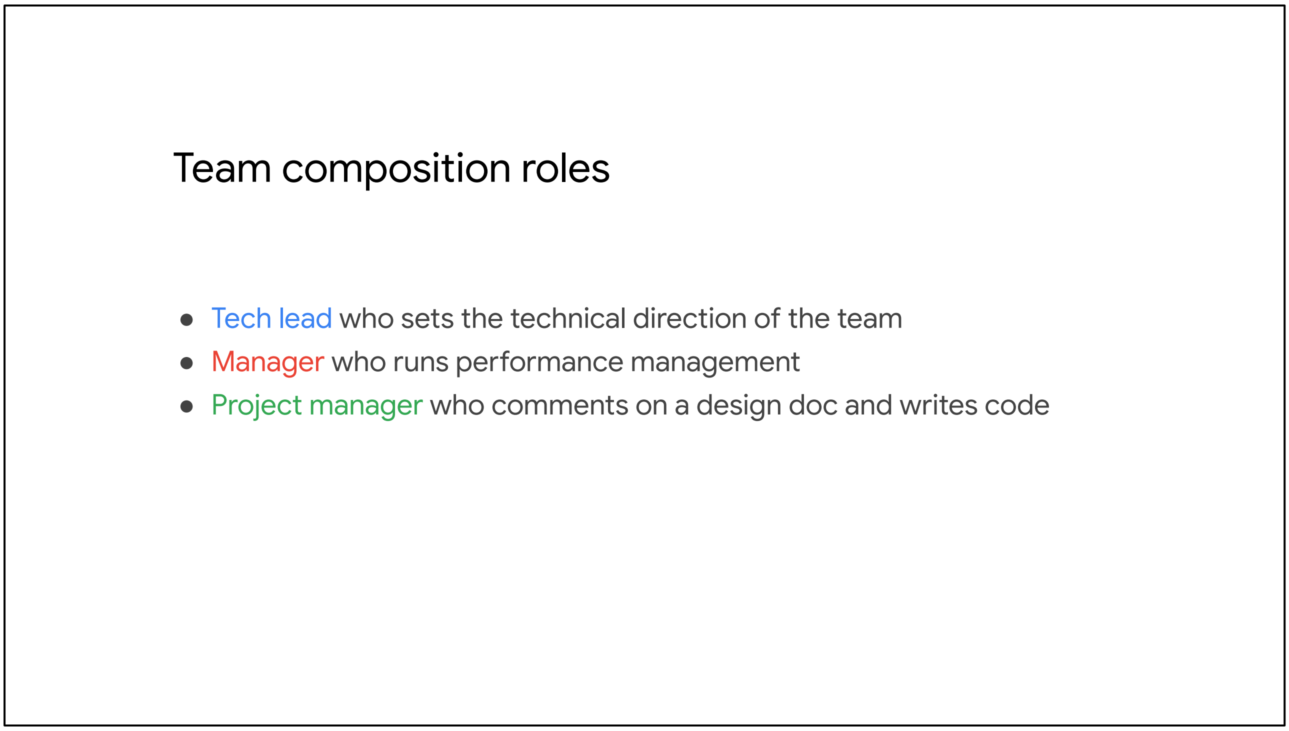
One way is through **service-oriented meetings.** This is a special kind of meeting where an SRE team reviews the **state of the service** or services in their charge to increase awareness of all stakeholders involved and to improve the operation of the service or services.

Service-oriented meetings usually happen **weekly for 30 to 60 minutes** and should have a **designated lead.**
**Attendance should be compulsory** for all the team members, because this is a major opportunity to interact as a group. **Setting a defined agenda** is important. For example, your agenda could be to cover upcoming production changes, metrics, outages, paging events, non-paging events, and prior action items.



Another way to create effective communication is to have a good **team composition**.

SRE is usually a distributed organization, spread across different countries and time zones. Because of this, the definition of *team* is variable. There can be local teams, the team on the site, the cross-continental team, various virtual teams, and everything in between.


## Team composition roles

- Tech lead who sets the technical direction of the team
- Manager who runs performance management
- Project manager who comments on a design doc and writes code

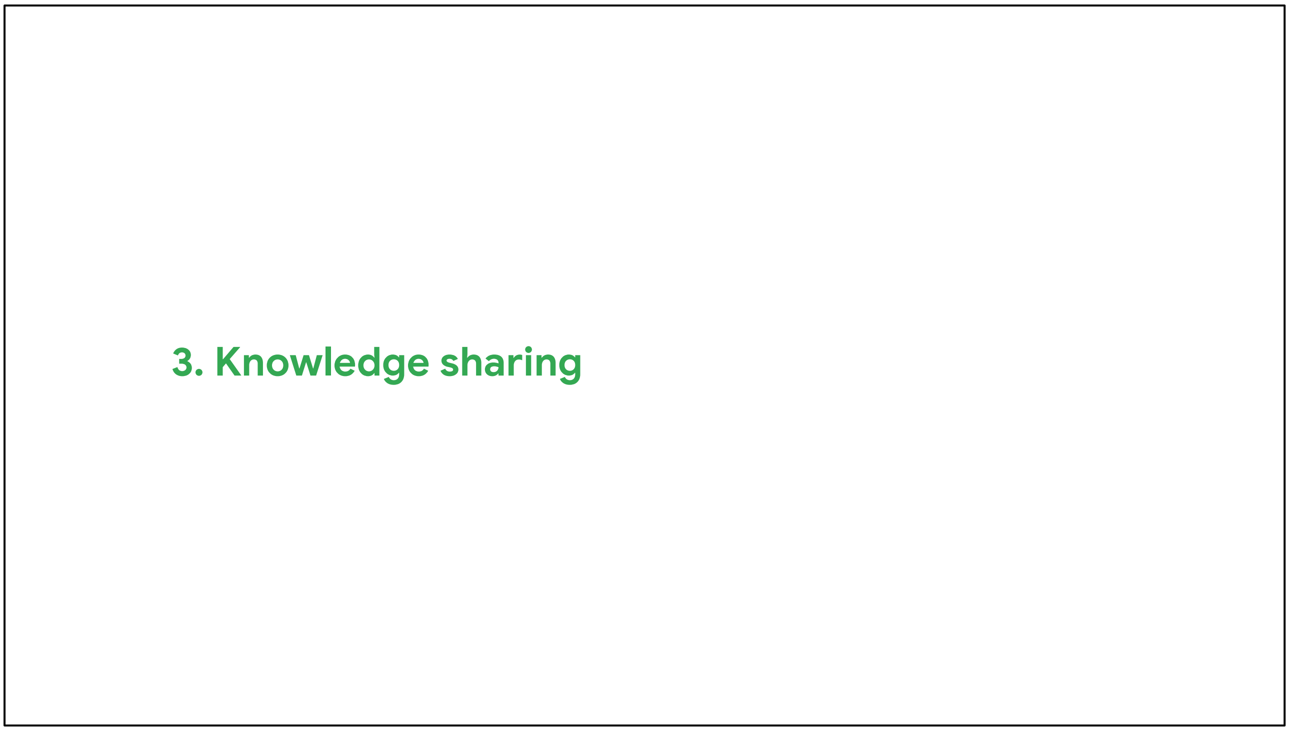
Google recommends a few specific roles in any SRE team:

- A **tech lead who sets the technical direction of the team**. This person comments on everyone's code, holds quarterly direction presentations, and builds consensus in the team.
- A **manager who runs performance management** and acts as the first point of contact for the team.
- A **project manager** who comments on a design doc and writes code.

For all these roles, excellent communication skills are required to effectively collaborate across time zones. In Module 6 you will learn more about suggested SRE team compositions and skills.

Collaboration between SRE teams and other teams is equally as important as communication.

Specifically, effective collaboration between product development and SRE teams is vital. This collaboration is at its best when it occurs as early in the design phase as possible; that is, before any line of code has been committed. SREs usually make recommendations about architecture and software behavior. They also use OKRs to track progress of their work with other teams. At Google, it's common for cross-functional teams to share an OKR, creating a shared agreement on output.

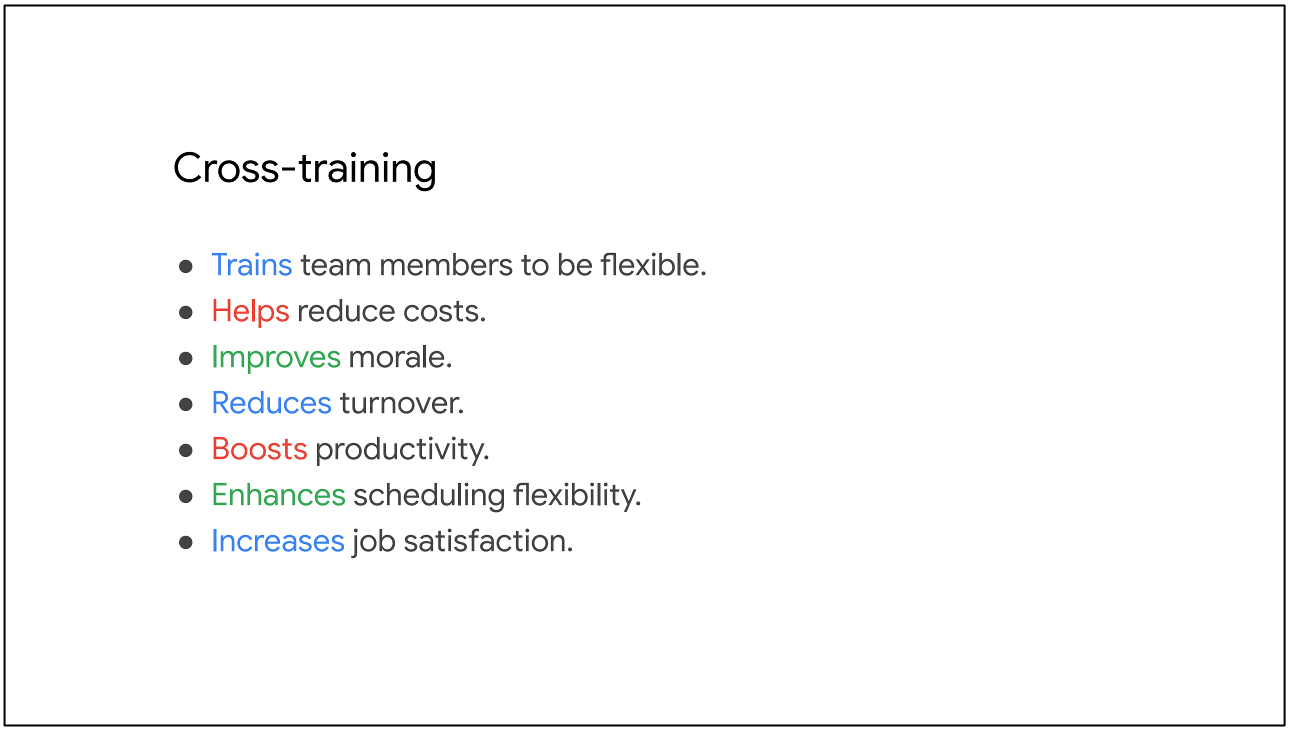


## 3. Knowledge sharing

Lastly, let’s look at how **knowledge sharing** helps reduce organizational silos.

Well hired and trained SRE teams are adept at performing more than one job function and have the skills to step into another as needed. In order to create this, your organization needs to focus on cultivating knowledge sharing among its team members.

There are a few ways to ensure that this happens on your team.

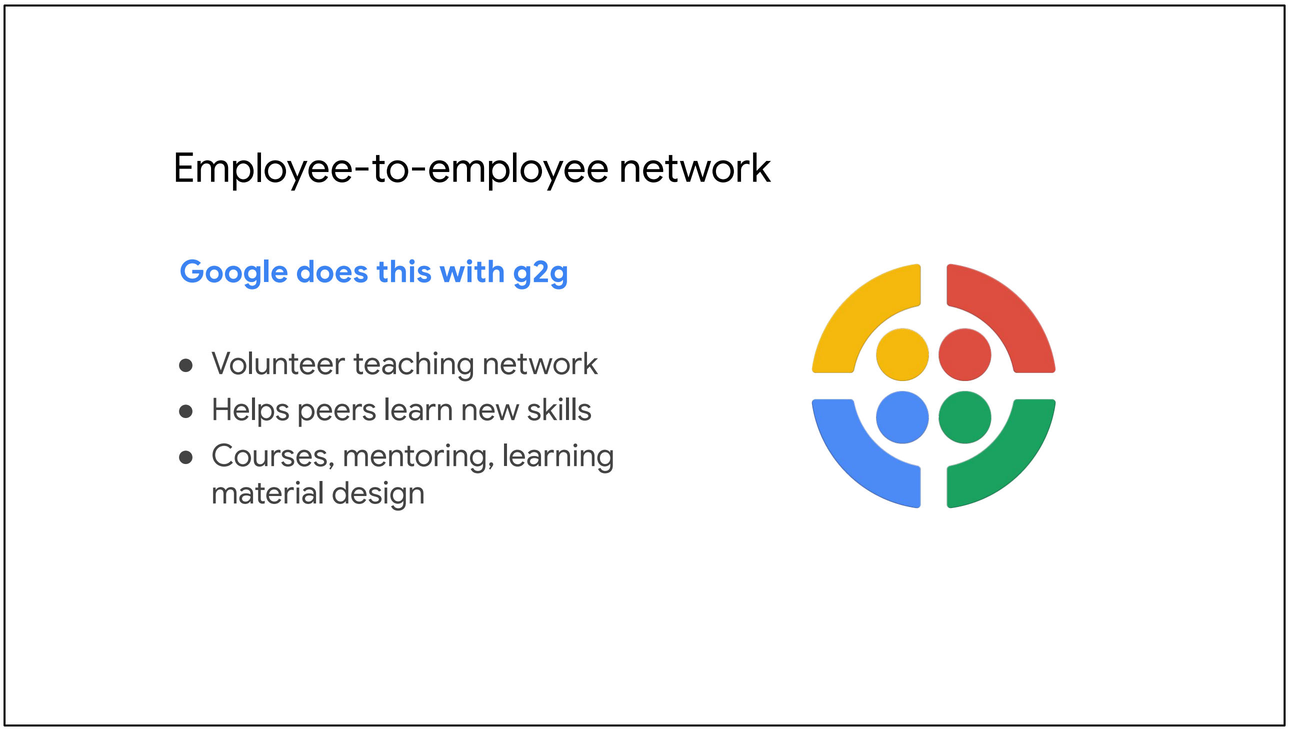


## Cross-training

- Trains team members to be flexible.
- Helps reduce costs.
- Improves morale.
- Reduces turnover.
- Boosts productivity.
- Enhances scheduling flexibility.
- Increases job satisfaction.

The first is to make **cross-training** a key competency when hiring SREs. This involves **training an employee for a flexible response** to changing production schedules. In practice, it means teaching an employee who was hired to perform one job function the skills required to perform other job functions.

A well-designed SRE cross-training program can **help reduce costs**, **improve employee morale**, **reduce turnover, and increase productivit**y. It can also give a company greater **scheduling flexibility**, and may even lead to operational improvements. Perhaps the most important benefit to companies that implement cross-training programs is **greater job satisfaction** among employees. Cross-training demonstrates that the company has faith in employees' abilities and wants to provide them with opportunities for career growth.



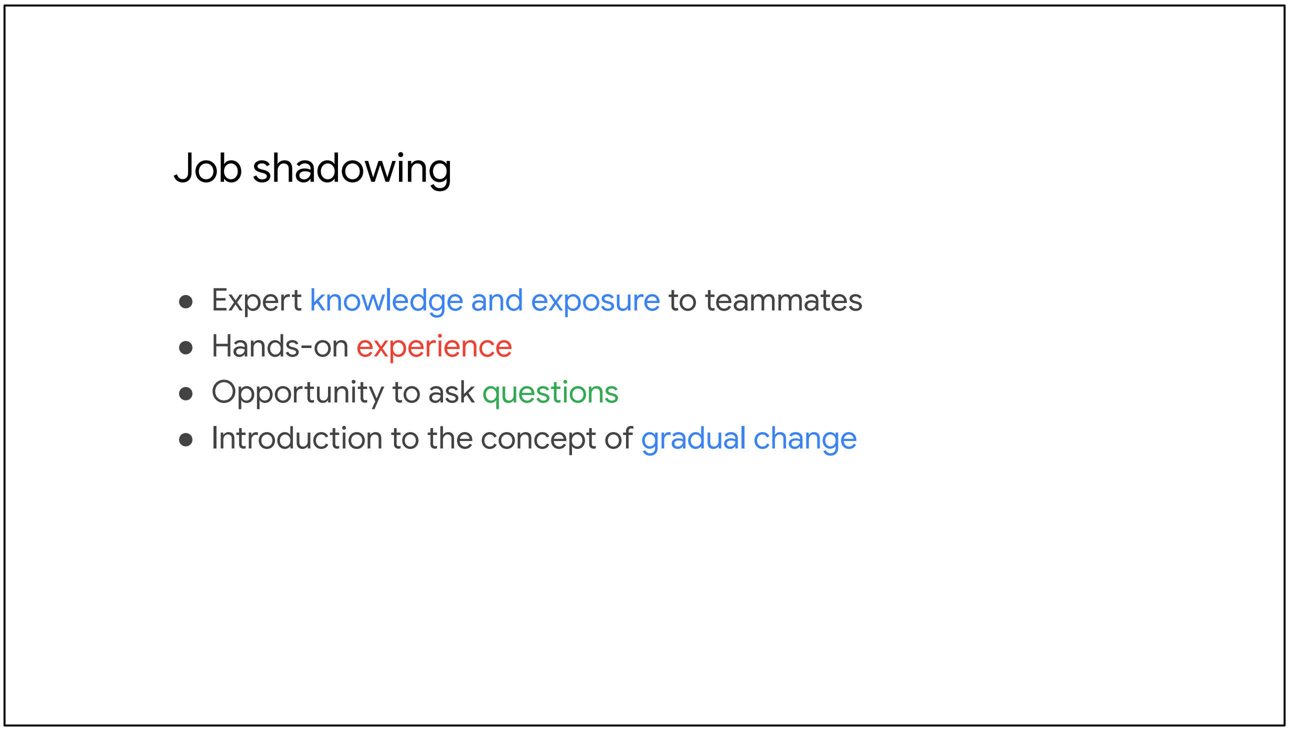
## Employee-to-employee network

**Google does this with g2g**

- Volunteer teaching network
- Helps peers learn new skills
- Courses, mentoring, learning material design

Another practice Google recommends is creating an **employee-to-employee network**. Employees can develop and grow by teaching others, given that they have first-hand knowledge on a topic.

At Google, 80% of all tracked trainings are run through an employee-to-employee network called **g2g, or Googler-to-Googler.** The **volunteer teaching network** of Google employees dedicates a portion of its time to **helping peers learn new skills**. Volunteers, known internally as g2gers, can participate in a variety of ways, such as **teaching courses, providing 1:1 mentoring, and designing learning materials**. They come from every department at Google.

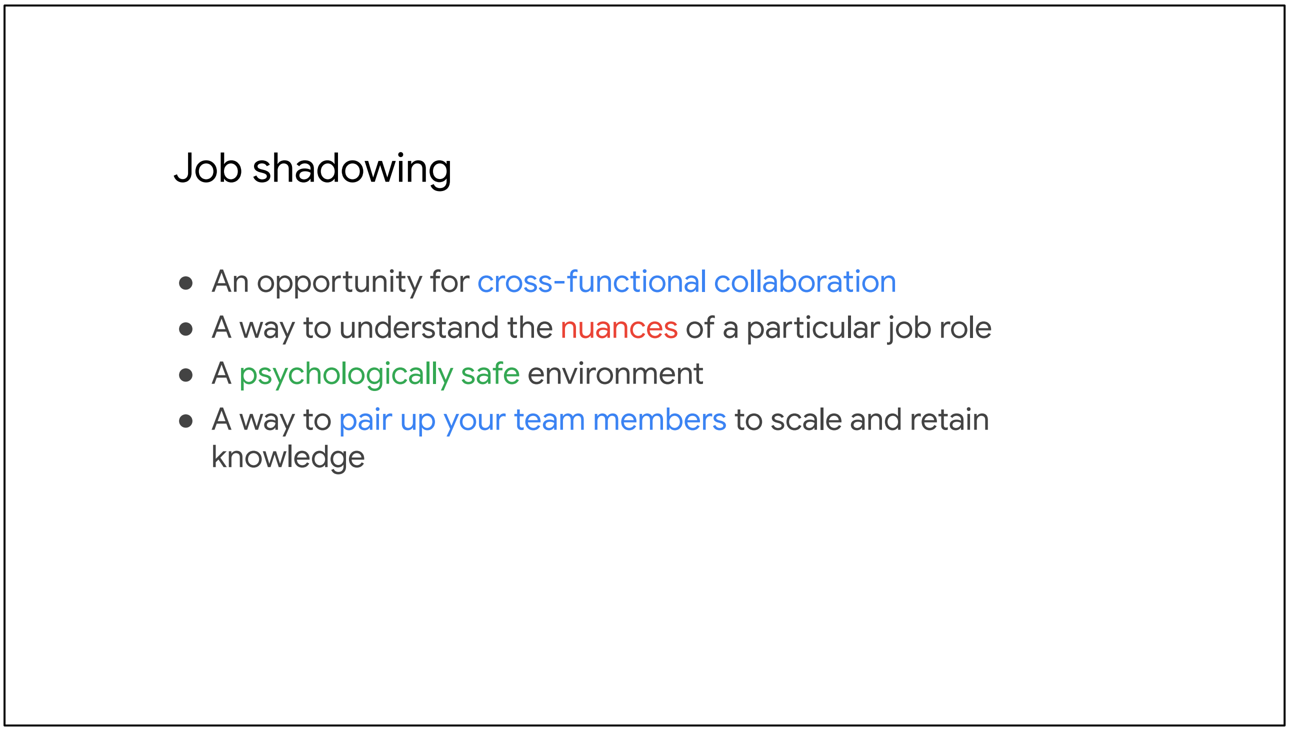


## Job shadowing

- Expert knowledge and exposure to teammates
- Hands-on experience
- Opportunity to ask questions
- Introduction to the concept of gradual change

Thirdly, **job-shadowing** is an effective means of generating knowledge sharing. Some examples of job-shadowing benefits in IT teams include:

- Expert knowledge and exposure for new hires to what others in the team do every day
- Hands-on experience for how the system should be maintained
- An opportunity to ask an expert any questions
- A good introduction to the concept of gradual change and a broader explanation of what it means to the team

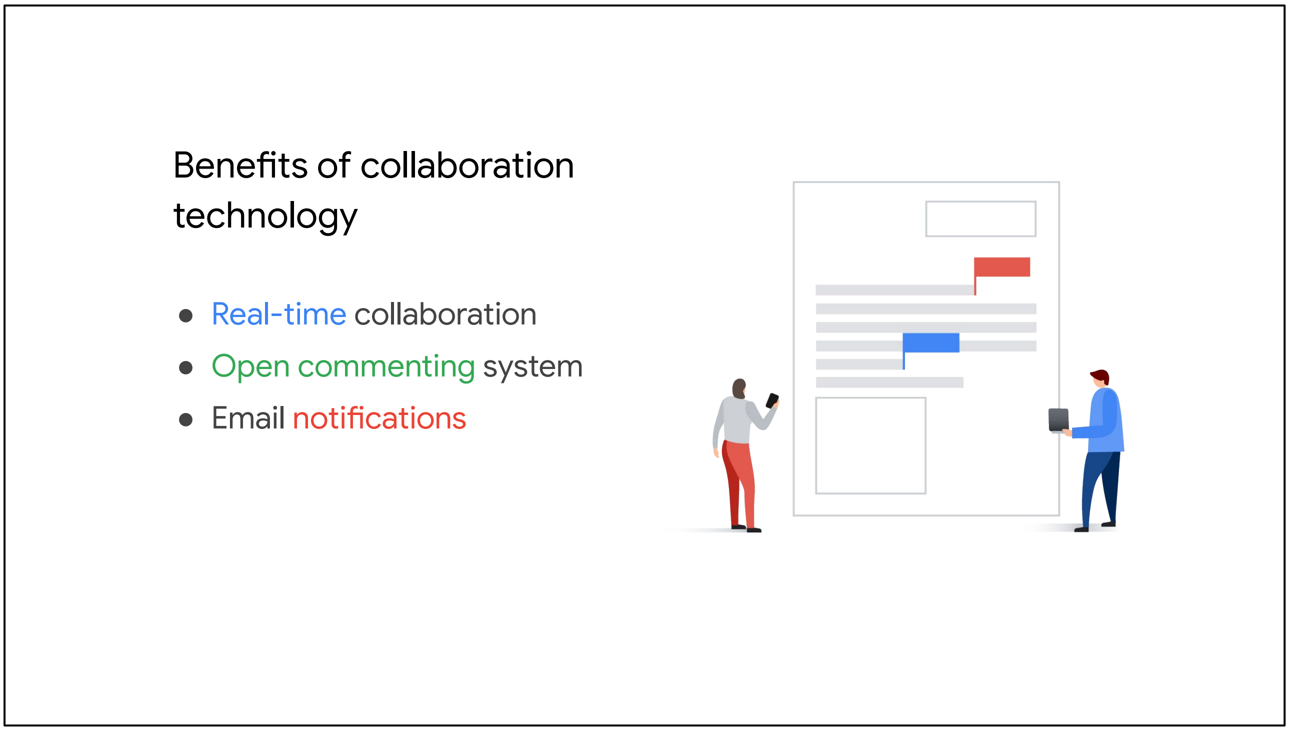


## Job shadowing

- An opportunity for cross-functional collaboration
- A way to understand the nuances of a particular job role
- A psychologically safe environment
- A way to pair up your team members to scale and retain knowledge

- A way to spot opportunities for cross-functional collaboration
- A great way to understand the nuances of what a particular job role entails
- A psychologically safe environment where it’s normal to ask questions and learn
- A way to pair up your team members that helps to scale and retain knowledge

Lastly, it’s worth noting that although we align postmortem practice to the “Accept failure as normal” pillar of DevOps, it also has overlap in reducing organizational silos, specifically with knowledge sharing. Postmortems are an SRE practice that helps you learn from your mistakes, but the means of delivering a postmortem helps with fostering collaboration and knowledge sharing.

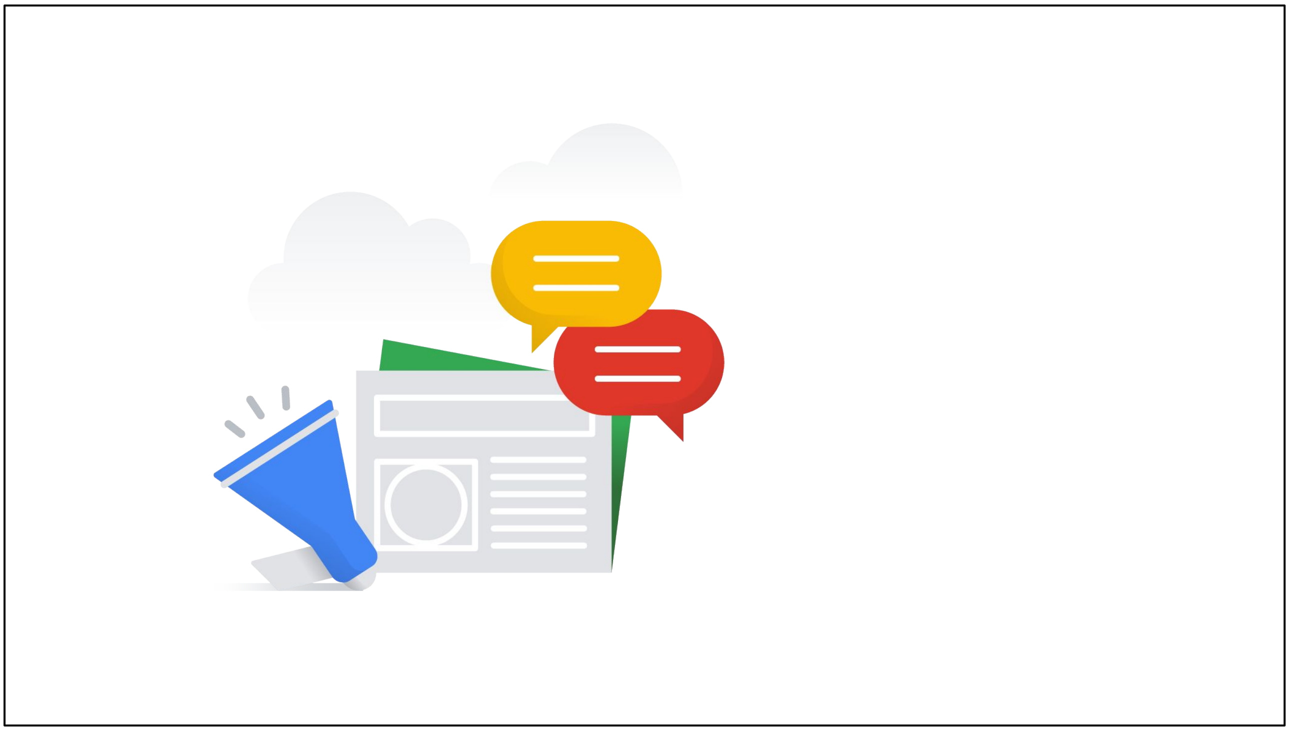


## Benefits of collaboration technology

- Real-time collaboration
- Open commenting system
- Email notifications

Postmortem workflows include collaboration and knowledge sharing at every stage. It's crucial to use technology, such as Google Docs, that enables some key features:

- **Real-time collaboration:** Enables data and ideas collection
- **An open commenting/annotation system:** Makes crowdsourcing solutions easy and improves coverage
- **Email notifications**: Directs collaborators or is used to loop others to provide input



At Google, we’ve helped numerous customers figure out ways to employ the cultural SRE practices of collaboration, communication, and knowledge sharing in their organizations.

Take a look at a top provider of online solutions in Germany. This customer formed their SRE team and, through an onsite Google workshop on SRE culture fundamentals, they were able to identify the rules of engagement for this new team. They highlighted the importance of reducing the number of communications channels and clearly describing what channel should be used for what purpose. They invested in technology that helped them collaborate in real time directly in project documents via comments or chat. They also used the collaborative tools to create their SRE knowledge repository.

You can see how these SRE cultural practices are just as important as the technical practices to help reduce silos within IT teams.

In the next module, the SRE journey continues with making tomorrow better than today, where you'll learn about SRE practices that align with implementing gradual change and leveraging tooling and automation.