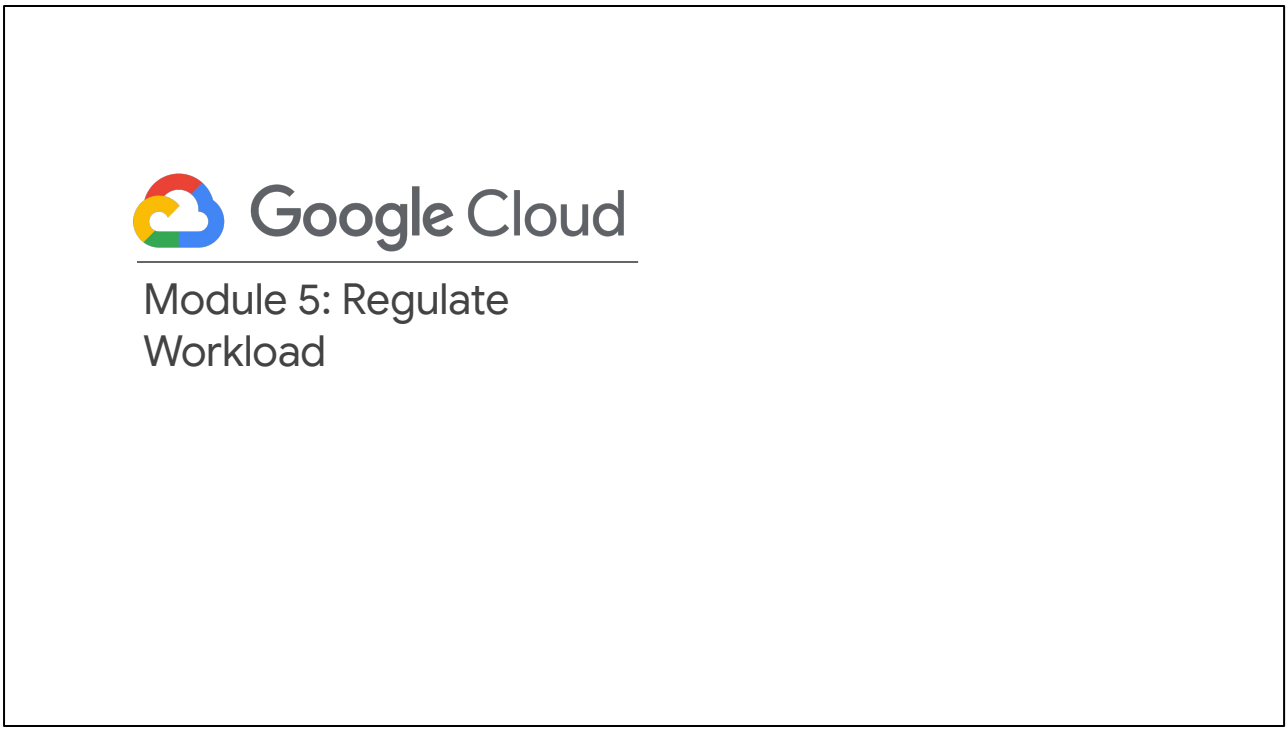


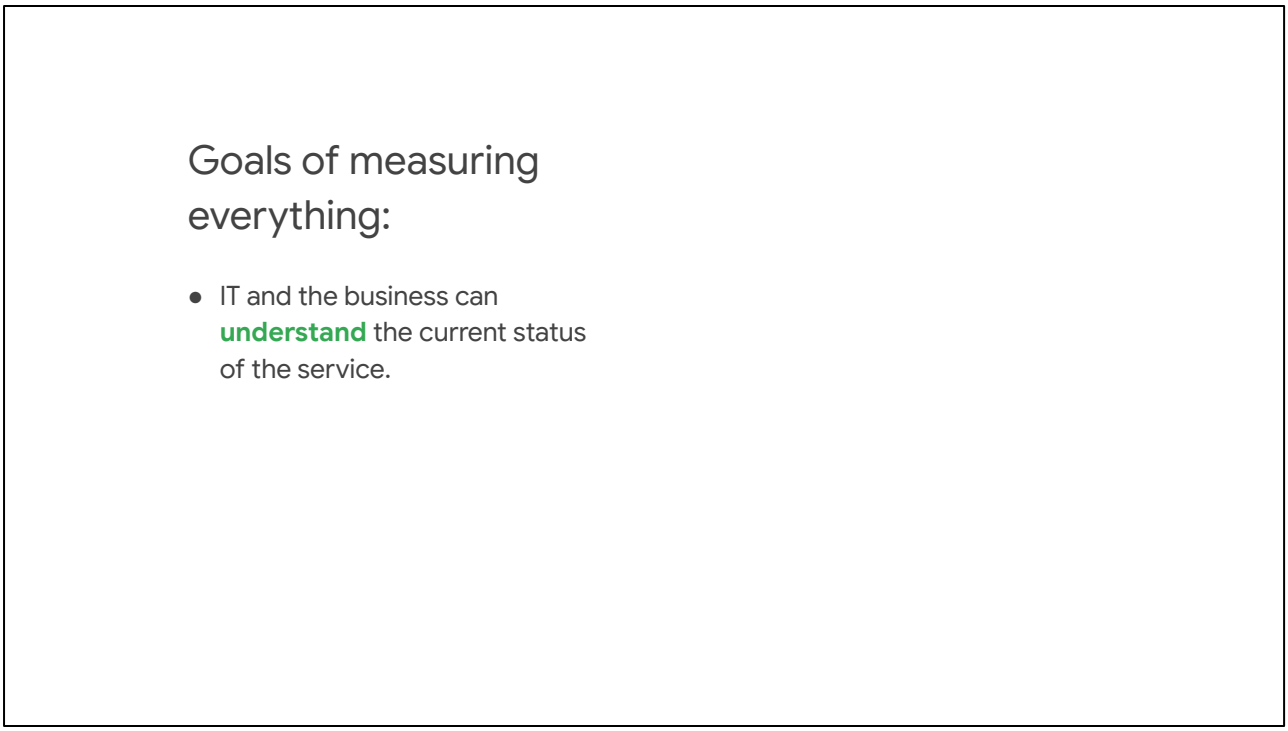
The final step on the journey to SRE is regulating workload.

Learning topics

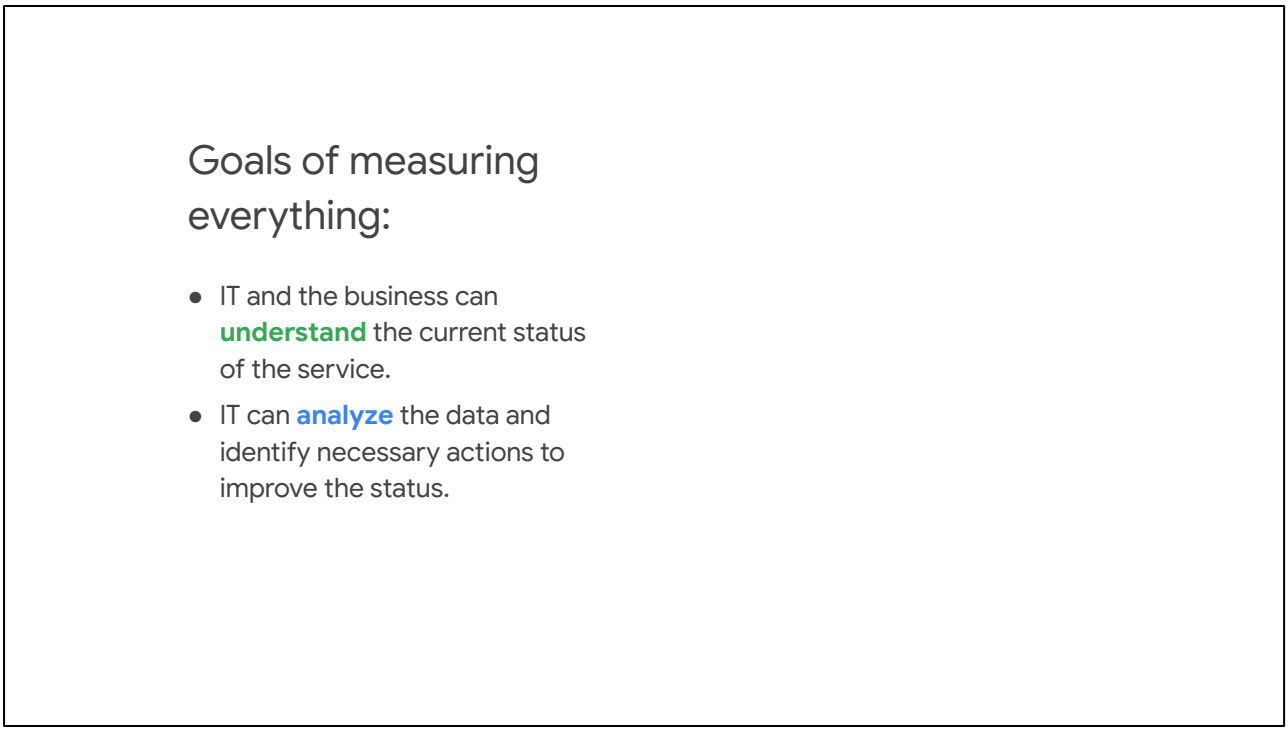
- Measure reliability and toil
- Monitoring
- Goal-setting, transparency, and data-driven decision making

In this module, you'll learn about the SRE practices related to measuring everything, specifically reliability and toil, and the concept of monitoring. We'll also cover the cultural fundamentals of goal-setting, transparency, and making data-driven decisions. As we discuss the cultural concepts, you'll start to see some overlaps with other DevOps pillars. This module will connect together what you've learned, as measurement truly applies to all the SRE practices discussed in this course.

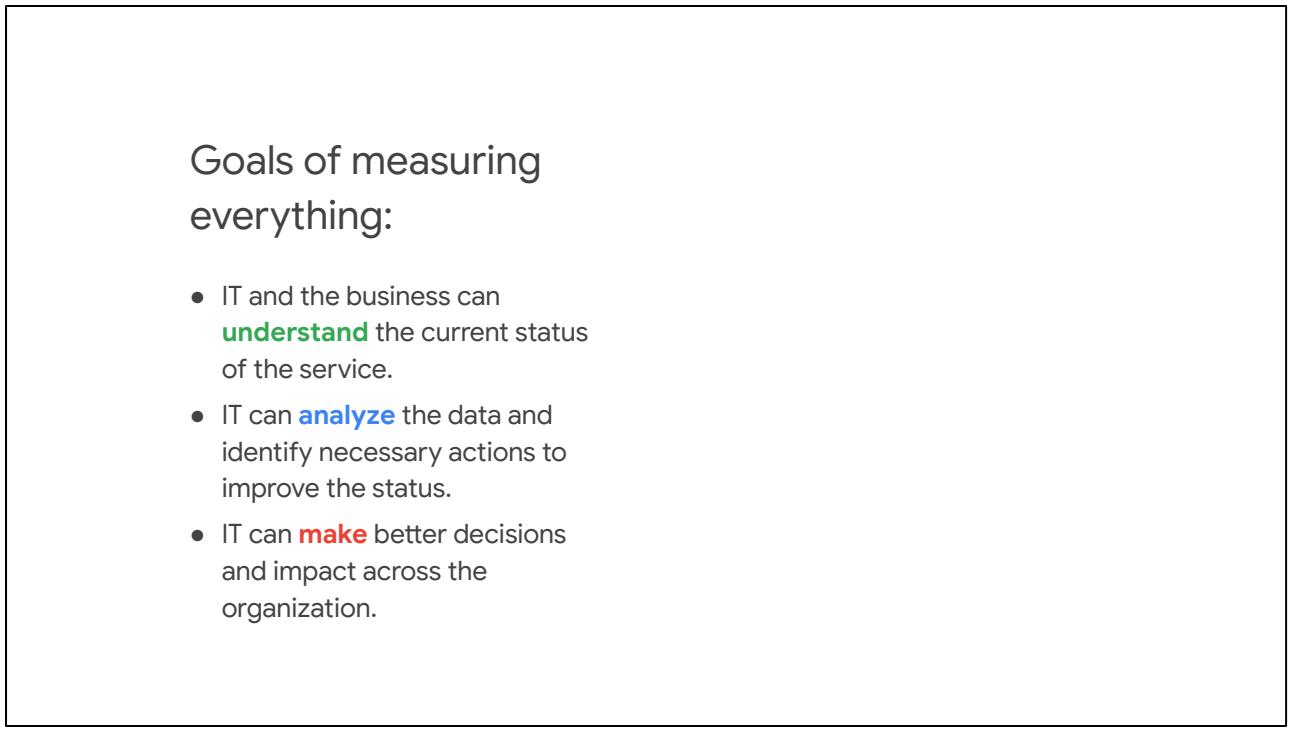
The last pillar of DevOps philosophy we will look at is Measuring Everything. Measurement helps you clearly see what's happening with your services. At Google, we believe there are three main goals of measuring everything.



First, the IT team and the business can **understand the current status of the service objectively**. You've already learned how you can measure reliability with SLIs and SLOs.



Second, the team can **analyze the data and identify necessary actions to improve the status.**



And third, the IT team can collaborate with the business to **start making better decisions and impact** across the broader organization.

Measuring everything with these goals in mind makes your business data-driven and ultimately helps you make better decisions based on the operational data you're collecting.

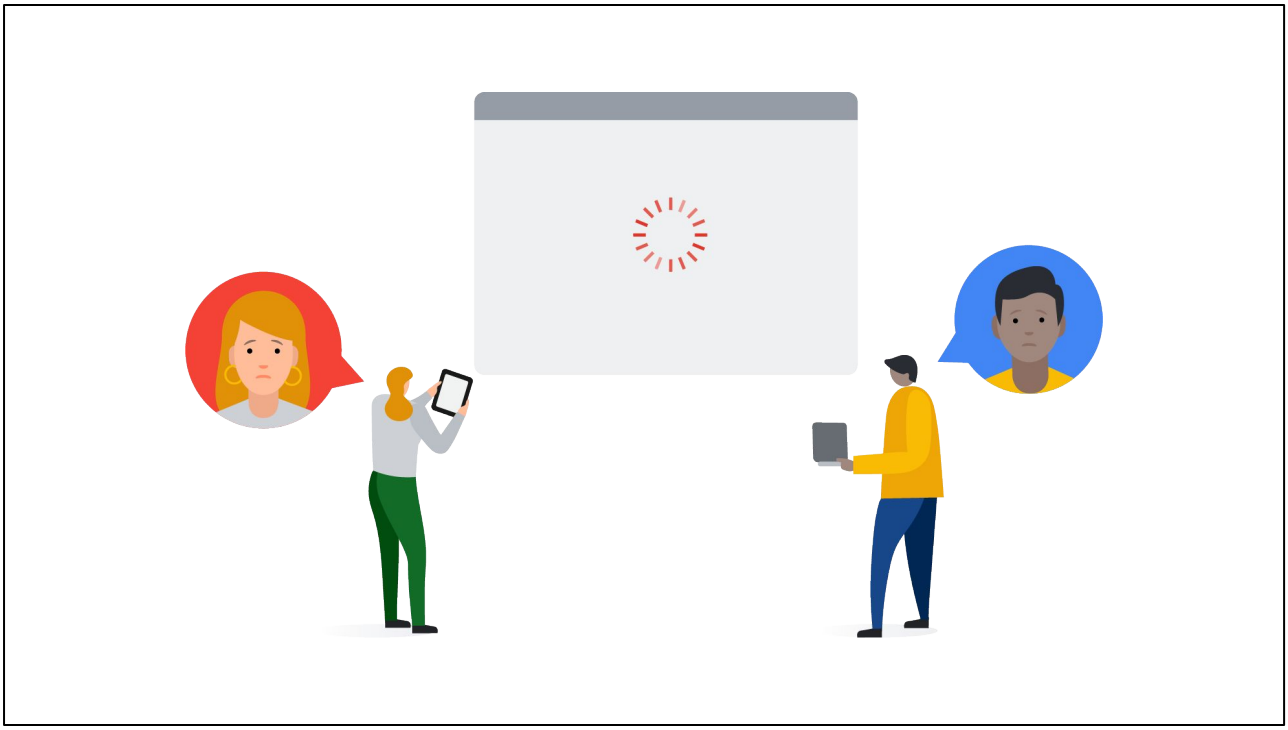
Measure everything

1. Measuring reliability
2. Measuring toil
3. Monitoring

You can’t improve what you don’t measure. In Site Reliability Engineering, there are three core practices that align to this pillar: **measuring reliability, measuring toil,** and **monitoring**.

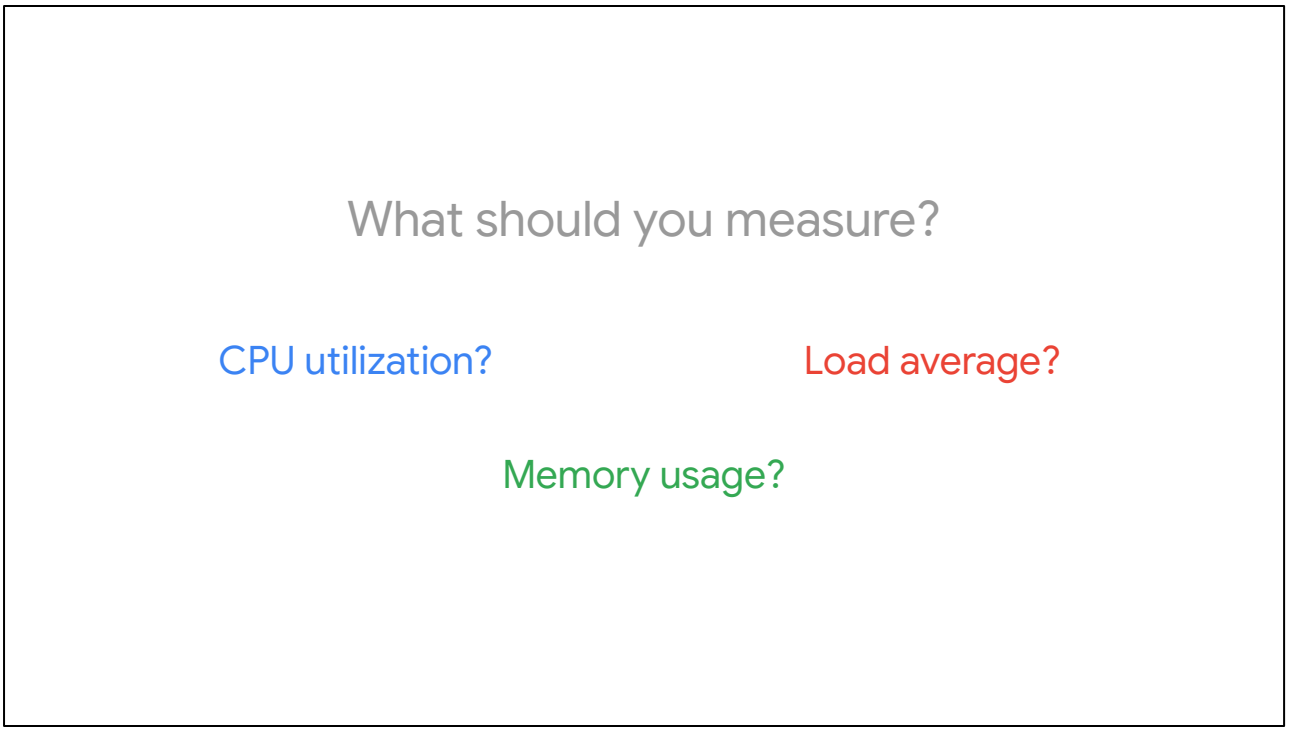
Let's start with **reliability.** In an earlier module, you learned about quantifying reliability with **error budgets, SLIs,** and **SLOs.**

*[To recap, an **error budget** is what you deem an acceptable level of unreliability that you can allocate other engineering work to. Product and engineering teams decide what the reliability target is together. Engineering teams inform the target, while product management defines it. This is not really a technical decision, but instead is defined by customer expectations, the competitive landscape, and the position you have in the market. An **SLI** is a quantifiable measure of a single aspect of service reliability that ideally has a close linear relationship with your users' experience with your service. An **SLO** combines an SLI with a target reliability—that is, it's the threshold that, if crossed, turns happy customers into unhappy customers.]*



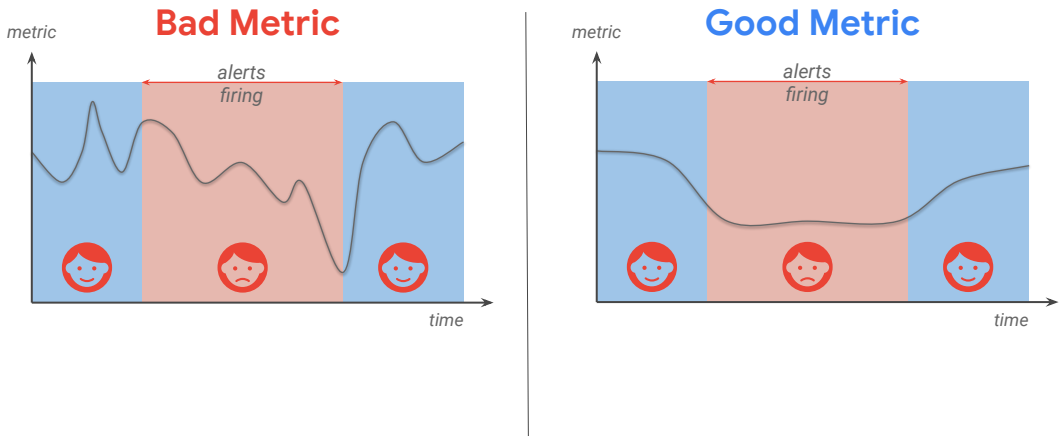
Choosing good SLIs is key to measuring reliability. You are making a connection between your SLIs and your users' experience, so you'll want to decide what to measure based on their perspective.

For example, it doesn't matter to a user whether your database is down or if your load balancers are sending requests to bad backends. They experience a slowly loading webpage, and that makes them unhappy. If you can quantify "slowness," you can then tell how unhappy your users are in aggregate, which lets you define your SLO.

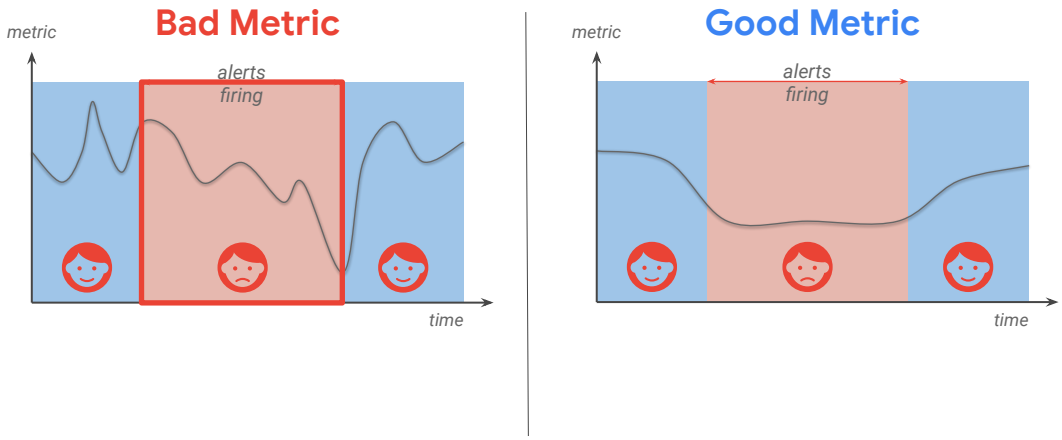


So what should you measure? CPU utilization? Memory usage? Load average?

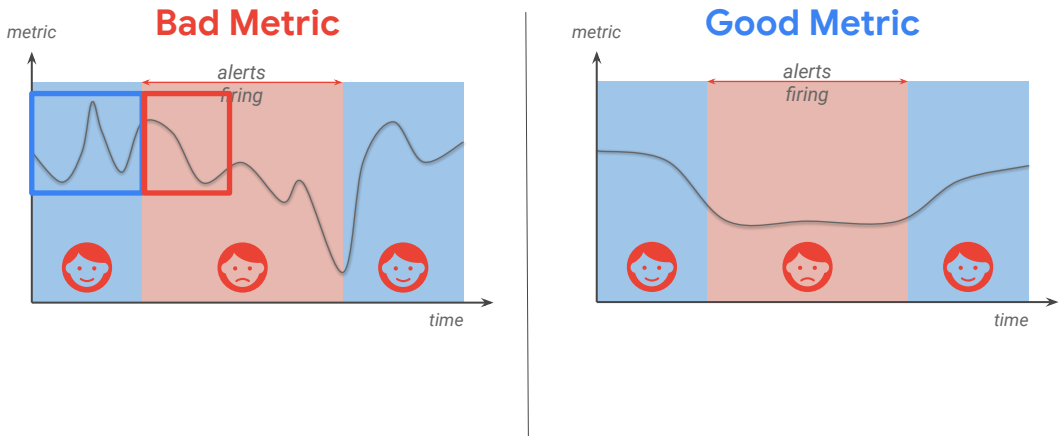
Well, as we just mentioned, these metrics don’t necessarily reflect whether your users are happy.



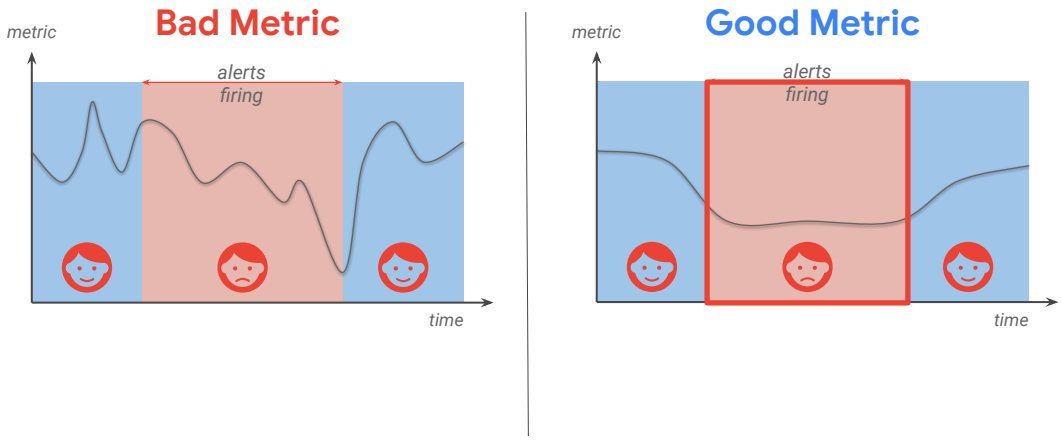
To illustrate the difference, let's look at the signal from these two metrics for the same "outage" in your service. If you assume that you have some way of knowing your users were "too unhappy" with your service, and that this time period is represented by the red area on these two graphs, then you can show that the metric on the right is a far more useful representation of user happiness.



Your "bad" metric does show an obvious downward slope during the outage period, but there is a large amount of variance.



The expected range of values seen during normal operation, shown in blue to the left, has a lot of overlap with the expected range of values seen during an outage, shown in red on the right. This makes the metric a poor indicator of user happiness.



On the other hand, your "good" metric has a noticeable dip that matches closely to the outage period. During normal operation, it has a narrow range of values that are quite different from the narrow range of values observed during an outage. The stability of the signal makes overall trends more visible and meaningful. This makes the metric a much better indicator of user happiness.

SLIs need to provide a clear definition of good and bad events, and a metric with a lot of variance and poor correlation with user experience is much harder to set a meaningful threshold for. Because the "bad" metric has a large overlap in the range of values, our choices are to set a tight threshold and run the risk of false-positives, or to set a loose threshold and risk false-negatives. Choosing the middle ground also means accepting both risks.

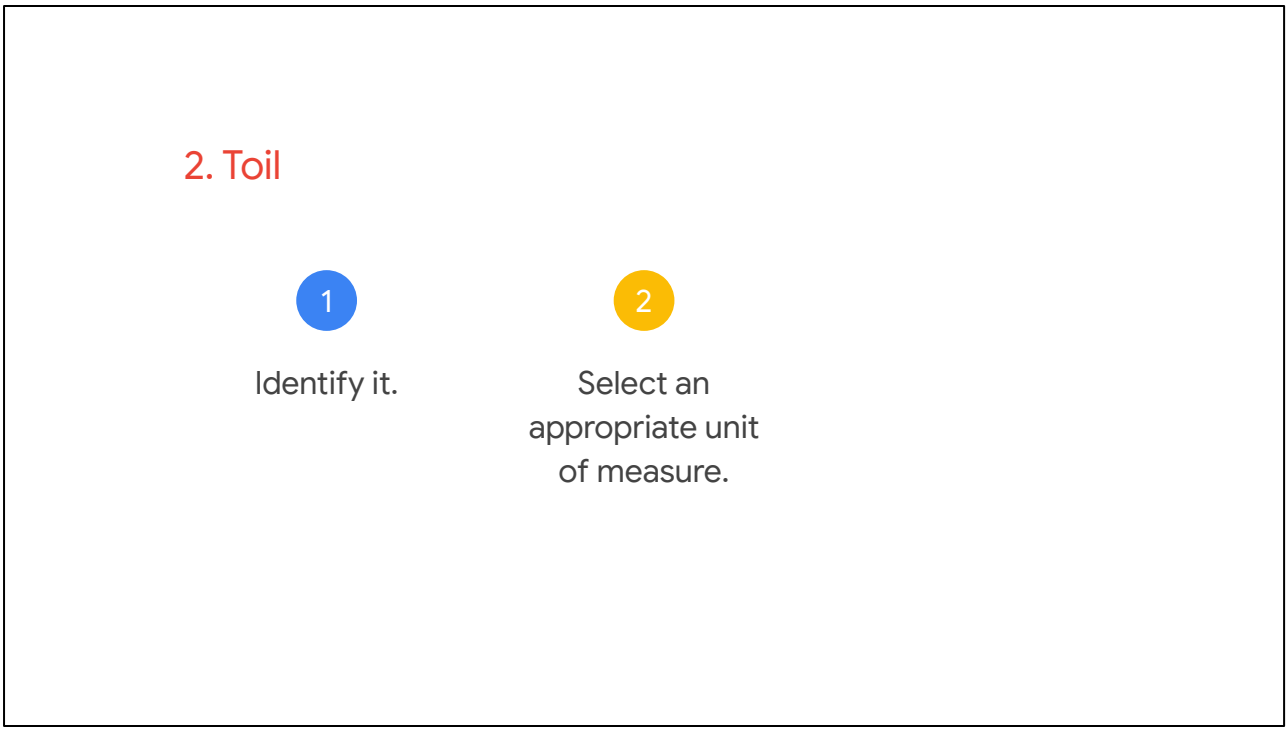
The "good" metric is much easier to set a threshold for because there is no overlap at all. The biggest risk you have to contend with is that perhaps the SLI doesn't recover after the outage as quickly as you might have hoped for.

2. Toil

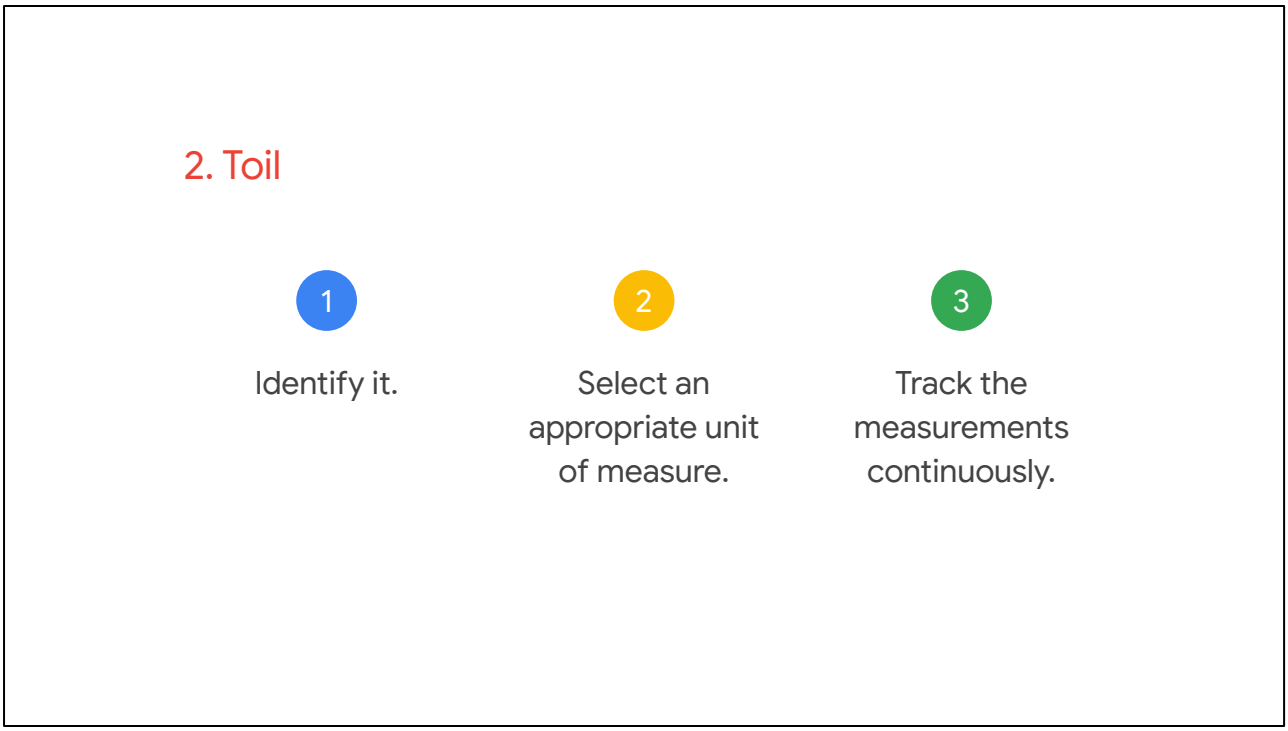
The second aspect to measure is **toil.** As you've learned, toil is work that is directly tied to running a service and that is manual, repetitive, automatable, tactical, and without enduring value.

You can measure toil in three steps.

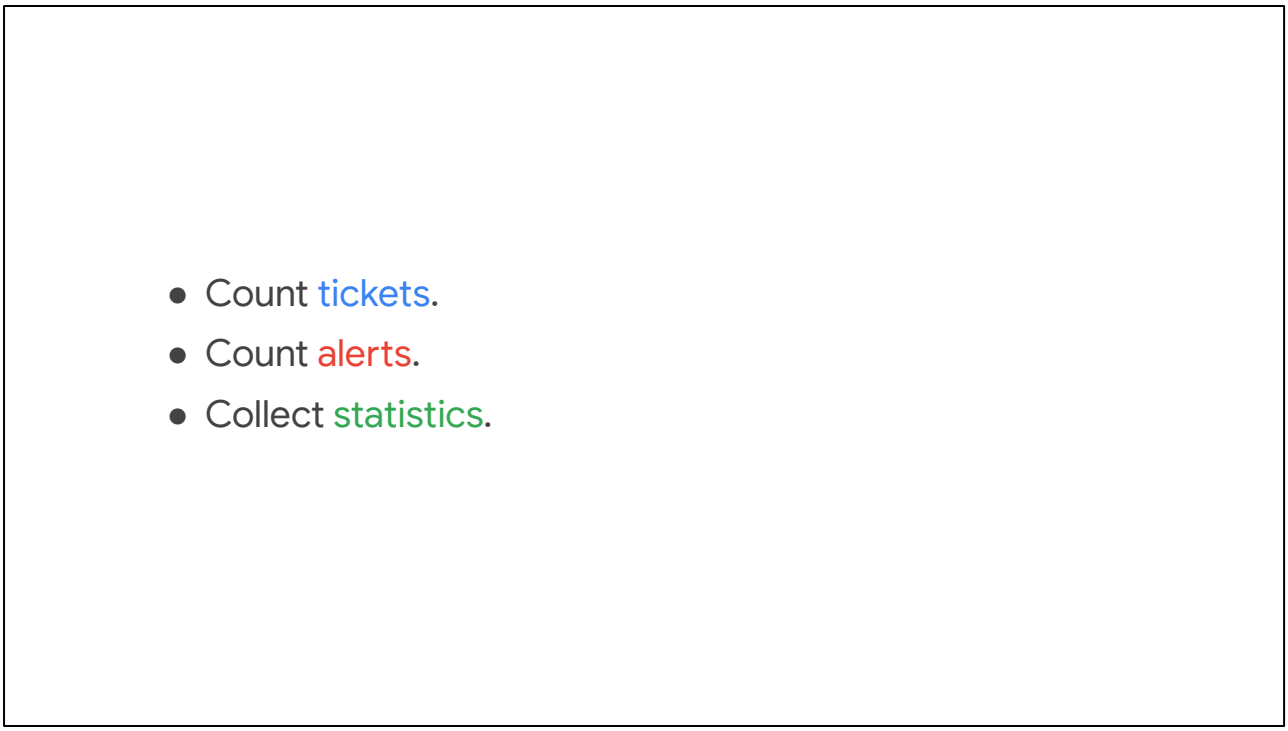
First, **identify it.** Who is best positioned to identify toil depends on your organization. Ideally, these people are stakeholders and those who perform the actual work.



Next, **select an appropriate unit of measure**. This unit needs to express the amount of human effort applied to this toil. Minutes and hours are a good choice because they are objective and universally understood.

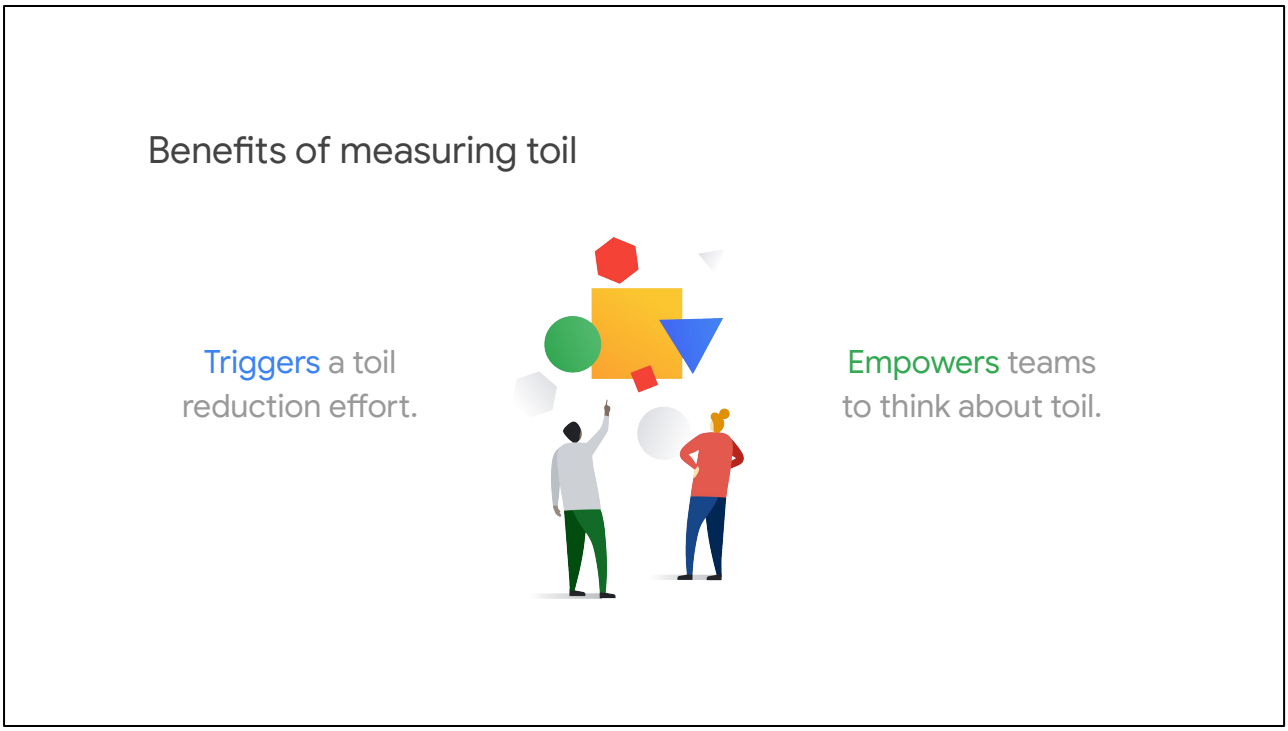


And third, **track the measurements continuously**. Do this before, during, and after toil reduction efforts. Streamline the measurement process using tools or scripts so that collecting these measurements doesn't create additional toil.

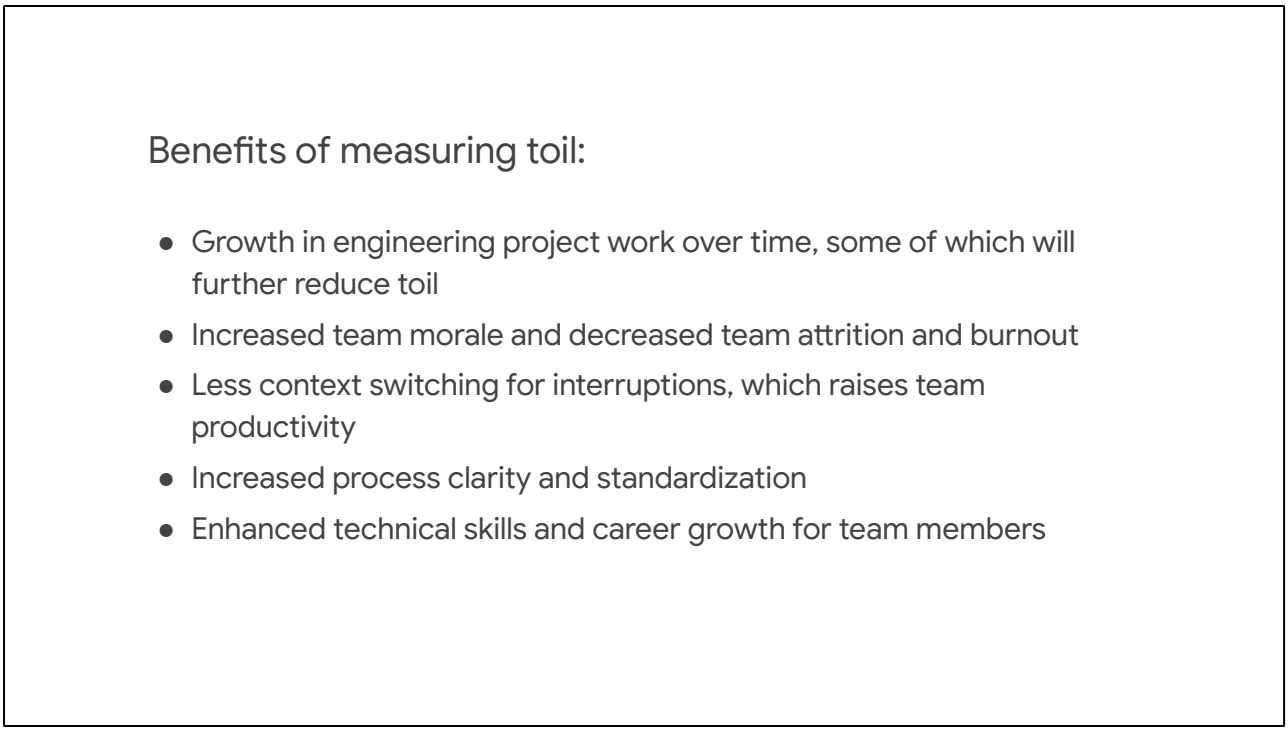


Start simple. **Count the number of tickets you receive. Count the number of alerts. Collect alerts stats on cause and action**. These can prove useful in identifying the source of toil. You can also measure actual human time spent on toil by collecting data, either in the ticketing system directly or by asking your team to estimate the time spent on toil every day or week.

There are clear benefits of measuring reliability of your services, but you may be wondering what the benefits of measuring toil are.



First, it **triggers a reduction effort**. Identifying and quantifying toil can lead to eliminating it at the source. And second, it **empowers your teams to think about toil**. A toil-laden team should make data-driven decisions about how best to spend their time and engineering efforts.



Additional benefits include:

- Growth in engineering project work over time, some of which will further reduce toil
- Increased team morale and decreased team attrition and burnout
- Less context switching for interruptions, which raises team productivity
- Increased process clarity and standardization
- Enhanced technical skills and career growth for team members

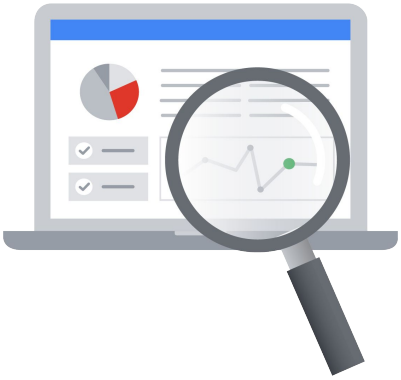
Benefits of measuring toil:

- Reduced training time
- Fewer outages attributable to human errors
- Improved security
- Shorter response times for user requests

- Reduced training time
- Fewer outages attributable to human errors
- Improved security
- Shorter response times for user requests

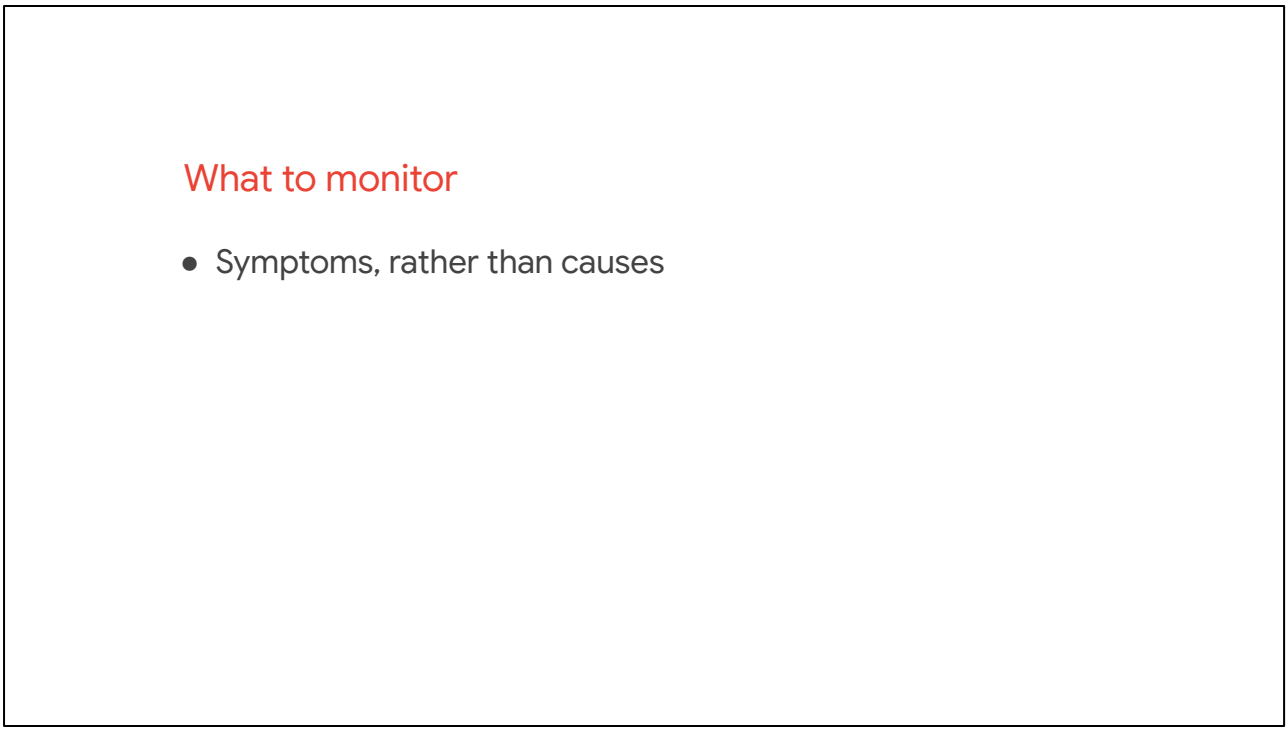
3. Monitoring

Allows visibility into a system.



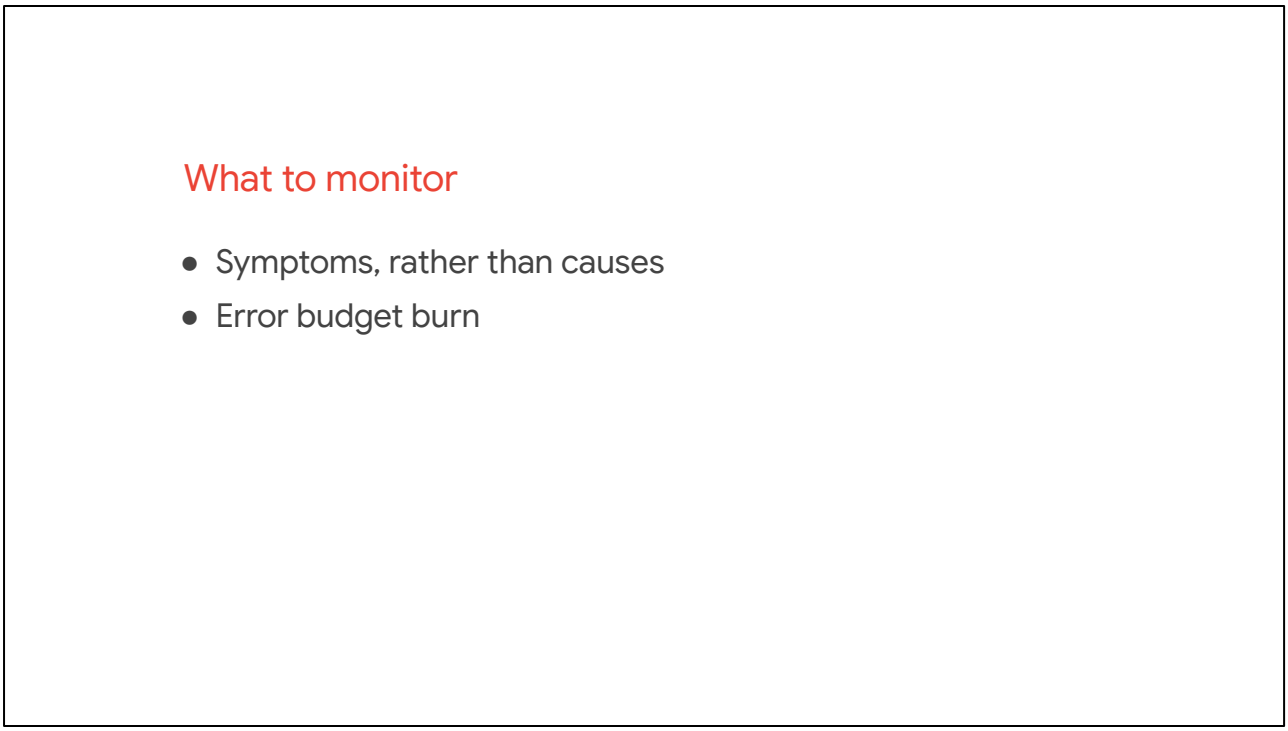
Finally, measuring everything involves **monitoring**. Monitoring **allows you to gain visibility into a system**, which is a core requirement for judging service health and diagnosing your service when things go wrong.

Let’s first look at what you should monitor.



It’s best practice ~~is~~ to **alert on symptoms rather than causes**. Users don't care whether they can’t get to your website because your router is rebooting or because the database is overloaded. Similarly, they don't care if CPU utilization is very high if they still can access the system and it feels fast.

Creating a separate alert for each cause typically results in a lot of spam alerts. It's better to have fewer symptom-based alerts combined with good debugging tools, like dashboards, that will allow responders to more easily identify a specific cause.



Ideally, Google recommends **alerting based on error budget burn**. You might page someone when you are consuming your error budget very quickly, for example when you've spent ten hours worth of error budget within one hour. You might just create a ticket for a lower burn rate, for example if you've spent three days worth of budget within three days.

Basically, only escalate to a human if you risk meeting your SLO for the month. If your SLO is set correctly, you won't be paging people for problems that users don't care about.

What to monitor

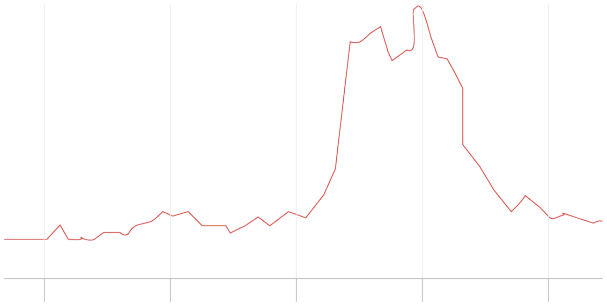
- Symptoms, rather than causes
- Error budget burn

Capacity alerts are an exception.

There are exceptions to this rule, of course, because there might be a problem that does not result in user-visible symptoms. Capacity alerts can be an example of this: if you know you will exhaust a particular limit soon, it warrants an alert even if there is no user-visible symptom yet.

Four golden signals

1. Latency
2. Traffic
3. Errors
4. Saturation



To simplify things, Google recommends monitoring for the four golden signals:

1. Latency
2. Traffic
3. Errors
4. Saturation

You can see how measurement and monitoring are important SRE practices and allow your development teams to focus on business critical work. In the next video, we'll discuss the cultural concepts of goal setting, transparency, and data-based decision making, and how supporting these practices is important for measurement.

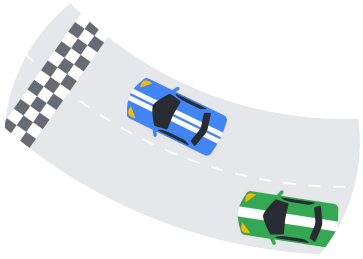
As we discussed, an important SRE practice is to measure everything, specifically reliability and toil. However, in order to measure everything, you need to ensure that you have a culture of goal-setting, transparency, and data-driven decision-making in your organization.

So how can you achieve this?

1. Goal-setting

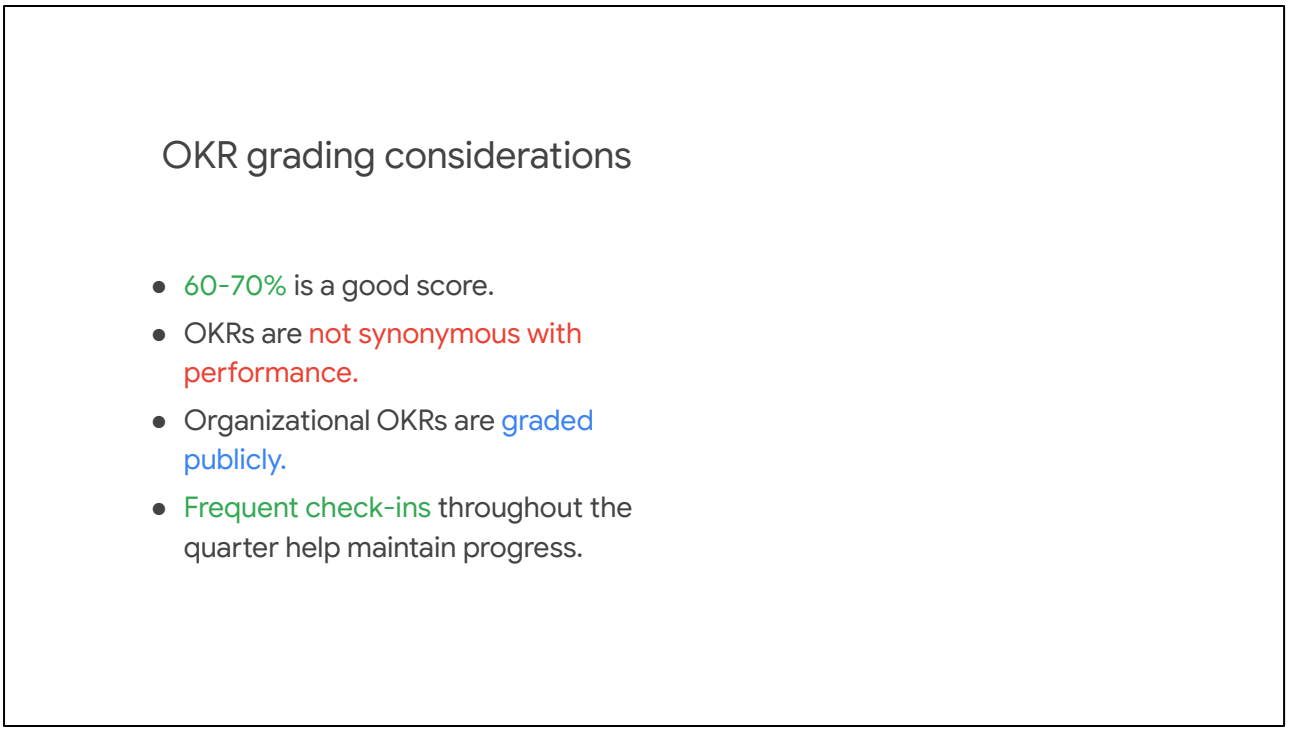
Let’s start with **goal-setting**. For this, you’ll want to create a data-driven goal setting process. You should look at KPIs—for who and for what you are measuring—and an approach—what to measure and how.

1. Goal-setting



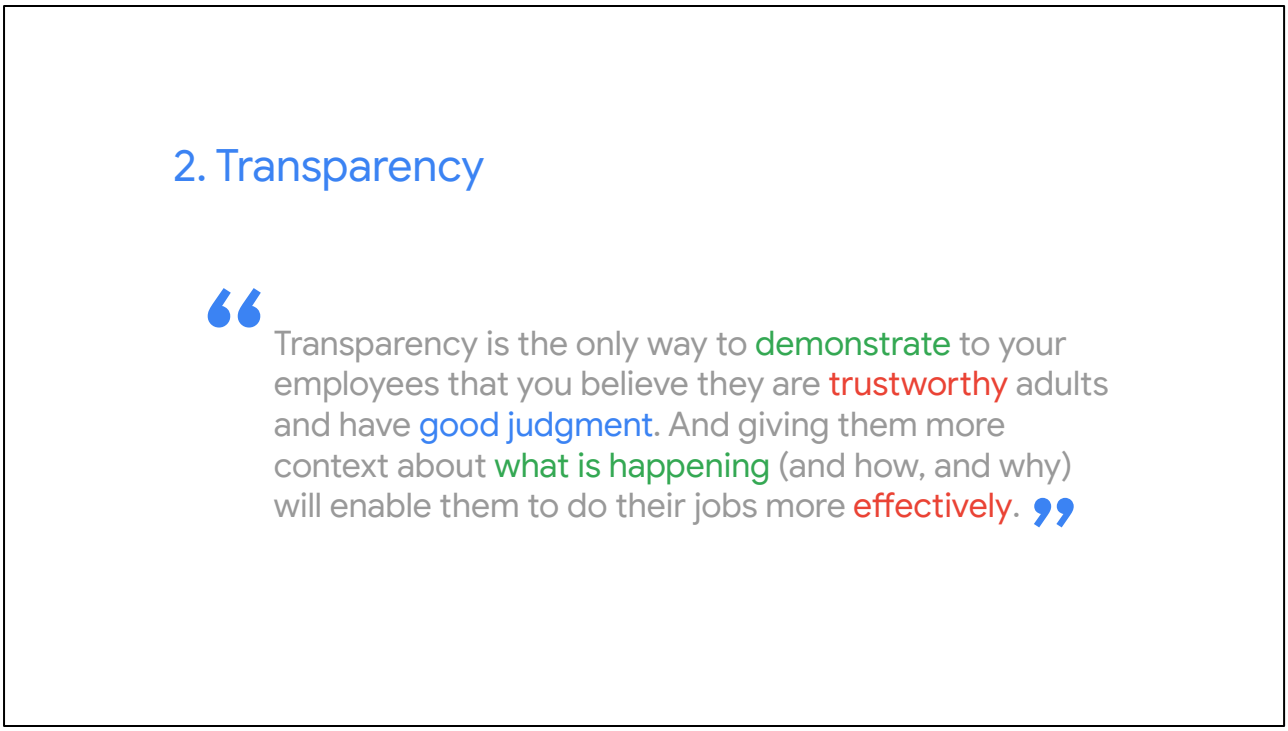
Google uses OKRs as KPIs,
graded from 0.0 to 1.0

As you learned in an earlier module, Google uses **OKRs**, objectives and key results, as **KPIs**. OKRs are usually graded on a scale of 0.0 to 1.0, where 1.0 indicates a fully achieved objective.



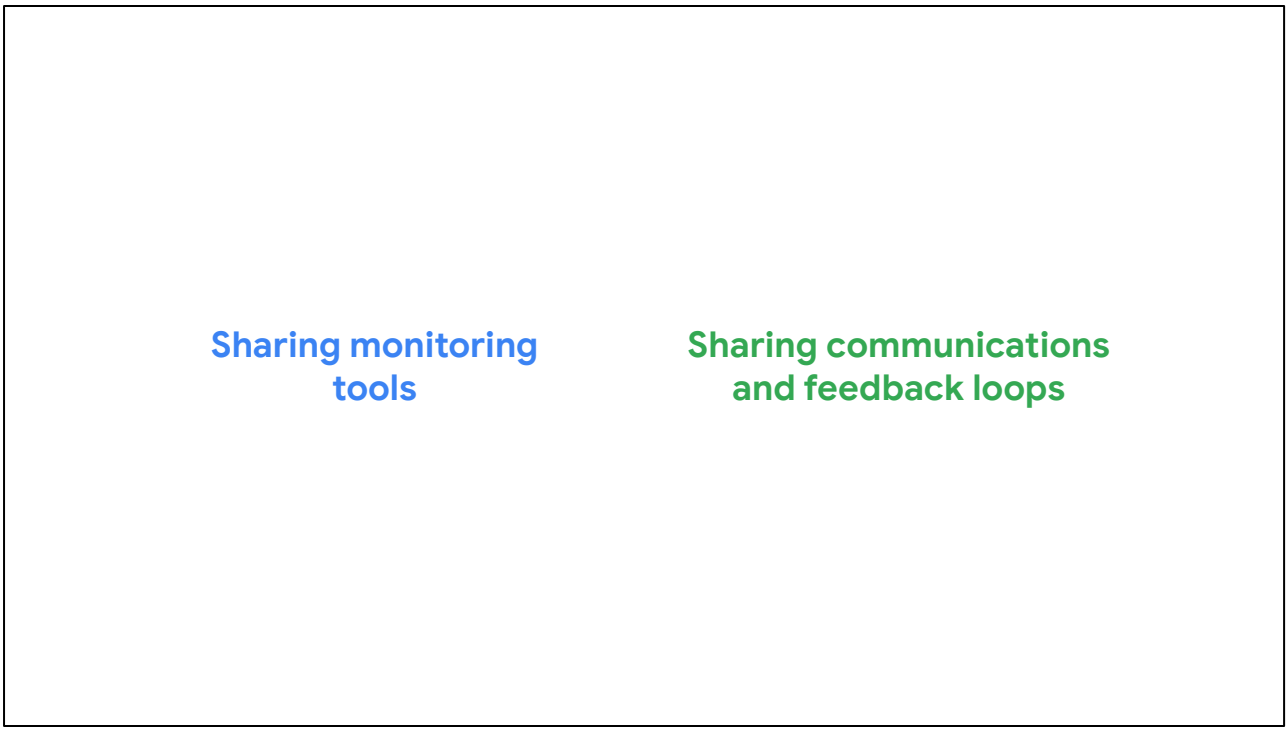
Consider these things when grading OKRs:

- The optimal point for an **OKR grade is 60-70%**. Think big when developing your OKRs!
- OKRs are **not synonymous with performance evaluation**. Instead, they show individuals contributions and impact.
- **Organizational OKRs are graded publicly** so everyone can see their progress.
- **Frequent check-ins throughout the quarter** help teams and individuals maintain progress.

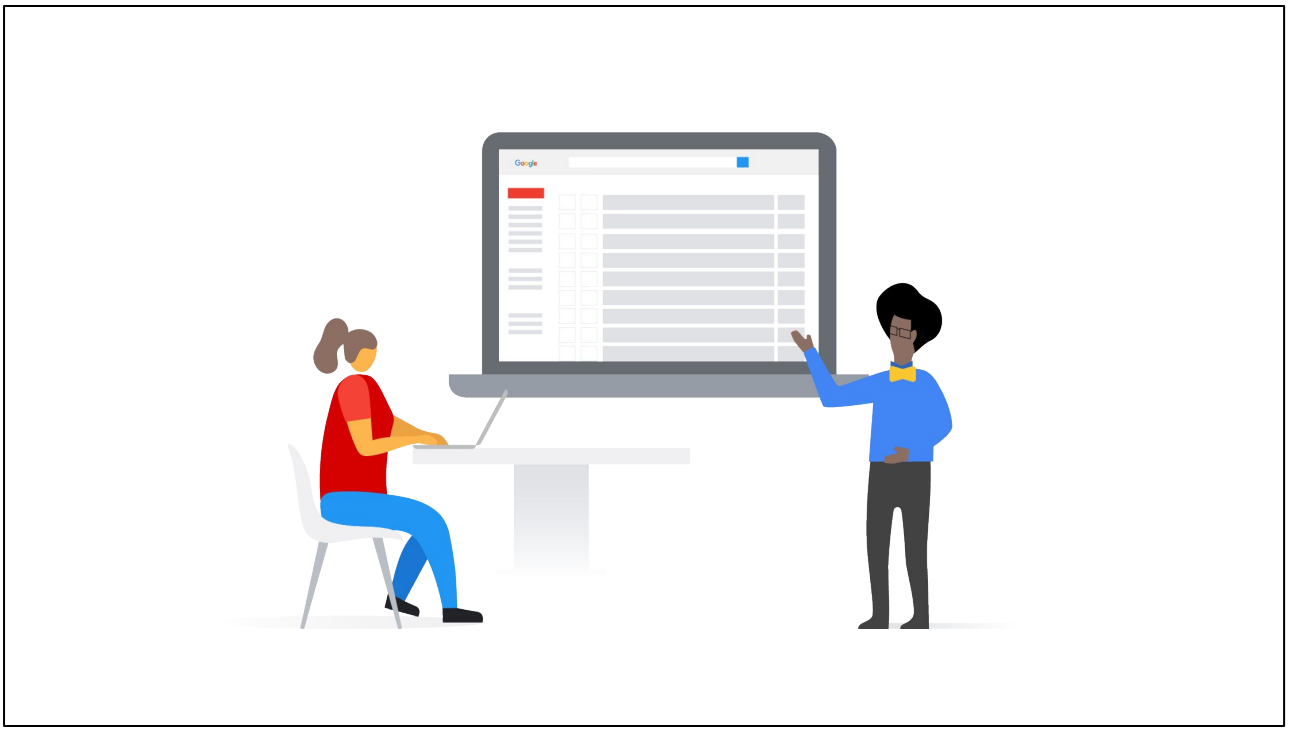


Next, you want your organization to embrace **transparency**.

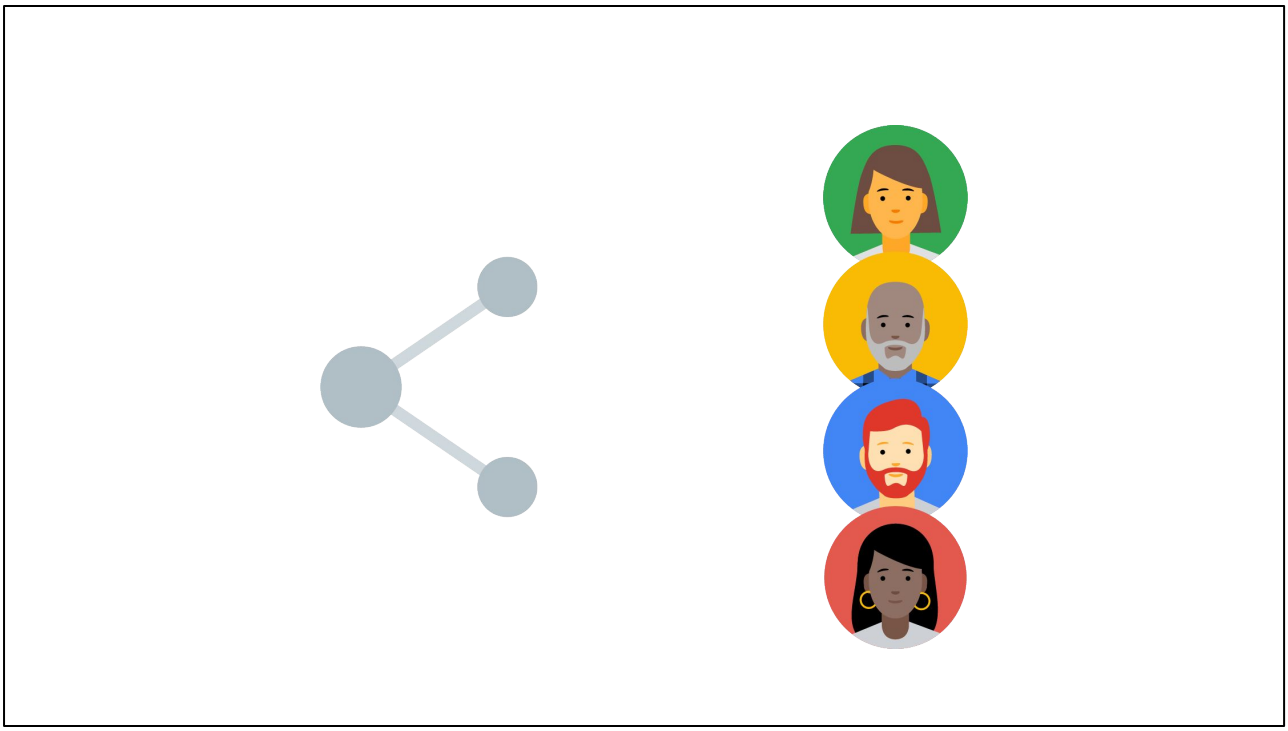
Lazlo Bock, the former SVP of People Operations at Google, once said that transparency “is the only way to demonstrate to your employees that you believe they are trustworthy adults and have good judgment. And giving them more context about what is happening (and how, and why) will enable them to do their jobs more effectively.”



In terms of SRE practice and culture, there are some specific ways of promoting transparency: **sharing monitoring tools** and **sharing communications and feedback loops**.

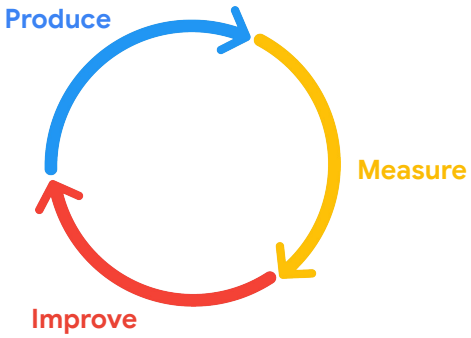


In terms of tools, Google uses an issue tracker called Buganizer. Everyone in the company can access this tool. Specifically, the development and operations teams can review issues and see progress towards their resolutions. Shared tools for development and operations teams promote transparency.

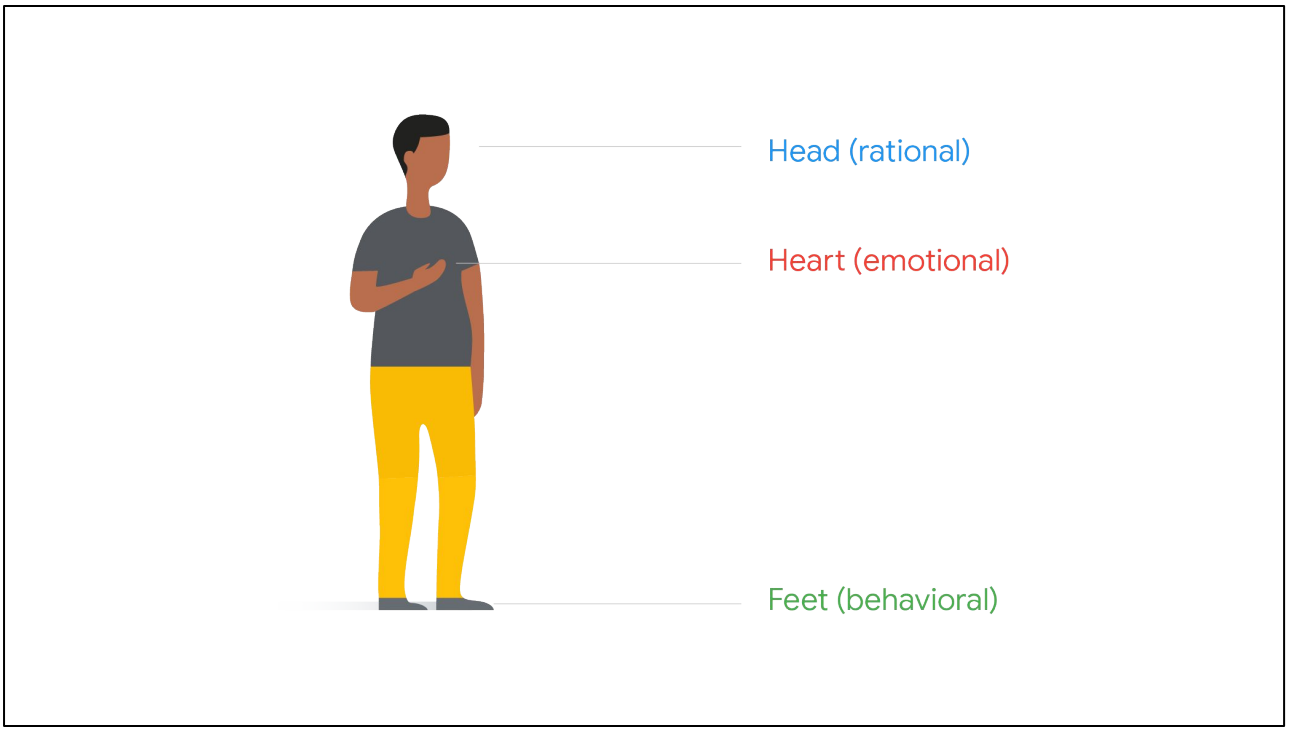


However it’s not just tool-sharing that will help your organization maintain transparency. A **culture of sharing information** needs to come from your systems as well.

Feedback loop



In communication transparency, it's important to remember **feedback loops**. Feedback loops are simple to understand: you **produce** something, **measure** information on the production, and use that information to **improve** production. It is a constant cycle of monitoring and improvement.



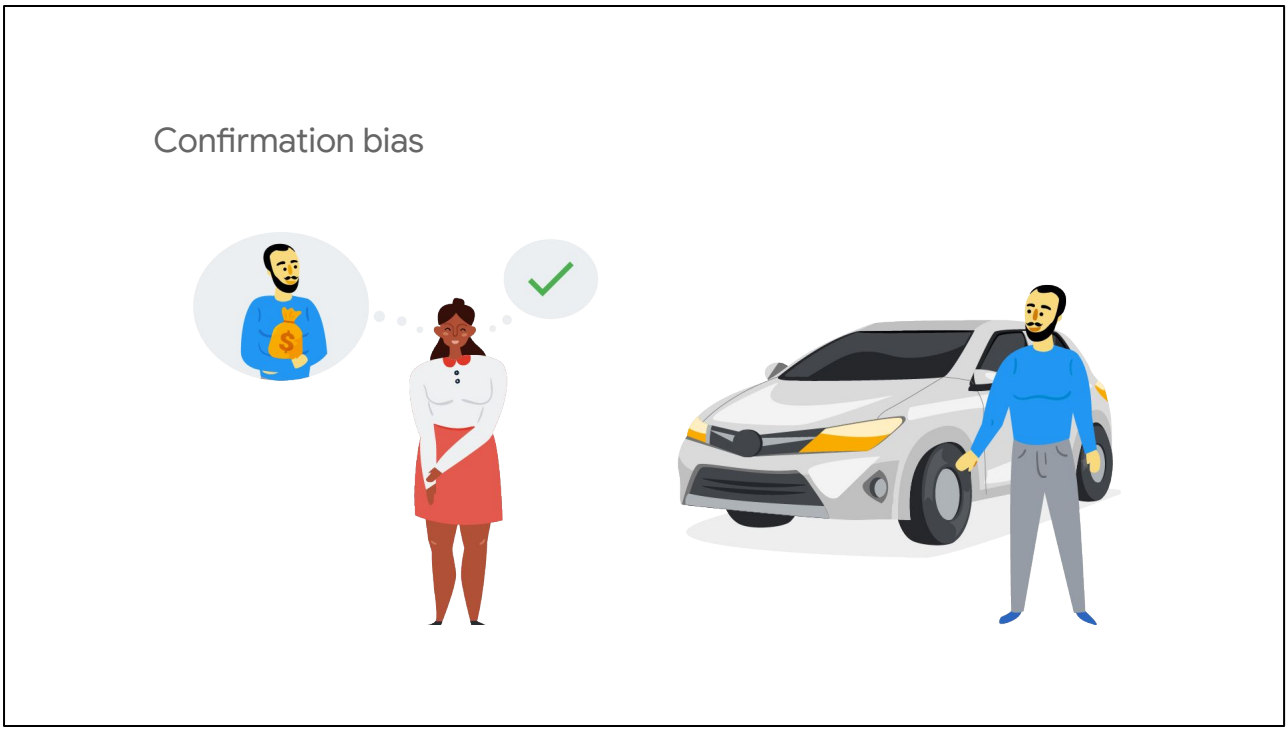
IT leaders need to make it a priority in their organizations. To do this, think about the Head, Heart, Feet model you learned about in the previous lesson about resistance to change.

3. Data-driven decision making

Data-driven decision making is another important aspect of SRE culture. To make truly data-driven decisions, you need to remove any unconscious biases. Unconscious biases are social stereotypes about particular groups that people form without realization. There are several common biases that show up in organizations and influence the ways decisions are made.

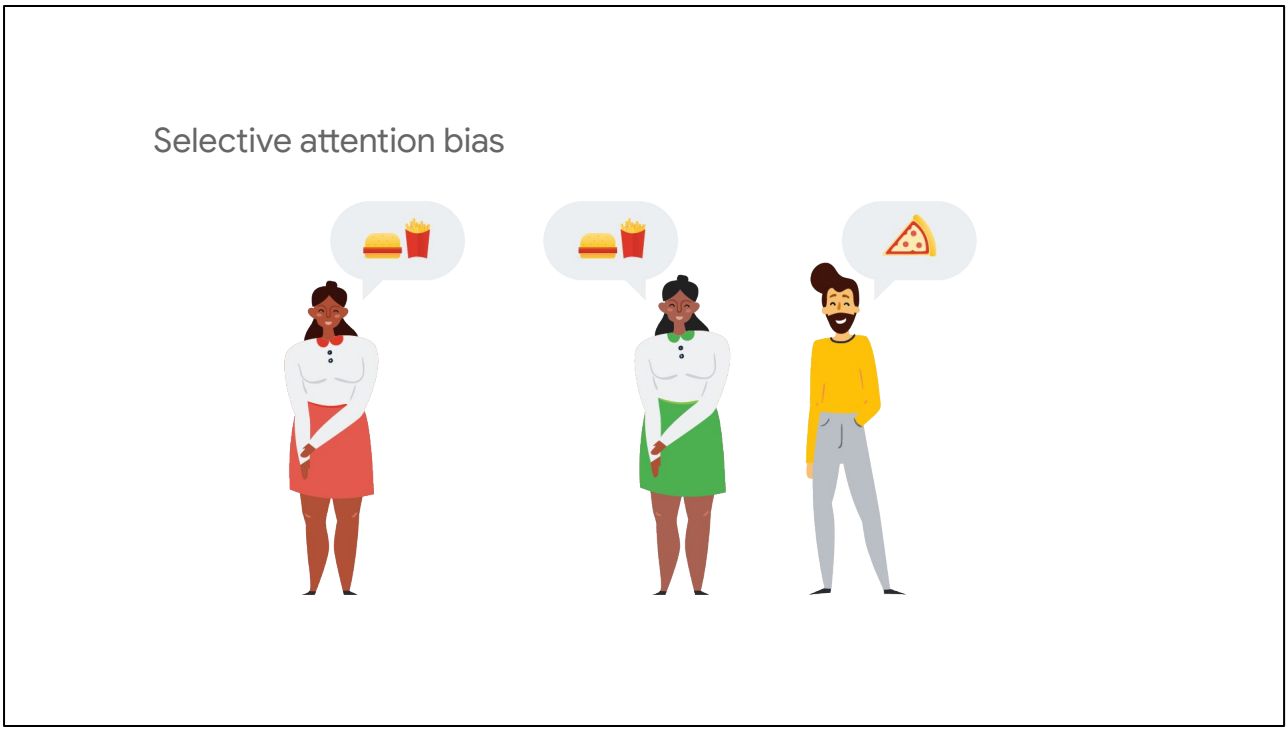
Affinity bias

Affinity Bias. This is a bias toward those who are similar to you. This could mean similar in many different ways, such as race, gender, socioeconomic background, or education level. People tend to gravitate toward those who are like them.



Confirmation Bias. This bias is the tendency to find information, input, or data that supports your preconceived notions.

Labeling Bias. This bias is making opinions based on how people look, dress, or show up externally.



Selective Attention Bias. This bias is when you pay attention to things, ideas, and input from people whom you tend to gravitate toward.

All of these biases can create environments that are not very diverse or inclusive. They can also impact the creativity, innovation, morale, engagement, and turnover in an organization. When employees see decisions being made based on any of these biases, it corrupts their work environment.

So how can you help remove these unconscious biases?

First, **question your first impressions**. Don’t stop with the first decision that comes to your mind, especially when you're determining whom to promote, hire, or add to the team.

Next, **justify decisions**. If you’re held accountable, you’ll be less unconsciously biased. Tell people why you decided what you did. If no one will listen to you, write it down.

Lastly, **make decisions collectively**. Ask people to repeat back what they heard, and keep each other’s unconscious biases in check.

Calling out unconscious bias can be a learning moment. So when you think you see it, speak up! Sometimes you will be wrong, and that's okay.

When you create a culture of data-driven decision making and removing unconscious bias, it makes decision-making easier. Teams can look at the data objectively and decide what should be done, without bias. And even if there is bias, you have empowered people to call it out when they believe it’s influencing decisions.

You've now learned about Google SRE technical and cultural fundamentals, and how they fit into an organization's journey to SRE adoption.

In the next module, we'll discuss how you can apply SRE in your organization by upskilling and hiring team members, understanding various SRE team implementations, and engaging with Google's Professional Services Organization.