Increasing Business Value with Better IT Operations

A guide to Site Reliability Engineering (SRE)

Google Cloud

Contents

Introducing SRE 1

What SRE provides 1

Google and SRE 2

A closer look at SRE 3

Balancing reliability with new features 3

SLOs and error budgets 5

Using SLOs with Google Cloud 7

Reducing toil 8

Identifying toil 8

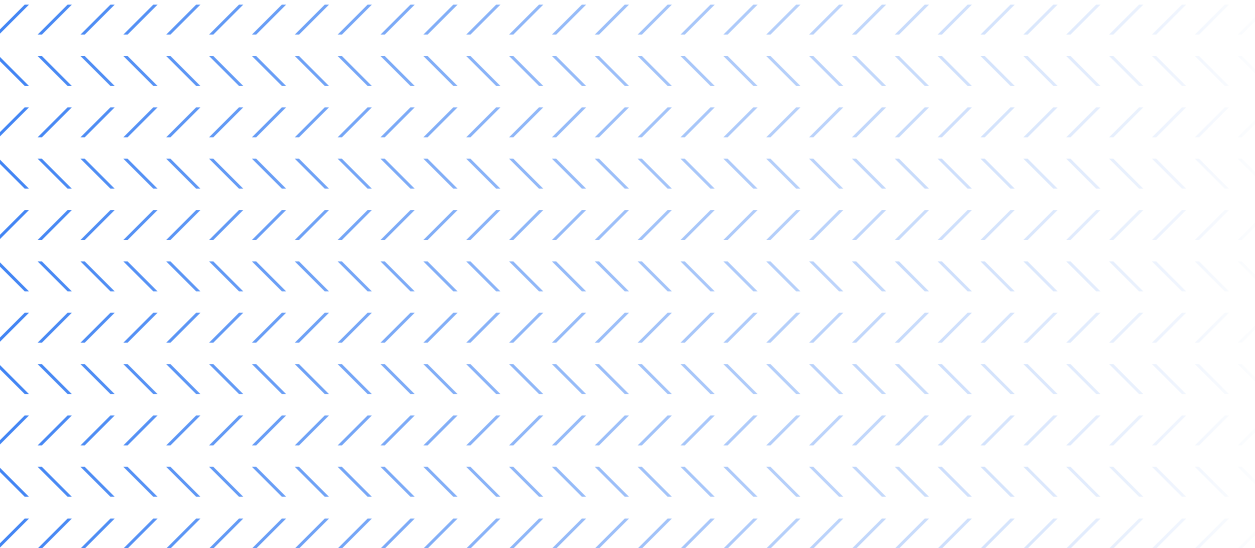
Reducing toil with Google Cloud 9

Other aspects of SRE 10

Adopting SRE 11

Next steps 13

For more information 13



Introducing SRE

Are you happy with how your organization does IT operations? To answer this, think about questions like these:

- Are your applications as reliable as their users need them to be? Can you effectively balance releasing new features with maintaining reliability?
- Are you able to retain your operations people because their work is interesting, not just repetitive toil?
- Does your operations staff respond to incidents in an organized, effective way?
- Once an incident has been handled, do you carry out an effective postmortem that helps you learn from the incident?
- Can you plan accurately for the future of your applications, determining the computing and human resources they'll need going forward?

If you're like many enterprises, the answer to many (maybe even most) of these questions is "no". The truth is that you're not happy with how you do operations, and you'd like to do better.

What SRE provides

One effective way to improve IT operations is to embrace *Site Reliability Engineering (SRE)*. SRE is a set of principles and practices for running production systems that address all the challenges just listed. It includes:

- Using *service level objectives (SLOs)* and *error budgets* to create clear expectations between developers, operations staff, and the business about how reliable an application must be and how rapidly new featurescan be deployed.
- Identifying *toil*, burdensome manual tasks, then automating or otherwise reducing those tasks.
- Creating an *effective incident management* process with well-defined roles and responsibilities.
- Performing *blameless postmortems* to find and fix underlying problems without finger pointing.
- Doing accurate *capacity planning*, letting you understand the computing and human resources an application will need going forward.

SRE can encompass more than this—this isn't an exhaustive list. And even though SRE addresses many different aspects of operations, you don't have to adopt everything at once. Most organizations start where they currently feel the most pain—improving application reliability, for instance, or freeing up operations staff time by reducing toil—then grow from there. Whatever makes sense for you, adopting SRE can help you be happier with how your organization does IT operations.

Google and SRE

Organizations across the industry use SRE today to improve their IT operations. But the fundamental SRE concepts and techniques were created at Google, growing out of our experience running Google's services. We invented SRE, and we've been doing it longer than anybody else. In fact, all of Google Cloud's production services are backed by Google's SRE staff. We believe this makes us your best partner for implementing SRE.

We also provide built-in SRE support in Google Cloud. For example, Google Cloud's operations suite includes tools for defining and monitoring SLOs and error budgets. If you choose to run your applications on Google Cloud, this explicit support helps you adopt SRE more easily.

Why should you embrace SRE? Because it can improve how you do operations, including meeting your customers' reliability expectations and increasing job satisfaction for your staff. This improvement in turn leads to what you care most about: happier customers.

Why adopt SRE if you've already implemented DevOps?

DevOps is an organizational and cultural movement that aims to increase software delivery velocity, improve service reliability, and build shared ownership among software stakeholders. DevOps and SRE share many of the same principles: increasing collaboration, learning from failures, and a focus on team health and happy customers. SRE provides concrete practices, vocabulary, and concepts that complement those of DevOps.

DevOps and SRE reinforce and enable one another. Use the principles and practices of each to continuously improve performance.

A closer look at SRE

Understanding how SRE can help your organization requires knowing a bit more about the ideas it includes. It also requires getting familiar with SRE terminology. (For example, SRE typically uses the more general term *service* rather than *application*, a convention we'll follow from now on.) This section takes a look at the most important aspects of SRE, along with describing how Google supports them.

Balancing reliability with new features

Suppose your organization has built a custom service that's important to your business. Especially if this service is used by customers, you probably want to keep enhancing it, adding new features as quickly as possible. You also want to make sure the service is as reliable as its users need it to be.

There's an inherent tension here. Developers are incented to add new features—they want to make changes to the service. But operations people often strive to minimize changes to the service, since every change brings the possibility of a problem that lowers reliability. Figure 1 illustrates the situation.

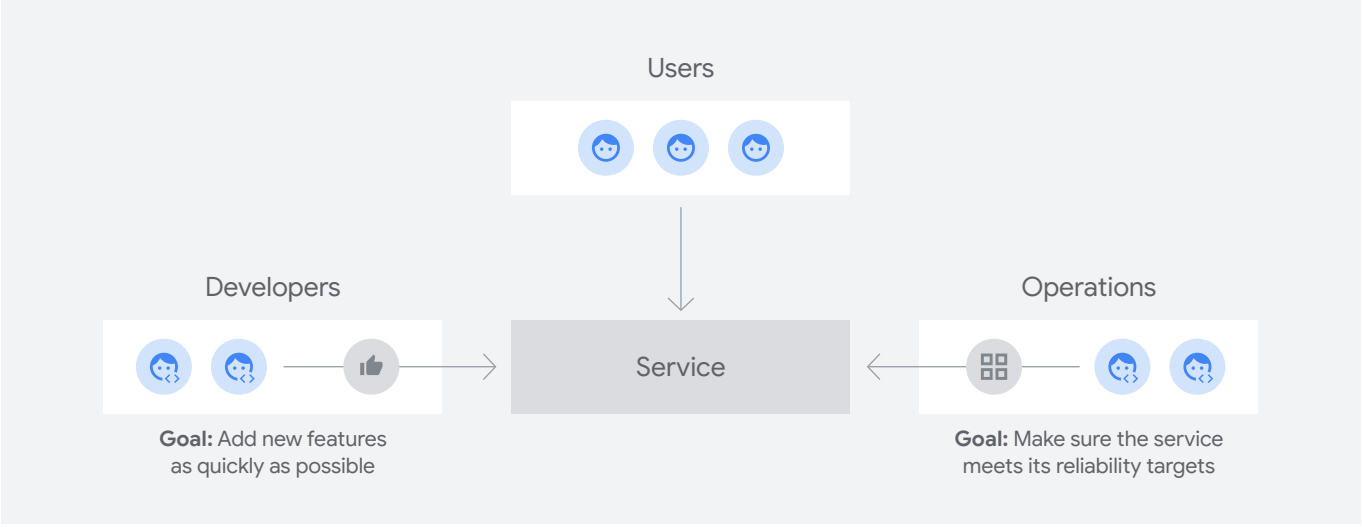


Figure 1: Developers want to add new features to a service, while operations people want to maintain the service's reliability by minimizing changes

What's needed is a clear way to manage this tension, with well-defined targets that both developers and operations can agree on. This is exactly what service level objectives provide. Each SLO defines a target, such as 99.5% uptime, for a service. Critically, SLOs are user-centric: they're based on something the service's user cares about, such as availability or latency.

To measure whether an SLO is being met, each one has associated *service level indicators (SLIs)*. And both SLOs and SLIs can play a role in meeting a *service level agreement (SLA)*. Figure 2 gives a perspective on how SLOs, SLIs, and SLAs fit together.

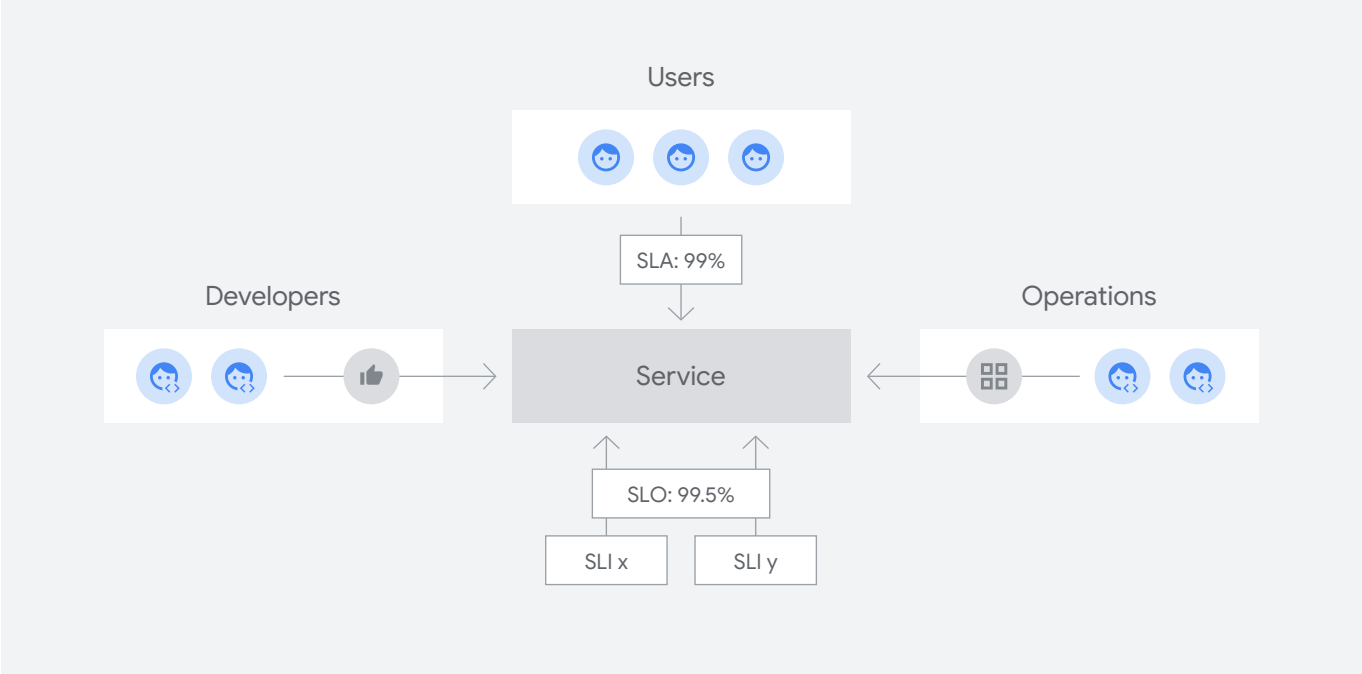


Figure 2: Each SLI measures some aspect of an SLO, while an SLA defines a business agreement with a service's users

Each SLI defines a quantitative measurement of some aspect of what the service provides. Importantly, SLIs should be defined with respect to what the service's users care about. For example, SLI x in Figure 2 might measure the percentage of user requests that return a correct response within 500 milliseconds, while SLI y might measure the percentage of valid login attempts that succeed. Defining an SLI based solely on an internal metric, such as CPU utilization, isn't usually the best approach.

Despite their similar name, SLAs are quite distinct from SLOs and SLIs. An SLA is a business agreement with your service's users, typically providing a monetary penalty if it's not met. SLAs are external, visible to the service's users, while SLOs and SLIs are used within your IT organization.

You should define SLOs that are more stringent than your service's SLA. In the scenario shown in Figure 2, for instance, the SLO is 99.5% while the SLA is only 99%. This lets the SLO act as an early warning. Not meeting an SLO probably means you're in danger of not meeting your SLA, a more painful (and public) event that costs your business money.

SLOs and error budgets

SLOs provide a way for both developers and operations staff to agree on the target reliability of a service. An SLO's percentage also defines a more precise agreement: it determines the number of minutes an application can be down. For example, an availability SLO of 99.5% implies that the service can be unavailable for 0.5% of each month, which works out to around three hours and 40 minutes. This allowable downtime is the service's error budget. Agreeing on an SLO also means agreeing on a precise amount of time the service can fail to meet that SLO.

A service's error budget provides a concrete way to balance the incentives of developers and operations staff while staying focused on the needs of the service's users. Because each new feature release brings risk to the service's reliability, developers are allowed to keep adding new features as long as the service stays within the error budget, i.e., as long as the service continues to meet its SLO. But as bugs in newly released features cause downtime that eats into the error budget, release of new features can slow down or even stop entirely. For example, an organization that normally does twice daily deployments of new features might drop to once-a-day deployment when half of a month's error budget is used up. It might then move to once-a-week deployments when three quarters of the error budget is consumed, then stop deployment entirely for the month when the error budget goes to zero. This process becomes self-regulating: when error budgets expire, developers cannot release more features without working with operations people.

Other things worth knowing about SLOs and error budgets include these:

- Because SLOs are based on something the service's users care about, choosing SLO values is fundamentally a business decision. In other words, the owner of a service must determine the level of reliability that best meets its business requirements. This means that SLOs aren't just a technology measure. They also provide a common language for a service's owner, its developers, and its operations team to think about and agree on the service's reliability needs.
- Even if your organization's leadership wants to achieve 100% reliability, don't expect to reach this goal. Whatever your service does, it depends on other services, and since none of them provide perfect reliability, neither can you. Just as important, increasing reliability means spending increasing amounts of money. The best SLO for a service balances reliability with cost.
- Using SLOs and error budgets requires buy-in from leadership. For example, if a development team uses up its error budget for the month, the operations staff must have the power to stop new deployments to ensure reliability. The only way to get this power in most organizations is to first get your leadership to support the SRE approach.

SLOs and the error budgets they imply let your organization think clearly about the level of reliability—and thus risk—that's right for a service and its users. They're a fundamental part of SRE.

SRE principles and practices can improve both speed of new features and reliability

There's often a trade-off between releasing new features quickly and maintaining reliability. Using SLOs, SLIs, and error budgets can help you manage this trade-off effectively.

But the DevOps Research & Assessment (DORA) State of DevOps report indicates that this isn't always true. According to DORA, "Our research has consistently shown that speed and stability are outcomes that enable each other." In other words, you can improve both at the same time.

An important reason for this is that the need to work together through SLOs gives the operations team a seat at the table with developers early in the design of a new service. This makes SRE a part of the development process, something that helps developers and operations people collaborate more effectively.

Managing the trade-off between speed and reliability is an important part of SRE. Yet surprisingly often, you can still move fast without breaking things.

Using SLOs with Google Cloud

Google Cloud Monitoring provides built-in support for creating and using SLOs, SLIs, and error budgets with services running on Google Cloud. Cloud Monitoring is part of Google Cloud's operations suite, which also includes Cloud Logging, Cloud Tracing, and more.

Using Cloud Monitoring, you can define SLOs for your service. To do this, you first define the SLI this SLO will depend on. This SLI can be based on various metrics, such as the service's availability, how quickly it responds to users, log information such as error rates, and others.

Once you've defined your SLI, you can then create an SLO that uses it. Figure 3 shows an example of how this looks.

This example sets the SLO at 99.5%, indicating that this number should be measured over the course of one month. Once this is done, Cloud Monitoring will automatically compute the error budget the SLO implies. You can then use Cloud Monitoring to check the status of your SLO, as Figure 4 shows. In this example, the SLO is being met, with just under 4% of the error budget remaining.

As these examples show, the ability to observe SLOs, SLIs, and error budgets is built into Cloud Monitoring; it wasn't bolted on later. Google Cloud provides this observability in-context, providing information where you need it.

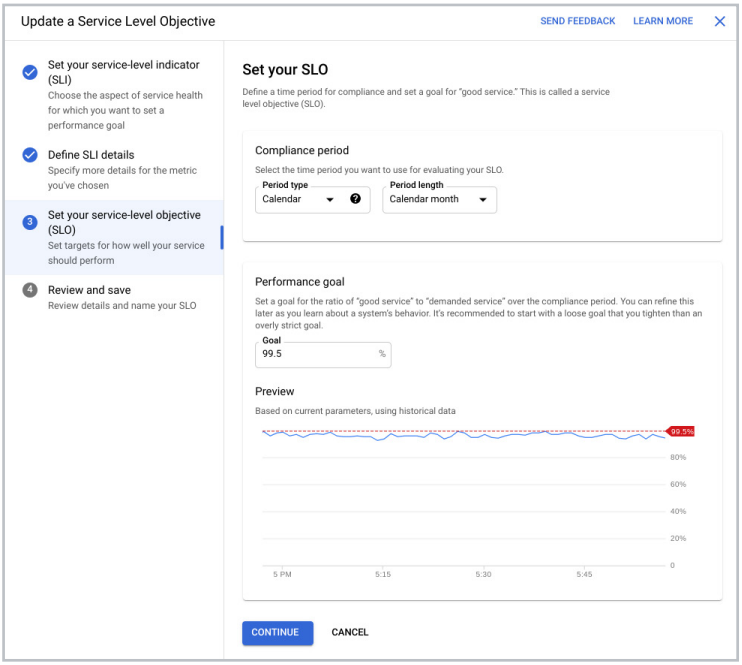


Figure 3: You can define an SLO for a service using Cloud Monitoring

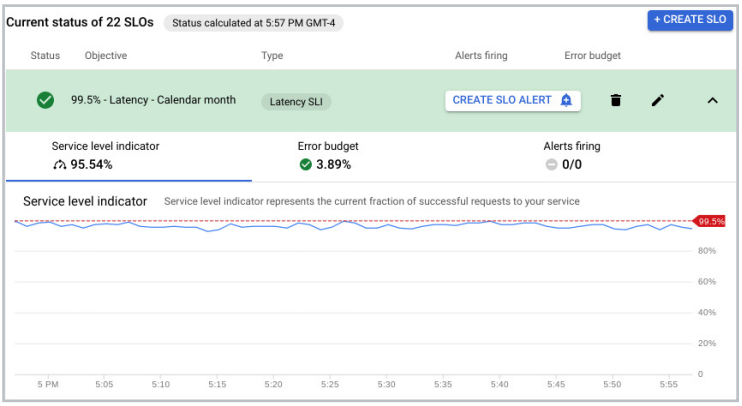
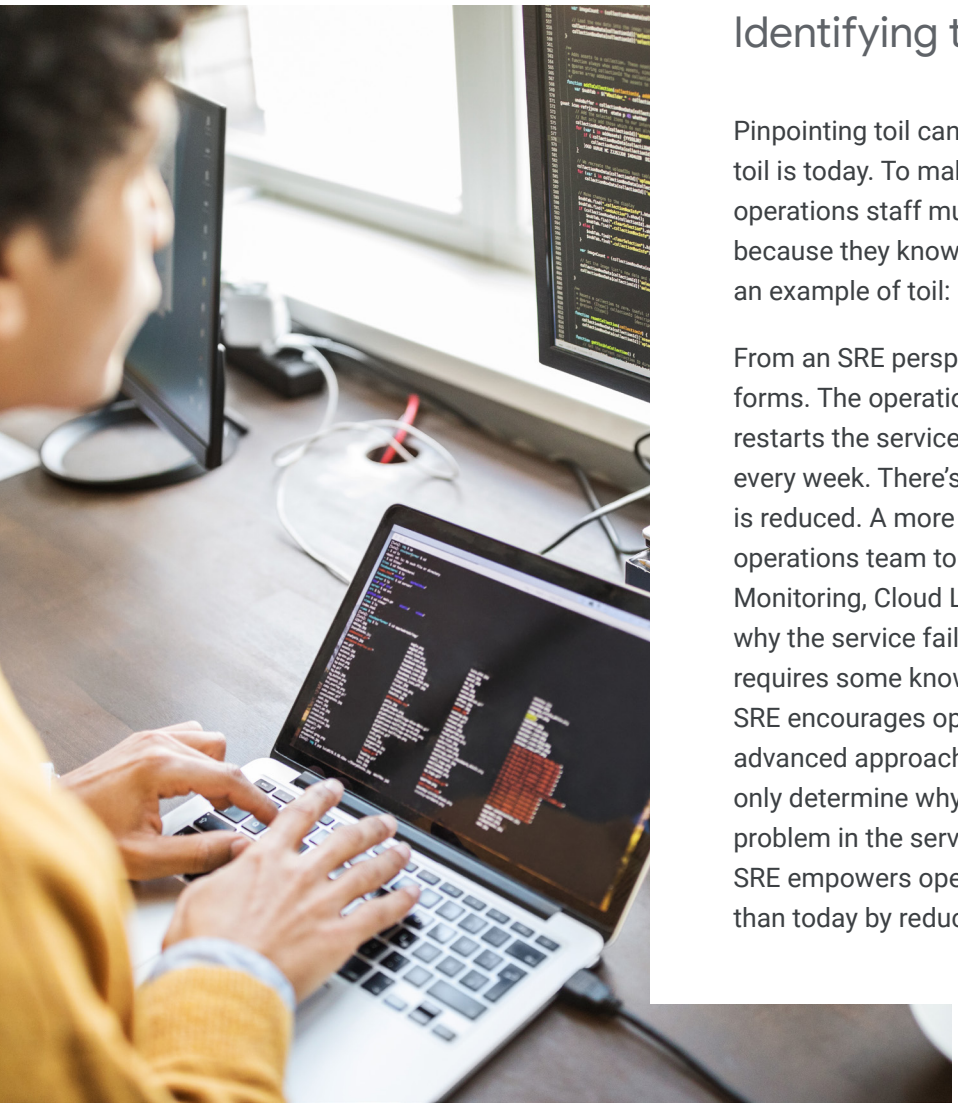


Figure 4: Cloud Monitoring can display an SLO's status, including the percentage of the error budget remaining

Reducing toil

The word *toil* has a special meaning in SRE. It doesn't include everything that operations people might not enjoy, such as attending team meetings. Instead, toil refers to specific types of work needed to run your service. Work that qualifies as toil is manual, repetitive, and usually automatable. Most important, toil is work that does not provide any enduring value.

Toil is a burden on your organization. Most people don't enjoy it, so requiring too much toil can make it harder to retain operations staff. Just as important, toil isn't the best use of people's time. If a task can be automated, why not do it? This lets operations staff devote their time to higher-value work.



Identifying toil

Pinpointing toil can be hard; you likely don't know what your toil is today. To make this idea more concrete, suppose your operations staff must manually restart a service every week because they know it will fail if it runs longer than this. This is an example of toil: manual, repetitive work.

From an SRE perspective, reducing this toil might take several forms. The operations team might first write a script that restarts the service, then set this script to run automatically every week. There's now less repetitive manual labor, so toil is reduced. A more advanced approach would be for the operations team to use observability tools, such as Cloud Monitoring, Cloud Logging, and Cloud Tracing, to understand why the service fails if it's not restarted weekly. Doing this requires some knowledge of the service itself, something that SRE encourages operations people to have. An even more advanced approach would be for the operations staff to not only determine why the service fails, but to find and fix the problem in the service's code. Whatever approach they choose, SRE empowers operations people to make tomorrow better than today by reducing toil through automation.

Should your operations staff become developers?

The traditional distinction between developers and operations people is breaking down. One example of this is the move to infrastructure as code, where operations staff programmatically configure their computing environment.

SRE is another step in this direction. Recall that the *E* in SRE stands for *engineering*, implying that software engineering skills are part of what's needed. In fact, adopting SRE helps put operations on a more level playing field with developers.

Embracing SRE doesn't mean you need to hire a wholly new set of operations people. But doing SRE right, which includes eliminating toil, does mean helping your people grow. One important example of this is ensuring that at least some of your operations staff have the knowledge to create and maintain automated solutions.

Reducing toil with Google Cloud

Google Cloud is designed to support SRE, including reducing toil. Here are two important examples:

- Google Cloud has broad support for automation and infrastructure as code. All of its services expose APIs that let you programmatically control your environment. While you're free to use our command-line gcloud tool or the UI console to create and manage Google Cloud resources interactively, everything this tool allows can also be done through API calls. This approach makes it possible to write code that automates toil whenever possible.
- Google Cloud operations suite provides straightforward ways to understand what's happening with services. Cloud Tracing, for example, gives operations people a clear view into what's happening as requests move through your services, while Cloud Logging enables detailed examination of what a service is doing. And along with the SLO support described earlier, Cloud Monitoring provides many other ways to understand what's happening. For example, you can use its Monitoring Query Language to examine a variety of time-series data about a running service.

Other aspects of SRE

SRE is a set of approaches to running production systems more effectively. These include using SLOs to balance new features with reliability and reducing toil through automation. There are several other aspects of SRE that are also important, including these:

- Creating an *effective incident management process*. This can include defining a clear separation of responsibilities, with an incident commander responsible for assigning roles to the people involved. The incident commander also maintains a living incident state document that tracks the incident resolution process. Having this pre-defined structure in place, then activating it when an incident occurs, is a powerful approach to finding and fixing problems.
- Performing *blameless postmortems*. Perfection isn't possible; problems will always occur. After an incident is resolved, it's essential to conduct a postmortem to document exactly what the problem was and how it can be avoided in the future. It's also essential that these postmortems avoid blame. The goal is to eliminate finger pointing, which just incents people to cover up mistakes. Creating a blameless postmortem culture helps everyone view failure as an opportunity to improve rather than a threat to their job security.
- Doing *successful capacity planning*. There are multiple ways to do this for cloud-based services. If you want to understand and predict your costs, for example, you might create a map from user interactions to required resources to expected costs. This will let you determine the cost of adding, say, a thousand new users a month. Whatever your situation, doing successful capacity planning is fundamental to running services well.

This isn't an exhaustive list—SRE includes more—but it gives you some idea of the diverse ways that SRE can improve your IT operations.

Adopting SRE

Like any organizational change, adopting SRE practices means changing parts of your culture. SRE is defined by a set of principles and behavioral tenets, so you'll lead with people and processes, not tools.

Because SRE is a set of largely independent practices, you can start wherever you see the most need. Most IT operations groups suffer from too much toil, for example, so identifying and reducing this toil is often a useful place to begin. If you manage services that need to balance speed of new feature release with reliability, it also makes sense to adopt SLOs and error budgets; giving operations a seat at the table early on brings real value. Other aspects of SRE, such as blameless postmortems and capacity planning, can be added whenever they make sense for you. And since SRE encourages continuous improvement, expect to iterate over these changes as you gain more experience.

Whatever your SRE path, you'll need to invest in your operations staff—training is key to success. What your people need to learn will depend both on their current skills and on which aspects of SRE you adopt. For example, using SLOs and error budgets successfully requires creating strong relationships between operations staff and developers. To do this, you might let people from the two groups partner to find and resolve a bug in some service. If your highest priority is toil-reducing automation, you might instead focus on training operations staff in development tools and techniques. For a more complete look at training for SRE, see Training Site Reliability Engineers: What Your Organization Needs to Create a Learning Program.

Adopting SRE also means mitigating the challenges this change brings. As mentioned earlier, for example, succeeding with SLOs and error budgets requires buy-in from IT leadership. Without this, operations probably can't enforce a slowdown in new feature deployment as the error budget shrinks. For organizations focused mostly on lowering the cost of IT, you'll need to make clear how adopting SRE can do this over time. One approach is to show immediate cost savings by eliminating toil through automation.

Adopting SRE can significantly improve how you do IT operations. Yet it's essential to realize that while the principles are common, organizations move to SRE in different ways. Before you start, think hard about your goals and where you want to begin.

How Google can help

Google's Professional Services Organization (PSO) offers paid consulting services to help organizations like yours adopt SRE. These can range from a six-week engagement to a multi-year relationship aimed at organizational transformation. Google PSO is a mixture of people with internal Google SRE experience and experience outside Google at enterprises and startups. This lets us capture both our own learnings and the knowledge of our customers.

Whatever your SRE goal, we can help. You might want to improve just one thing, such as capacity planning or defining SLOs for one of your services. Alternatively, you might want to restructure all or part of your operations organization around SRE. Whatever your needs, our goal is to teach you fundamental SRE principles, helping you be successful after the engagement ends.

"Google's tools and methodology have played an instrumental role in helping reshape our SRE practices and better serve our customers... With these efforts, we've been able to go from one release every two weeks to 20+ releases daily—a 300x increase."

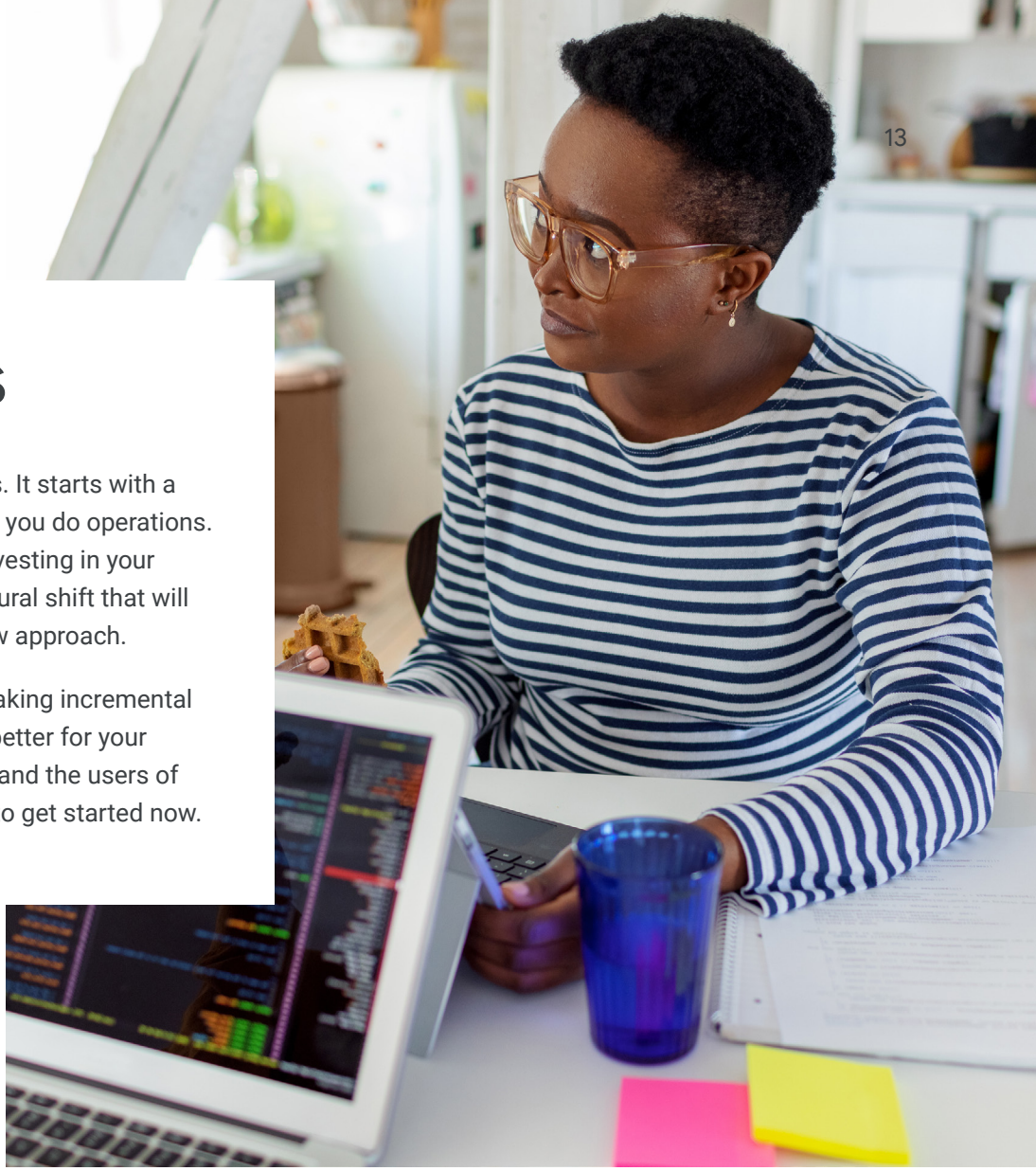
Vivek Balivada
Director, Digital SRE & Ops, Lowe's Companies, Inc.

Rahul Mohan
Kola Kandy, Sr. Manager, Digital SRE, Lowe's Companies, Inc.

For more, see How Lowe's leverages Google SRE practices.

To contact Google PSO, click here.

To read more about what Google Cloud offers for SRE, visit the Google Cloud SRE website.



Next steps

SRE isn't just SLOs and other tools. It starts with a mindset, a way to think about how you do operations. Embracing this mindset means investing in your operations people, driving the cultural shift that will let you be successful with this new approach.

In the end, SRE comes down to making incremental changes that gradually make life better for your operations staff, your developers, and the users of your services. We encourage you to get started now.

For more information

What is Site Reliability Engineering (SRE)?

The SRE Book: Site Reliability Engineering: How Google Runs Production Systems

Creating SLOs with Google Cloud Monitoring

Training Site Reliability Engineers: What Your Organization Needs to Create a Learning Program

SRE at Google: Our complete list of Customer Reliability Engineering (CRE) life lessons

Google Cloud's operations suite