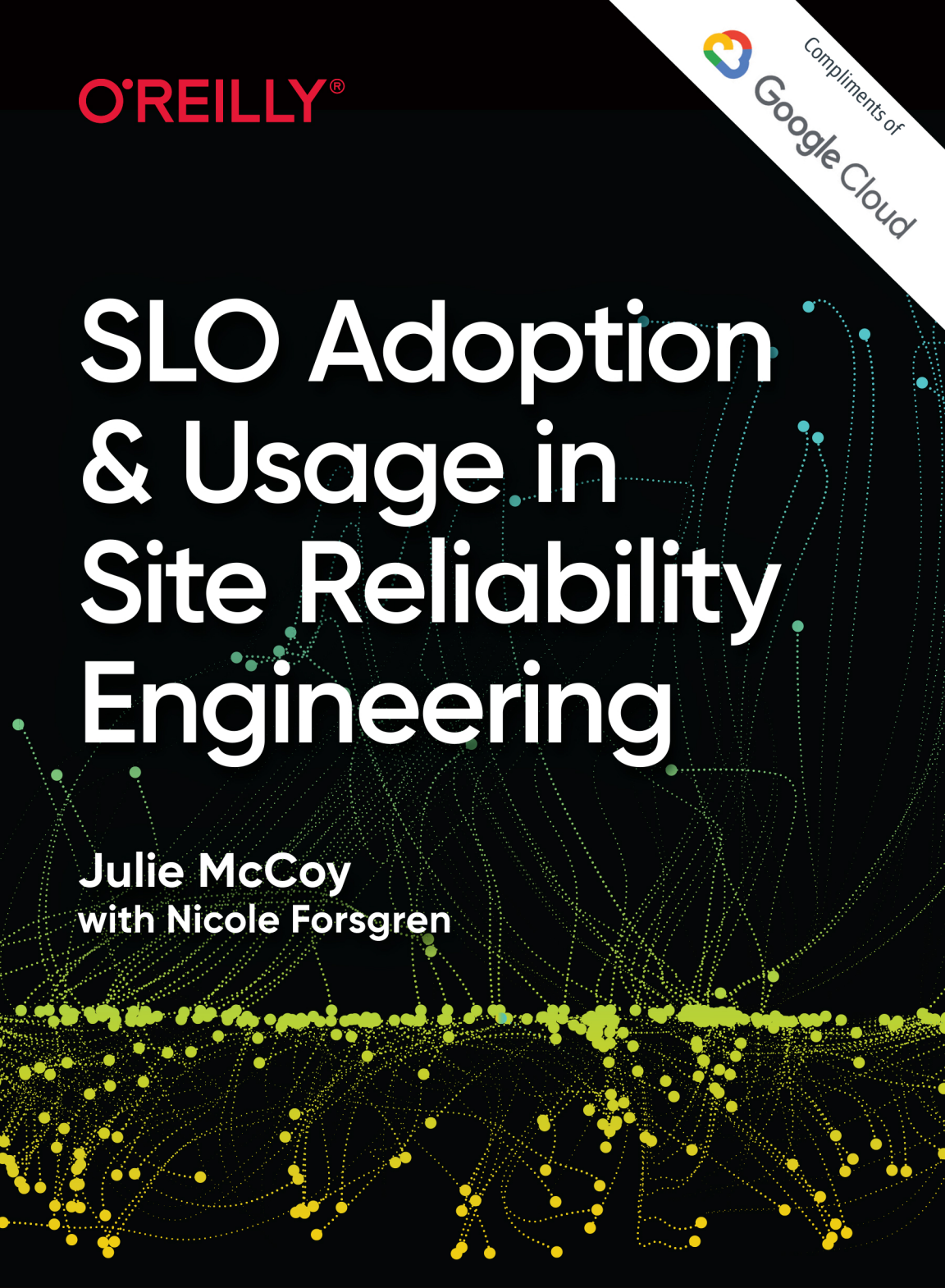
O'REILLY®

Compliments of Google Cloud

SLO Adoption & Usage in Site Reliability Engineering

Julie McCoy
with Nicole Forsgren



REPORT

Google Cloud

Want to know more about SRE?

To learn more, visit google.com/sre

SLO Adoption and Usage in Site Reliability Engineering

Julie McCoy with Nicole Forsgren

Beijing • Boston • Farnham • Sebastopol • Tokyo O'REILLY®

SLO Adoption and Usage in Site Reliability Engineering
by Julie McCoy with Nicole Forsgren

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://oreilly.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Aquisitions Editor: John Devins
Development Editor: Virginia Wilson
Production Editor: Kate Galloway
Copyeditor: Scarlett Lindsay
Proofreader: Sonia Saruba
Interior Designer: David Futato
Cover Designer: Karen Montgomery
Illustrator: Rebecca Demarest

April 2020: First Edition

Revision History for the First Edition
2020-04-01: First Release

See *http://oreilly.com/catalog/errata.csp?isbn=9781492075363* for release details.

978-1-492-07534-9

Table of Contents

Executive Summary v

Preface ix

1. SLOs: The Magic Behind SRE 1
Defining SRE Terms for Measuring and Managing Your System 2
SLOs Are the Driving Force Behind SRE Teams 4
SLOs Are Powerful Business Tools That Drive Financial and Operational Performance 6
Summary 8

2. Summary of the Data 9
Who Took Our Survey 10
Most Firms Have Had SRE Teams for Fewer Than Three Years 12
Who Uses SLOs 14
How Organizations Use SLOs 18
Availability Is the Top SLI Measurement 28
Summary 29

3. Selecting SLOs 31
Do Not Let Perfect Be the Enemy of Good 32
SLOs: What They Are and Why We Have Them 33
Characteristics of Meaningful SLOs 35
Best Practices for SLO Selection 37
Summary 39

4. Constructing SLIs to Inform SLOs. 41
Defining SLIs 42
SLIs Are Metrics to Deliver User Happiness 43
Common SLI Types 44
SLI Structure 45
Developing SLIs 46
Tracking Reliability with SLIs 49
Ways to Measure SLIs 51
Use SLIs to Define SLOs 53
Determine a Time Window for Measuring SLOs 54
SLO Examples for Availability and Latency 55
Iterating and Improving SLOs 56
Summary 57

5. Using Error Budgets to Manage a Service. 59
The Relationship Between SLOs and Error Budgets 60
Negotiating Technical Work Versus Development Velocity 61
Stakeholder Buy-in and Establishing an Error-Budget Policy 63
Summary 65

6. SLO Implementation Case Studies. 67
Schlumberger's SLO Journey 67
Evernote's SLO Journey 76
Summary 85

7. Conclusion. 87

Executive Summary

Site Reliability Engineering (SRE) is an emerging IT service management (ITSM) framework that is essential to defending the reliability of an organization's service by balancing two often competing demands: change/feature release velocity and site reliability. SRE methodology aligns teams on a common strategy for change management. Executives, product owners, developers, and Site Reliability Engineers agree upon a standard definition and acceptable level of reliability and decide what will happen if the organization fails to meet those standards. Organizations use these well-defined, concrete goals to set internal and external expectations for stability and to manage system changes against these specific performance metrics. The result is lower operational costs, enhanced development productivity, and increased feature release.

SRE and DevOps methodologies emphasize different metrics for measuring IT infrastructure and improving software delivery performance. The Accelerate State of DevOps Report published by DevOps Research & Assessment (DORA) identifies four key metrics for measuring software development and delivery (which some refer to as "DevOps performance"): deployment frequency, lead time for changes, time to restore service, and change failure rate. Although these metrics are important, focusing on speed and stability alone is not sufficient for organizations that deliver services and applications online.

DORA's report also finds that elite DevOps performers prioritize availability. Once you deliver a service to customers, they are unforgiving. Customer happiness and the value of the product diminish if users cannot access your service. SRE organizations deliver value by

identifying and monitoring service reliability behaviors that matter most to end users.

To realize the full benefits of SRE, organizations need well-thought-out reliability targets known as service level objectives (SLOs) that are measured by service level indicators (SLIs), a quantitative measure of an aspect of the service. As detailed in the following section, the measurable goals set forth in an organization's SLOs eliminate the conflicts inherent in change management and event handling that cause the pace of innovation to slow and business to suffer.

Understanding how well your service meets expectations also gives managers valuable business perspectives. SLO compliance can inform whether you invest in making your system faster, more available, or more resilient. Or, if your system consistently meets SLOs, you may decide to invest staff time on other priorities, such as new products or features.

Managing Change with SLOs and Error Budgets

In many organizations, the rift between development and operations teams runs deep. The teams have different vocabulary and assumptions for risk and system stability. Their goals also oppose one another. Development teams want to pursue maximum feature release velocity, while operation teams must protect service stability, which they achieve by rejecting changes. This tension results in considerable indirect costs because each team creates hurdles (e.g., launch and change gates or fewer releases) to prevent the other from advancing their interests. These defensive mechanisms slow feature releases and put stability at risk, which are two results no organization wants.

By design, the SRE framework mitigates this structural conflict by aligning teams on customer-centered SLOs and requiring these teams to comanage an error budget, which dictates when to roll out new releases.

SLOs Solve the Dev/Ops Split

SLOs are a precise numerical target for a service level. A core tenet of SRE is that 100% is the wrong reliability target for your system—in part because 100% reliability isn't possible. Instead, product

teams, with the guidance of SREs, should define SLOs that are less than 100%. If the business meets their well-crafted SLOs, customers should be happy. If the business misses SLO targets, customers will likely complain or will potentially stop using the service.

Product and SRE teams determine appropriate SLOs by evaluating SLIs. SLIs are server- and client-side metrics that measure a level of service, such as request latency and availability. When you understand current performance as measured by SLIs, you can set an appropriate goal, or SLO, for the service. SLOs become the shared reliability goal for development, operations, and product teams.

SRE and product teams manage the service to the SLO using an error budget. Error budgets represent the difference between 100% reliability and the identified SLOs. The error budget is one minus the SLO. A service with a 99.99% SLO has an error budget of 0.01%, meaning that the service may experience 0.01% of degraded service according to the SLIs associated with it. Development and operations teams comanage the error budget, together determining the best use of the permitted unavailability.

The error-budget concept facilitates innovation and fast release of new products and features because it allows for an acceptable failure rate. By removing the internal and external goal of zero outages, downtime becomes an accepted and expected part of innovation. Product and SRE teams can spend the error budget getting maximum feature velocity without the fear associated with failures.

SRE concepts are novel, and they are a paradigm shift for product managers, developers, and operators. It may seem counterintuitive to allow and plan for failure, but maintaining 100% reliability is prohibitive and ultimately slows progress toward business objectives. Google is not the only company to experience optimal reliability and release velocity using SLOs. This report includes case studies that detail how developing SLOs and using error budgets to manage their systems helped two companies—Schlumberger Limited and Evernote—drive better business performance and outcomes.

Key Findings from the SLO Adoption and Usage Survey

As SRE continues to gain popularity as a framework for managing enterprise software systems, we want to support organizations as they explore, consider, and adopt SRE principles. We surveyed industry professionals across a variety of industries, geographical regions, and company sizes to understand how they currently use SRE principles, especially SLOs.

Highlights from the survey include the following:

- Nearly 54% of the respondents do not currently use SLOs, but half of those respondents plan to do so at some point.
- Of the 46% of companies that use SLOs, 40% have had them in place for one year or less, and nearly two-thirds have used them for less than three years.
- We found that 43% of respondents have SRE teams. Of those, 57% implemented their teams within the last three years (31% in the last year and 26% one to three years ago).

We are encouraged to see organizations embracing SRE teams and practices in recent years. However, many organizations may not realize the full advantages of SRE because they do not have SLOs in place. Without SLOs, it is impossible to uncover business insights and drive business outcomes. It is also better to implement SLOs early on and allow them to scale with your business. Establishing even a few SLOs supports reliability, making it less costly to introduce or expand the structure later.

For those using SLOs, we hope that you are seeing the benefits of having an agreed-upon measurement for service reliability. If SLO and SLI measurements do not facilitate decisions about feature release versus stability work, take your SLO practices to the next level by revisiting and revising your SLOs. The survey reveals that only 50% of those with SLOs rarely or never engage in this best practice. Continuous improvement is a core tenet of SRE. Regularly evaluating and refining SLOs ensures that they remain relevant and measure the features most important to user happiness.

Preface

Site Reliability Engineering (SRE) is a broad field that is quickly and constantly evolving as more organizations outside of Google implement and refine SRE practices to align the needs of their services, customers, and business objectives. Although Google shaped many of the core tenets of SRE, we are excited by this next chapter and what we can learn from others' experiences.

We conducted the *SLO Adoption and Usage Survey* to get a snapshot of where organizations are in adopting SRE practices. We focused much of the survey on how organizations use service level objectives (SLOs), which we have found to be the driving force behind what makes SRE an effective framework for managing services and ensuring user happiness.

This report aims to provide insight into how Site Reliability Engineers (SREs) identify, build, and measure the effectiveness of SLOs and how organizations use the collected data to improve the reliability of their services. It also identifies gaps between usage and SRE best practices—information that can help organizations recognize opportunities to improve their own SRE practices.

The purpose of this report is to share the survey results with the industry and further the conversation about how and why to adopt an SLO- and error-based approach to managing your services. The first part of this report reviews the results of our survey and identifies areas in which organizations can strengthen their SRE practices in order to realize the full benefits of implementing the methodology. The next sections of the report provide best practices for implementing three fundamental components of SRE: SLOs, service level indicators (SLIs), and error budgets. Finally, the last chapter tells the

story of two organizations at very different points in their SLO journeys. Case studies from Schlumberger Limited and Evernote give real-world examples of how these businesses use SLOs to make better-informed decisions that drive business outcomes.

What We Hope You Take Away from This Report

Whether you are an SRE, executive, engineer/developer, operator/sysadmin, product manager, or technical lead/architect, our hope is that this report will inspire you to initially implement or advance SRE practices by implementing or enhancing an SLO and error-based approach to measuring and managing your service. SLOs, SLIs, and error budgets give enterprises a framework for managing reliability, operability, and feature release, all of which drive user happiness. Customers benefit from a reliable system that maintains a rapid release of new features that enhance their experience using your product.

Realizing the full benefits of SRE requires organizations to thoughtfully establish SLIs that inform SLOs and error budgets, and to continually evaluate and improve the quality of their SLOs.

Resources Available

This report aims to provide an overview of service-level terminology, concepts, and best practices for implementing SLOs and SLIs. If you are not familiar with SRE and SLOs, consider referring to the following materials to gain further context on these concepts and survey results.

Google's *Site Reliability Engineering* book (O'Reilly) introduces the philosophy and principles of SRE as it applies to Google's production, engineering, and operations. Our second O'Reilly publication, *The Site Reliability Workbook*, expands on the content by providing practical and replicable implementation detail to the principles outlined in *Site Reliability Engineering*. Specifically, we suggest you read Chapter 2, "Summary of the Data," which provides a step-by-step guide for establishing SLOs. To learn how other organizations successfully implemented SRE and SLOs, also read the Evernote and Schlumberger Limited case studies in this report, which build upon the information provided in the *Workbook* and provide more detail

about the survey's findings on specific practices that many companies are not implementing.

"The Calculus of Service Availability," by Treynor, Dahlin, Rau, and Beyer, is another great resource for organizations that are initially establishing or looking to expand how they use SLOs. It discusses SLOs that focus on service dependencies, looking specifically at how dependencies inform the availability of a service and how to code to avoid critical dependencies.

Acknowledgments

A special thanks to our technical reviewers, Fred Moyer, Matthew Hite, and Eric Harvieux, who provided thoughtful comments and revisions throughout the report.

CHAPTER 1
SLOs: The Magic Behind SRE

As one might gather from the name, Site Reliability Engineering (SRE) prioritizes system reliability.[1] Ben Treynor Sloss, Google's vice president of 24/7 operations, who coined the term *SRE*, says reliability is the most vital feature of any service. If a system is not available and reliable, customers cannot use it, and the value the product provides diminishes.

No matter where your organization is in adopting SRE methodology, establishing service level objectives (SLOs) is a core best practice that organizations cannot overlook or put off until the future. SLOs safeguard the reliability of your service, and your customers likely will not be happy unless your service is reliable. SLOs set an explicit threshold for how reliable your service should be, which provides all stakeholders within an organization—executives, Site Reliability Engineers, developers, product managers, and operations teams—a common language for measuring service reliability and quality goals.

Thoughtfully constructed, customer-centric, and continuously improved-upon SLOs benefit the entire business. Think of SLOs as goals used to measure success, and service level indicators (SLIs) as the direct metrics of the service's performance that inform whether

1 For our purposes, *reliability* is defined as "the probability that a [system] will perform a required function without failure under stated conditions for a stated period of time." See P. O'Connor and A. Kleyner, *Practical Reliability Engineering,* 5th ed. (Hoboken, NJ: Wiley, 2012), 2.

the service is meeting the SLO. In other words, SLIs tell you good events from bad events. SLOs tell you what proportion of good/bad events is acceptable, all in support of organizational goals.

Organizations must define objectives (SLOs) and indicators (SLIs) for measuring progress toward the goal. Teams then strive to reach targets by aligning their priorities and decision making with the goals.[2] Organizations using SLOs and SLIs can make data-informed decisions about where to invest resources. If your service performs better than its SLOs, then you can invest in greater development velocity (for example, new features). If the system experiences issues and violates its SLOs, stakeholders can slow development and prioritize tasks to make the system more reliable.

In this chapter we define common service-level terminology and detail how organizations can leverage SLOs as powerful business tools.

Defining SRE Terms for Measuring and Managing Your System

Before exploring how SLOs drive business outcomes, we will review and differentiate between SLOs, SLIs, and service level agreements (SLAs). We'll also explain how, together, they provide a framework for defining and measuring the level of service provided to customers.

As we define these terms, readers may find it helpful to frame the relationship between these three concepts in the following way: SLIs drive SLOs, which inform SLAs.[3]

SLIs: How Do We Measure Performance Against Our Goals?

You cannot implement SLOs without first identifying and defining SLIs for your service. Google's *Site Reliability Workbook* defines SLIs as metrics over time that inform the health of a service. SLIs reflect

2 See Fred Moyer, "A Guide to Service Level Objectives, Part 1: SLOs & You," Circonus, July 11, 2018, *https://www.circonus.com/2018/07/a-guide-to-service-level-objectives*.

3 Google Cloud Platform, "SLIs, SLOs, SLAs, oh my!" video, 8:04, March 8, 2018, *https://www.youtube.com/watch?v=tEylFyxbDLE&t=6m08s*.

business objectives and customer expectations; they specify which aspects of the service you are measuring and how you measure them.

Many organizations use SLIs to measure common system characteristics, such as the following:

Availability
: The fraction of the time that a service is usable, as expressed by a fraction of well-formed requests that succeed.

Latency
: How long it takes to return a response to a request.

Error rates
: Expressed as a fraction of all requests.

System throughput
: Often measured in requests per second.

Durability
: The likelihood that the system will retain the data over a long period of time.

SLIs are quantitative measures often aggregated over a measurement window and expressed as a rate, average, or percentile. We typically prefer to work with SLIs stated as a ratio of two numbers: the number of good events divided by the total number of events. For example, you may look at the number of successful HTTP requests / total HTTP requests. The SLI value would range from 0% (nothing works) to 100% (nothing is broken).

Once you know how you will measure your service using SLIs, you can determine the targets you want to achieve and state them as an SLO. In other words, SLIs are the base metric used to compose an SLO.

SLOs: What Are Our Goals?

SLOs set a precise target level of availability for customers as measured by an SLI. They are defined thresholds for how often SLIs must be met. SLOs allow organizations to judge whether the value of an SLI measurement is good or bad. SLOs align all stakeholders—from individual contributors to vice presidents—on a common definition and a measured standard for reliability, focused on the customer.

This common understanding promotes a shared sense of responsibility for the service.

At Google, SLOs frame all conversations about whether the system is running reliably and whether it is necessary to make design or architectural changes in order to meet the SLO.[4] Because SLOs have product and business implications, product owners should choose SLOs, with the input of Site Reliability Engineers.

Determine SLO thresholds by looking at your SLIs. SLOs are binding targets for SLIs and may take the form of a target value or range of values. Common structures for SLOs include SLI ≤ target, and lower-bound ≤ SLI ≤ upper-bound. They should specify how they're measured and the conditions under which they are valid. For example, we might say that 99.9% of GET RPC < 10 ms over 28 days, as measured across all backend server logs.

SLAs: What Level of Service Are We Promising Our Customers?

SLAs are business-level agreements (often in the form of legal agreements between two organizations) that state the SLOs your service will meet over a given time frame. SLAs also detail the remediation you will take, such as issuing money back or providing free credits, if the service misses the SLOs. The business typically sets the terms of the SLAs with customers, but Site Reliability Engineers focus on defending the SLOs included in SLAs.

We recommend that SLAs contain a slightly less restrictive SLO than your internal target, such as an availability target of 99.95% internally versus 99.9% shared with customers. A more restrictive internal SLO will provide you with an early warning before you violate the SLA.

SLOs Are the Driving Force Behind SRE Teams

As a part of committing to SRE, organizations must believe their success is predicated on service reliability. Site Reliability Engineers

4 See Jay Judkowitz and Mark Carter, "SRE Fundamentals: SLIs, SLAs and SLOs," Google Cloud Platform, July 19, 2018, *https://cloud.google.com/blog/products/gcp/sre-fundamentals-slis-slas-and-slos*.

cannot manage their services correctly if they have not identified the behaviors most important to the service and to customers or how to measure them. Carefully considered SLOs prioritize the work of SRE teams by providing data points that allow leaders to weigh the opportunity cost of performing reliability work versus investing in functionality that will gain or retain customers.

SLOs establish thresholds for acceptable levels of reliability that SREs must protect and maintain—this is their primary responsibility and drives their priorities and daily tasks. Defending SLOs is also a core competency of Site Reliability Engineers. Their skill set goes far beyond automating processes or troubleshooting outages. Instead, organizations should align SREs' tasks and priorities with the most important aspects of the most important services as defined by SLOs.

SLOs help SREs defend user happiness by clearly defining a target for service performance. SRE teams can easily see when quality of service declines below the SLO threshold and know that action must be taken.

SLOs are also critical elements in the control loops that SREs use to manage systems:

- Monitor and measure the system's SLIs.
- Compare the SLIs to the SLOs and determine whether teams must take action.
- If action is needed, determine what needs to happen in order to return service to the target level.
- Take action.

Without SLOs, organizations will fail to realize the full value of their SRE teams. SLOs allow these specialized engineers to systematically communicate reliability while enhancing reliability by improving the product's codebase.[5]

5 See Kurt Andersen and Craig Sebenik, *What Is SRE? An Introduction to Site Reliability Engineering* (O'Reilly, 2019).

SLOs Are Powerful Business Tools That Drive Financial and Operational Performance

SLOs not only guide SRE teams but also provide insightful perspectives that drive financial and operational performance. How is that possible? SLOs and SLIs eliminate all the often fuzzy definitions of *reliability* and provide hard data by which you can measure whether your service is meeting its reliability targets.

SLOs and Error Budgets Allow Maximum Change Velocity While Protecting Stability

Part of what makes SLOs and SLIs powerful business tools is that they leave room for failure. We recommend that organizations set SLO thresholds below 100%. Each application has a unique set of requirements that dictate how reliable it must be until customers no longer notice a difference. Most customers will not and cannot distinguish between 100% reliability and 99.9% or, in some cases, 99.0% reliability. However, the costs and technical complexities associated with maintaining 100% reliability are immense. If the target is less than 100%, the organization can allocate resources that would otherwise be spent maintaining 100% reliability to other strategic initiatives.

Setting SLOs below 100% also allows organizations to leverage another important SRE service management tool: error budgets. The difference between 100% reliability and the SLO is your error budget, or the specified amount of downtime the service can experience during a set period. You can have an error budget only if your SLO is less than 100%.

By using an error budget to manage progress toward SLOs, organizations can confidently manage risk and make decisions about when to release features without sacrificing user happiness. All stakeholders must agree that if you exhaust or come close to exhausting the error budget, teams should stop other development work and focus on restoring stability. If you have a sufficient error budget, you can take actions, such as binary releases or configuration pushes, that may result in outages. Although users may experience unhappiness during your allotted downtime, the alternative of not releasing new features will likely cost the business more.

SLOs Keep Business Decisions Focused on Customer Happiness

It is tempting to react to metrics that your team really cares about—for example, CPU or memory utilization. These metrics possibly indicate reliability issues and are easy to graph and understand, but a central tenet of SRE is that organizations should establish SLOs that measure the service attributes that matter most to their end users. Your users don't care how many cores you're using, or how much memory is available, as long as your service works for them.

As discussed earlier, SRE methodology is built on the principle that reliability is the most important attribute of your system. But, SRE also says that's true only to a point. Because SLOs define the point at which customers will become unhappy with the reliability of your service, they also ensure that SREs, product teams, developer teams, operations teams, and executives understand and measure reliability as it matters to the customer.

SLOs Set Customers' Expectations

Organizations must set appropriate expectations for the reliability of their service. Including SLOs in SLAs or publishing SLOs for internal customers explicitly communicates how available your service will be. If your service is used as a backend to other services (for example, a database accessed by a web frontend), your SLO needs to be better than the SLO the frontend desires to meet. Having a defined metric prevents users from under-relying on your system or over-relying on your system.

When organizations count on excessive availability, they may create unreasonable dependencies on your system. Google experienced this with Chubby, our lock service for loosely coupled distributed systems. Because Chubby rarely went down, service owners added dependencies to the system. When Chubby did go down, the services could not function properly, causing the outages to be visible to end users.

To solve the problem of over-reliance, SRE ensures Chubby meets but does not exceed its SLOs. If, in any quarter, a true failure does not drop availability below the target, we will deploy a controlled outage.

Organizations also do not want to beat their SLOs too much. If you start running your service at a performance level that's consistently better than your actual SLO, users will come to expect that level, and they will be unhappily surprised if they're trying to build other services on top of yours.

Summary

SLOs and error budgets are powerful data points that stakeholders can use to manage their services. SLOs can—and should—be a part of nearly all service-related business discussions. These numerical thresholds give decision makers data-driven insights that allow them to better balance development velocity and operational work.

Adopting an SLO and an error-based approach helps inform development work and associated risk discussions about change management. Teams can spend the error budget as they wish, as long as they do not exceed the SLO. Because the goal is no longer zero outages, SREs and product developers can manage the budget to attain maximum feature velocity.

In the next chapter, we will share the findings of the *SLO Adoption and Usage Survey*. We will highlight key SLO practices that the data indicates organizations are not implementing. Failure to implement these practices may be keeping organizations from fully attaining the valuable business insights we described throughout this chapter.

CHAPTER 2
Summary of the Data

When done well, using SRE methodology to manage your services leads to improved business perspectives and outcomes. We've seen the power of SRE at Google and at many other organizations that span a variety of industries, sizes, and company cultures. However, when we engage the industry in conversations about SRE, organizations frequently question why they have not seen the tangible results and benefits described in our books or shared at industry conferences or in publications.

To understand what prevents organizations from realizing the full benefits of SRE, we decided to survey the industry on how they apply SRE practices and SLOs. The S*LO Adoption and Usage Survey* focuses on SLOs because they provide a foundation for all other aspects of SRE methodology. Without them, you are not fully engaged in SRE and likely won't realize the benefits you seek.

The *SLO Adoption and Usage Survey* quantifies SLO adoption and gauges the extent to which organizations exercise SRE and SLO best practices. It also looks at how the maturity of respondents' SRE practices and their SLOs impact how they use these tools. Our analysis finds that if organizations are in the initial phases of establishing SLOs, they often skip or delay best practices, but doing so may result in fewer benefits for their organizations. If you already have SLOs, compare these survey results against your own practices to discover areas of opportunity for improvement.

Who Took Our Survey

We surveyed a cohort of professionals with an interest in development and operations topics. We received responses from 572 industry professionals around the world.

Geographic Region

The majority of the survey respondents work in North America (42%) and Europe (35%). The remaining 23% work in Asia, South America, and Africa/the Middle East.[1]

Principal Industry

The survey captured data from respondents in 12 diverse industries (see Figure 2-1). The technology industry dominated, representing one-third of all respondents, followed by the financial services (14%) and retail (8%) industries.[2]

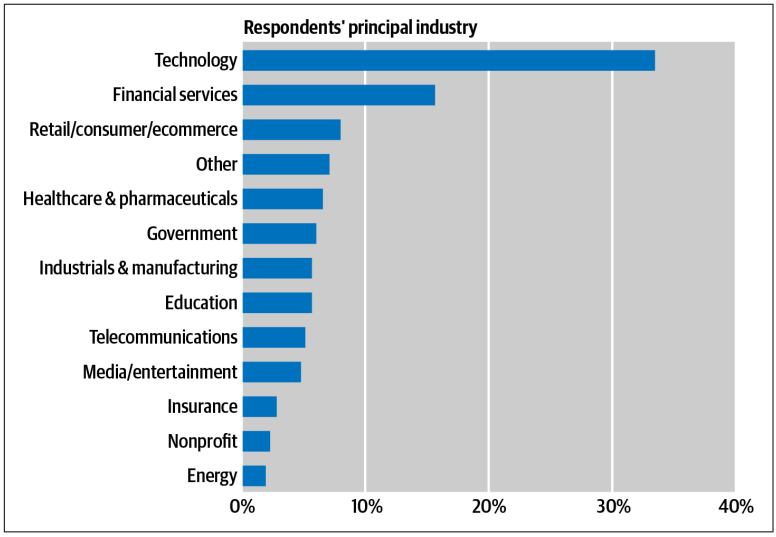


Figure 2-1. Respondents' principal industry

1 The survey specifically asked, "Where do you work?" It did not ask about the location of the organizations' corporate headquarters.

2 Eight percent of respondents answered "Other," which made up the fourth largest share of respondents.

Organization Size

Figure 2-2 shows the size of respondents' organizations as measured by the number of employees. Nearly 50% of respondents work in organizations with more than 1,000 employees. Companies with 100 or fewer employees make up 40% of the respondents, which constitutes the largest share of any one group. Respondents from midsize businesses (101–1,000 employees) make up the smallest share of respondents (14%).

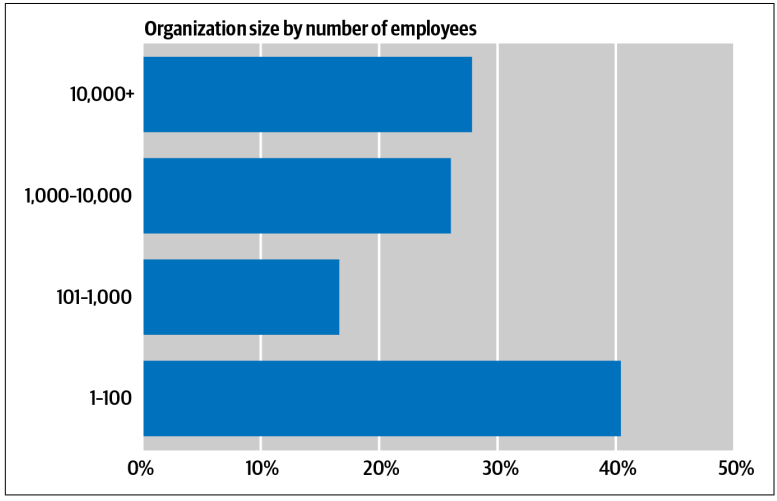


Figure 2-2. Organization size by number of employees

Titles of Survey Respondents

Survey respondents represent a range of positions and titles (see Figure 2-3). Overall, technical roles dominate, with developers, technology leads, and architects making up two-thirds of respondents. Of those, the best-represented group is engineers/developers (33%), followed by DevOps/admin/systems (18%). Managers and C-level executives comprise 20% of the survey respondents. Site Reliability Engineers represent 7% of the respondents.

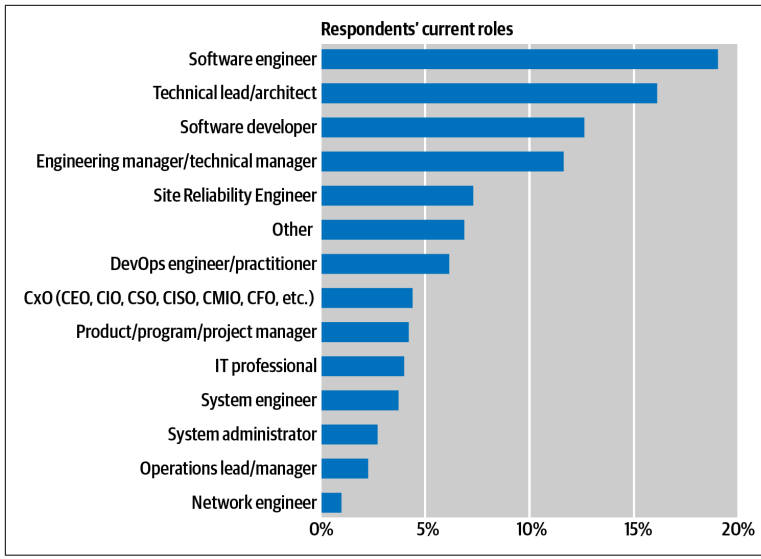


Figure 2-3. Respondents' current roles

With an understanding of the makeup of our survey respondents, let's now look at how they use SRE practices.

Most Firms Have Had SRE Teams for Fewer Than Three Years

SRE is a relatively new profession, so it is not surprising that only 43% of those surveyed have an SRE team.[3] SRE teams are new to many of those organizations. Of those with SRE teams, the majority of respondents (57%) established their SRE teams within the last three years, with 31% establishing teams just in the last year (see Figure 2-4). The recent decision for many companies to implement SRE teams may be due to increasing awareness in the industry of SRE and the prevalence of media, reports, and conferences that highlight the benefits of having a team dedicated to SRE practices.

Though the majority of respondents established SRE teams within the past three years, it is important to point out that 31% of

3 We did not define the term *SRE team* for respondents, so we do not have information on the composition and responsibilities of these teams.

respondents say their organizations established an SRE team five or more years ago.

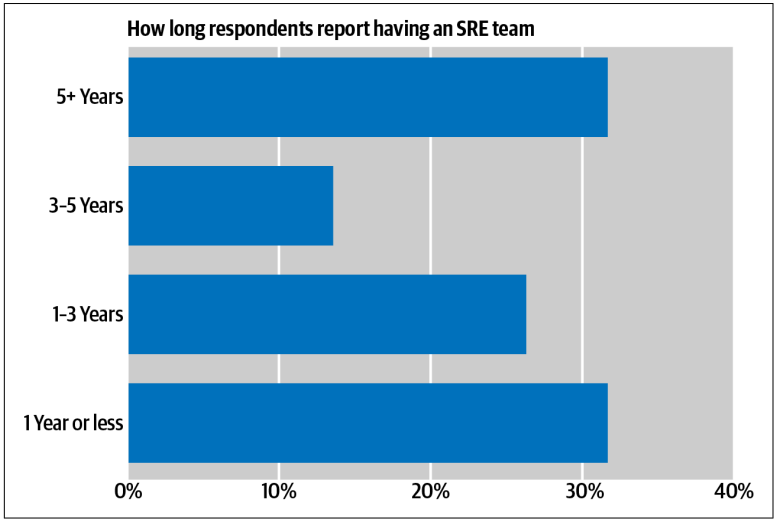


Figure 2-4. How long respondents report having an SRE team

Figure 2-5 shows a distinct relationship between company size and whether organizations have SRE teams. Sixty-four percent of the respondents in organizations with 10,000+ employees have SRE teams, compared to 44% of companies with 101–1,000 employees and 32% of those with 100 or fewer employees. This is not surprising because large organizations typically have more resources to dedicate to SRE than smaller organizations do and can often hire specialized SRE engineers.

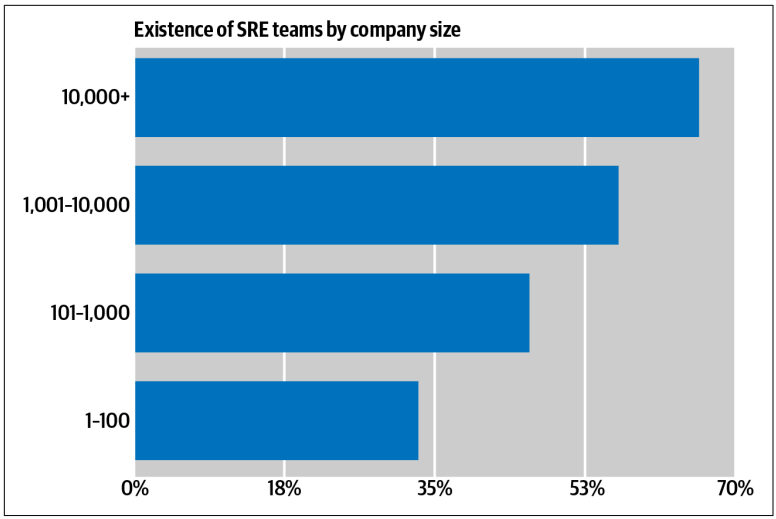


Figure 2-5. Existence of SRE teams by company size

Now that we have a snapshot of how the industry has implemented SRE teams, let's see how organizations use the most powerful SRE tool: SLOs.

Who Uses SLOs

The survey shows that SLOs are an underutilized SRE practice. Less than half (46%) of the respondents use SLOs. However, many organizations plan to adopt SLOs. Of the 54% of respondents who do not have SLOs, 47% intend to develop them in the future. Interestingly, 35% said that they use a different metric to track user happiness.

Of those with SLOs, the way in which respondents work with SLOs is fairly evenly distributed.[4] In Figure 2-6 you can see that most respondents are involved in identifying metrics for evaluating SLO (36%). Working with users to develop SLOs was the least selected option, with only 28% choosing this answer.

4 Respondents could choose more than one option.

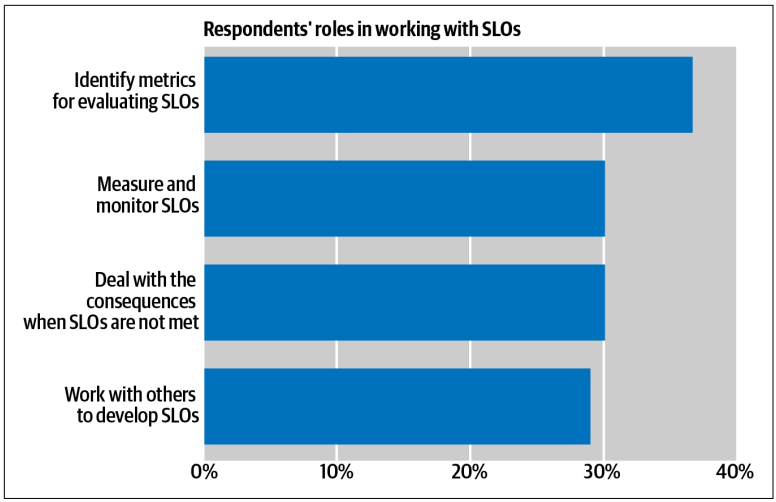


Figure 2-6. Respondents' roles in working with SLOs

SLOs Are a New Practice for Many Organizations

The data also shows that, similar to how SRE teams are a newer addition to several organizations, measuring services with SLOs is a relatively new practice for many (see Figure 2-7). Of the 46% who have SLOs, 44% implemented them in the last year, and two-thirds (66%) began using them in the last three years. One quarter of respondents established SLOs five or more years ago.

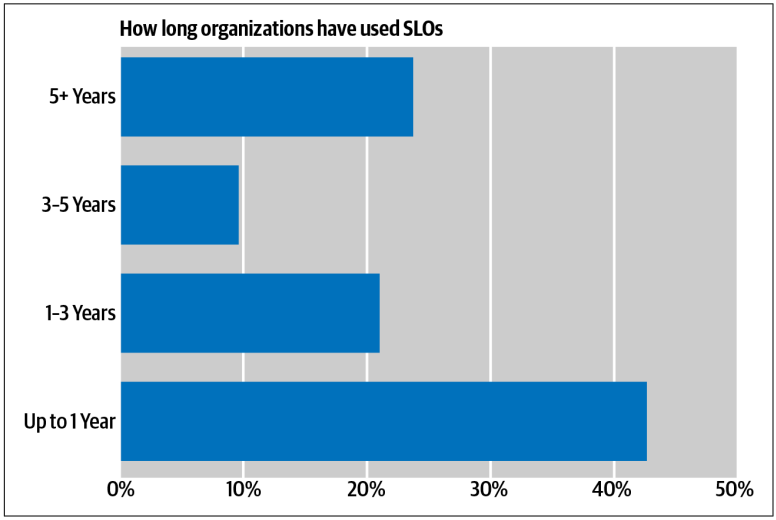


Figure 2-7. How long organizations have used SLOs

Large Companies Have More Experience Using SLOs

Large-sized organizations are most likely to have had SLOs in place for five or more years (Figure 2-8). Small companies have the least experience with SLOs, mostly falling in the "Up to 1 year of experience" category and having little representation in the 3+ year groups. The smallest organizations (1–100 employees) make up the largest share of those with no SLOs and those with a year or less of SLO experience.

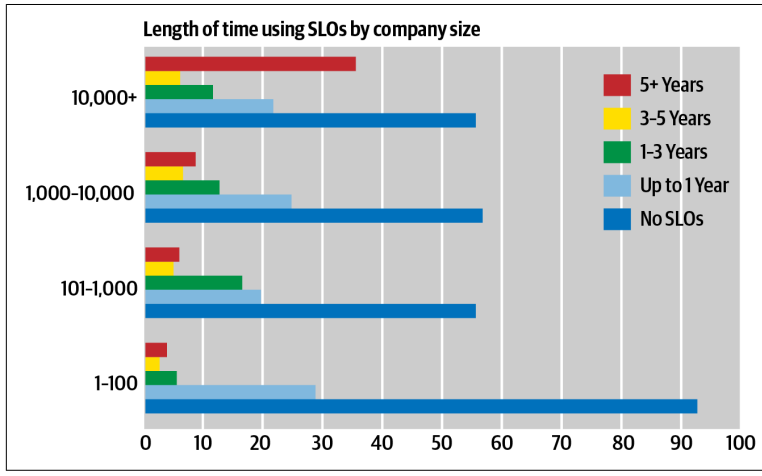


Figure 2-8. Length of time using SLOs by company size

Companies with thousands of employees may have a longer history with SLOs because they traditionally utilized formal SRE practices to coordinate their larger workforces and customer bases. Whereas smaller companies may not have the resources to dedicate to fully separate SRE organizations or teams, they are recognizing the value provided by SRE methodology and practice and are beginning to support SRE-based initiatives.

In particular, organizations both large and small can benefit from and successfully implement SLOs. SLOs are a cornerstone of reliability practices that can tie customer-focused outcomes to DevOps work and prioritization, and can be carried out by any organization.

Recent SLO Adoption in Europe Outpaces Other Regions

Figure 2-9 shows that SLOs are a new practice for many companies around the globe. Of the cohort that uses SLOs, 50% of European organizations adopted SLOs in the past year, followed by 38% of Asian and 33% of North American (33%) companies.

Respondents from North America and Asia have used SLOs for much longer than respondents from Europe. In both North America and Asia, 31% percent of respondents have had SLOs in place for more than five years, compared to 21% for European respondents.[5]

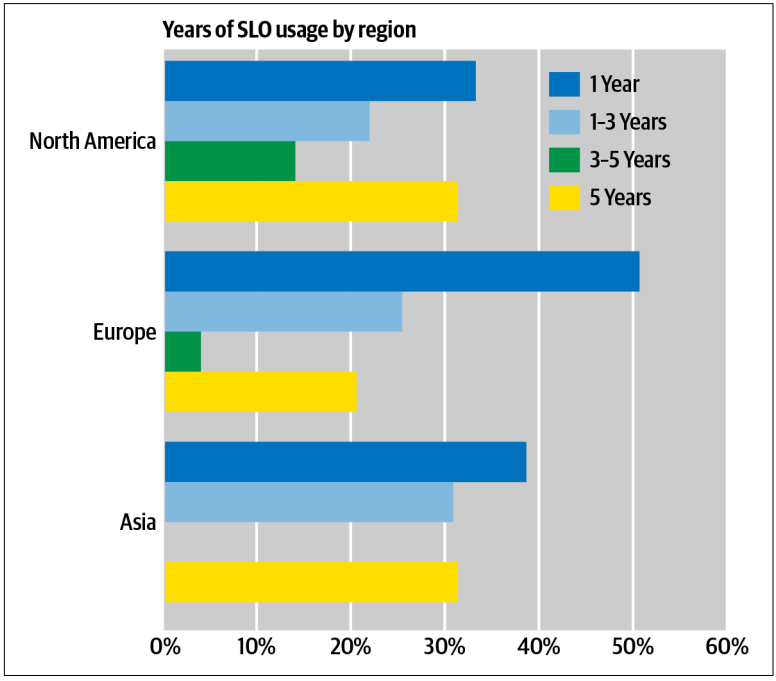


Figure 2-9. Years of SLO usage by region

5 We did not include Latin America due to the small number of respondents from the region.

How Organizations Use SLOs

We also looked at how respondents implement their SLOs in practice. This section discusses the following:

- SLO usage compared to other common SRE best practices
- The types of services respondents are most likely to measure using SLOs
- SLO thresholds
- Actions taken when a service does not meet its SLOs

Most Firms Embrace SRE Practices but Fail to Engage in SLOs

The survey looks at how respondents use other SRE practices compared to SLOs in their work.[6] Figure 2-10 shows that most respondents engage in the SRE best practices of applying software engineering to operations work (75%), capacity planning (45%), and blameless postmortems (40%).[7] Just over a third of respondents (34%) develop SLOs that describe the availability or performance of their service. The same number have SLAs that describe the impact of not meeting their SLOs. The fewest respondents develop specific SLI metrics to inform their SLOs (31%). (In the Chapter 6 case study, we look at how specific SLI metrics enhanced Schlumberger's ability to respond rapidly with a DevOps approach to software development.)

6 When analyzing the data we found a discrepancy between the number of respondents who report using SLOs and those who selected SLOs as an answer option when asked what SRE practices they use (Figure 2-10). For example, 4% of the respondents who answered that they do not use SLOs selected the option that they provide their users with SLOs. Respondents may have interpreted the questions related to this data differently. Figure 2-7 is based on the question, "How long have you been using SLOs," which includes answer options using "We," which may have resulted in survey-takers answering on behalf of their organization. Figure 2-10 is based on a question, "What SRE practices do you use in your work," which includes options that specify a relation between users and SLOs. We are unsure of how survey-takers interpreted the question, so it is difficult to explain the difference in users reporting that they use SLOs.

7 Data calculated based on the number of respondents who answered the question.

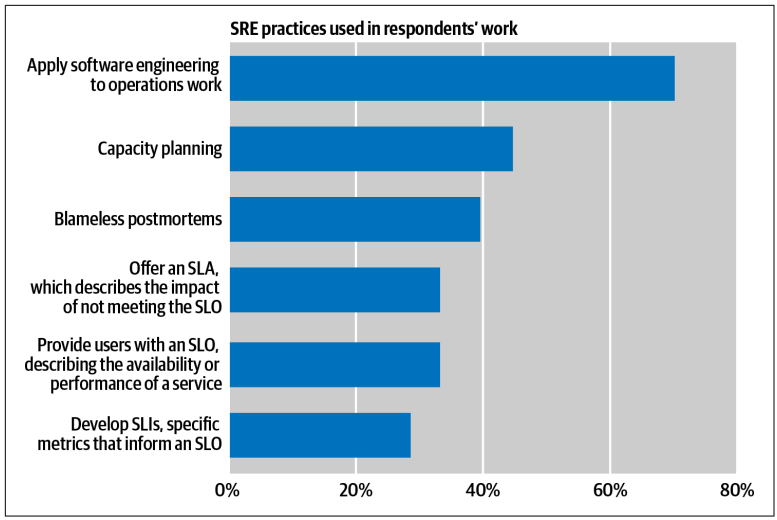


Figure 2-10. SRE practices used in respondents' work

Figure 2-11 shows the distribution by maturity of the six common SRE best practices listed previously. The chart supports the notion that, with the exception of capacity planning, many organizations are just starting to adopt SRE best practices, with many organizations adopting these practices within the last three years. The distribution in Figure 2-11 for applying software engineering to operations work has a different pattern compared to the other SRE practices. It skews toward recent adopters of SRE practices, indicating that applying software engineering to operations is a first step many organizations take when implementing SRE.

SLO adoption follows the same pattern as the best practices listed in Figure 2-6. The largest cohort adopted SLOs within the past year, followed by those using SLOs for more than five years. However, SLOs are the least commonly used SRE practice when compared to the other SRE practices that firms implemented in the past year.

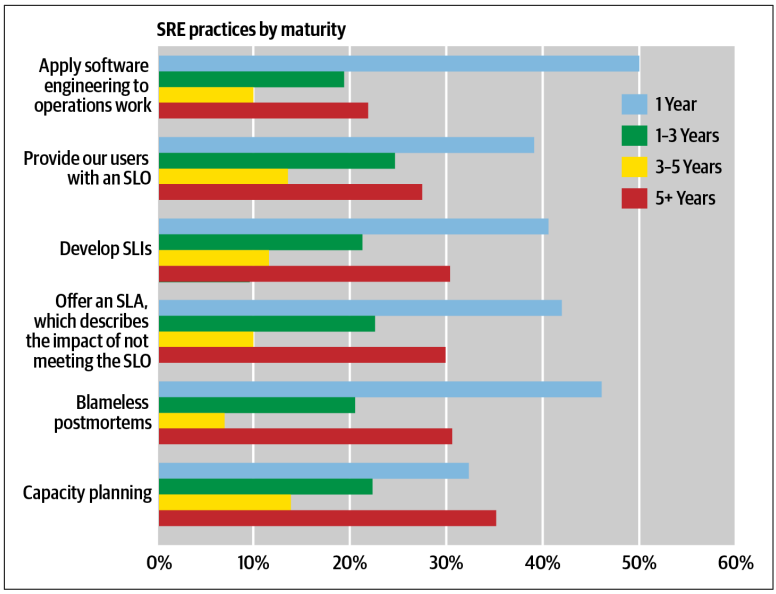


Figure 2-11. SRE practices by maturity

Figure 2-12 shows that the longer organizations have had SLOs in place, the more likely they are to utilize many SRE practices. Of the respondents with more than five years of SLO experience, 44% used five or more of the SRE best practices, and 26% used two or fewer SRE practices.

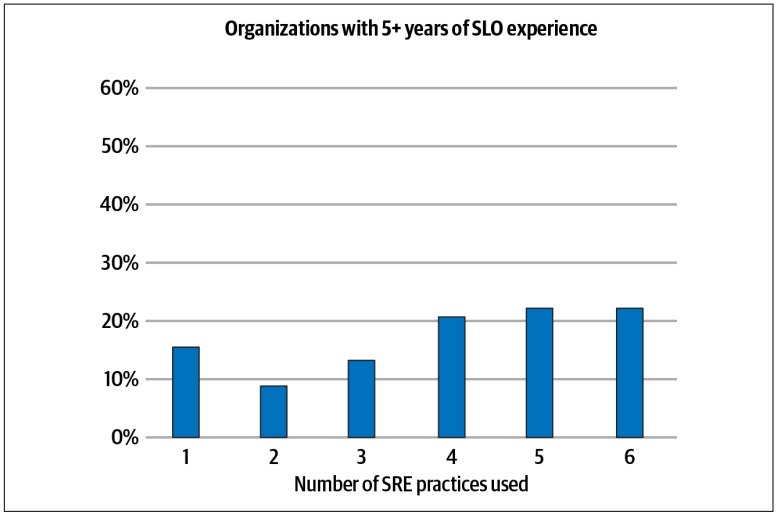


Figure 2-12. Organizations with 5+ years of SLO experience

Of the 55% of respondents who use two or fewer SRE practices, only 10% have more than five years of SLO experience, and 77% report having no SLO experience.

SRE is a new field, and the data implies that organizations are still adopting these best practices, including SLOs. We are encouraged to see that companies are thinking about reliability. However, SRE practices, such as applying software engineering to operations, are only one part of the SRE equation. These activities constitute the practical ways you build and maintain the reliability of systems, but SLOs and SLIs measure your success. Without these metrics, you cannot effectively decide between investing in functionality that will win and retain customers, and investing in reliability and scalability that will keep customers happy.

Critical Infrastructure Is the Most Common Service Measured by SLOs

Figure 2-13 shows that when choosing which services should have SLOs, respondents are most likely to establish SLOs to measure their critical infrastructure. Survey-takers also prioritize establishing SLOs for user-facing services and services that have paying users.[8]

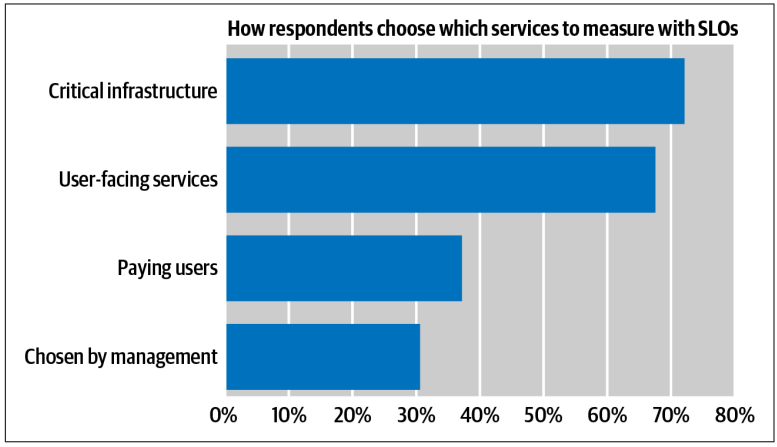


Figure 2-13. How respondents choose which services to measure with SLOs

8 Data calculated based on the number of respondents who have SLOs. Data excludes the 11% who have SLOs but did not select one of the options provided. Respondents were able to select more than one answer.

Other than to measure critical infrastructure, the reasons for establishing SLOs seem to be mostly externally driven. User-facing services and services with paying users were the most externally driven options provided by the survey: 77% of respondents selected at least one of those options, and 29% chose a combination of options that included both user-facing and paying-user options. The combination of critical infrastructure and user-facing services was the most popular, with 24% of respondents selecting both of those options. The second most popular combination, selected by 16% of respondents, was critical infrastructure, user-facing services, and services with paying users. Only 8% selected all four options.

Figure 2-13 indicates that SLOs are largely derived independently from management. Of those who answered the question, 70% selected a combination of options that did not include "Chosen by management." Of the 31% who selected "Chosen by management," 7% chose only that option. Fifteen percent included "Chosen by management" with at least two other options.

Although the data indicates that management does not often participate in selecting services for SLOs, management does have exposure to SLO performance. Figure 2-14 shows that survey respondents are more likely to select customers and executive management as the primary audience for their SLOs.[9] When we look at the combinations of answers selected by respondents, executive management was selected in combination with SRE teams and development partners 29% of the time. Ten percent chose "Executive management" as their only response.

9 Data calculated based on the number of respondents who have SLOs. Data excludes the 11% who have SLOs but did not select one of the options provided. Respondents were able to select more than one answer.

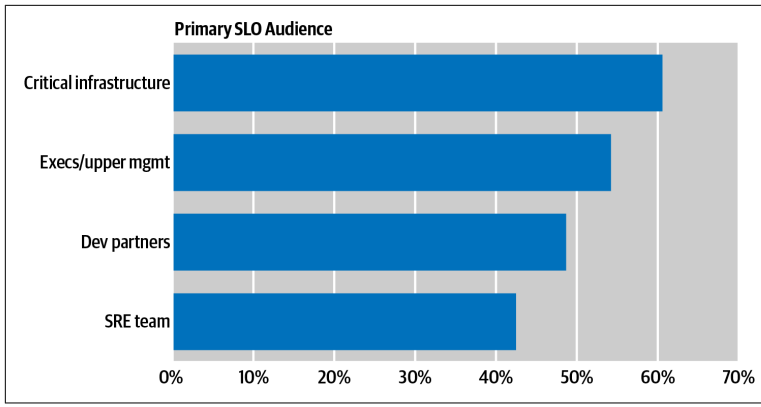


Figure 2-14. Primary SLO audience

Majority of Respondents Measure "Some" of Their Services with SLOs

Respondents do not measure all of the services they are responsible for. The vast majority of respondents have SLOs for some (54%) or most (33%) of the services they support, whereas only 12% have SLOs for all of their services (see Figure 2-15). The remaining 2% have no SLOs.[10] Because so many respondents are fairly new to using SLOs, these responses are not surprising. Implementing and measuring SLIs and SLOs is not a small task, though as we demonstrated earlier, it is an important one. We recommend that organizations begin with measuring just a few services that are critical to reliability and/or the customer experience. Once you have a few SLOs in place, you can expand your SLO practice.

10 Due to rounding, totals are greater than 100%.

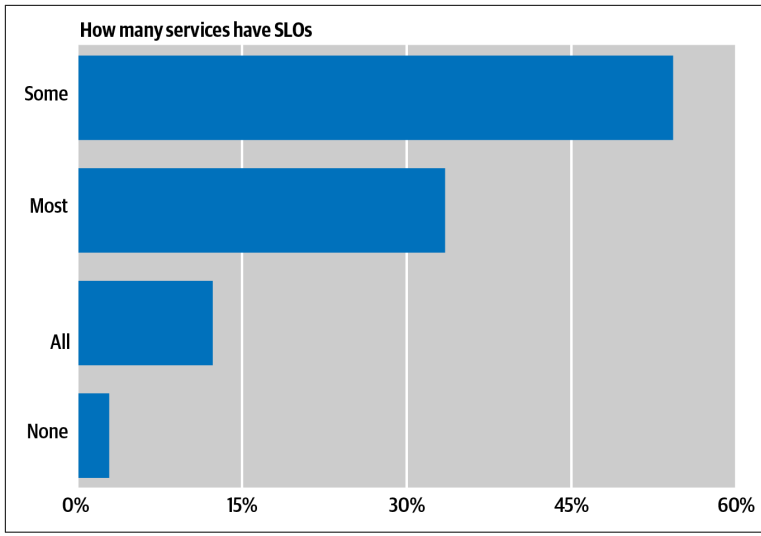


Figure 2-15. How many services have SLOs

SLOs Above 99% Are Most Common Among Respondents

The most common SLO[11] targets set by respondents are 99.0% (27%), 99.90% (25%), and 90% (22%). Six percent most commonly set their target at 100%.

Figure 2-16 shows that 61% of respondents set their strictest SLOs above 99%. Of those with SLOs higher than 99%, 19% set their strictest SLO at 99.99%, 24% set it at 99.9%, and 18% set it at 99%. Nearly 30% of the respondents say their tightest SLO is 90% or lower, with 18% setting it at 90%. Despite the fact that 100% availability is an unattainable goal, 12% of the respondents set 100% as their highest SLO target.[12]

11 Defined as, "What level of reliability is required of most of the systems you support."

12 Due to rounding, totals are greater than 100%. Data calculated based only on those respondents who indicated they have SLOs.

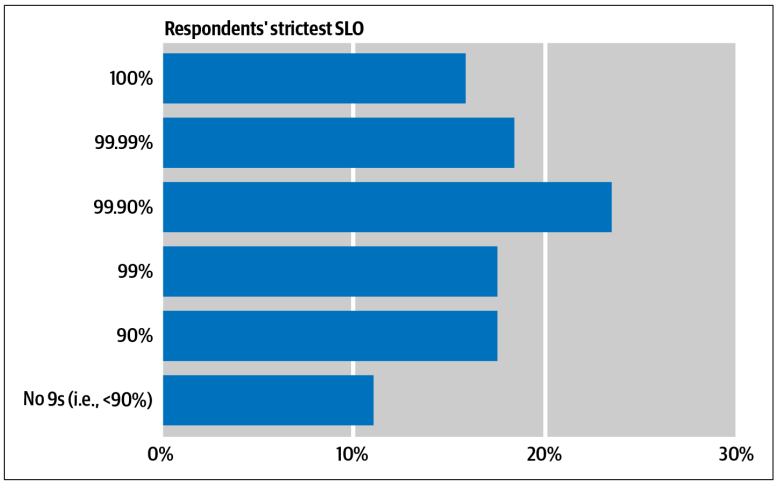


Figure 2-16. Respondents' strictest SLO

Of those respondents who use SLOs, 80% say that they often or almost always comply with their SLOs. More than one-third (35%) of the respondents say that they sometimes meet their SLOs, while 1% report that they never meet them (see Figure 2-17). Similar results, in which many respondents indicated that they do not review the SLO themselves, appear in "SLO Reviews Are Underutilized by the Majority of Respondents " on page 27.

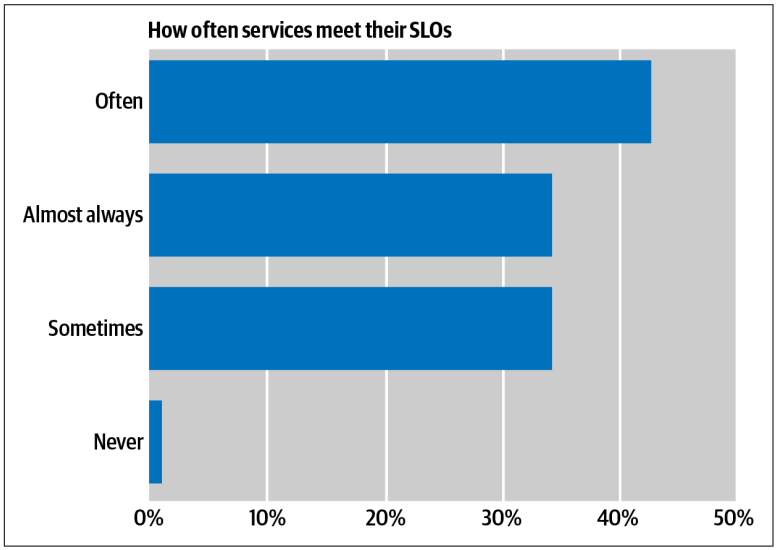


Figure 2-17. How often services meet their SLOs

Internal Action Is the Most Common Response to Missing SLOs

SLOs are meant to drive action and business decisions, so we also asked respondents what happens when their services fail to meet their SLOs. Figure 2-18 shows that 90% of respondents take some action when they miss SLO targets.[13] Nondisciplinary, internal-only action was the most common response, with 73% of the respondents selecting it as one of their answer options and 55% choosing it as their only response to the question. Although most survey-takers address SLO misses internally, 31% of the respondents offer service credits to all users, not just those affected by the outage.

Internal actions may be the most popular answer for two reasons. The first is that many organizations set tighter internal SLOs than what they promise to customers (an SLA). If the service comes close to or exceeds the SLO, it serves as an early warning sign to the team, and they can address the issue before it impacts customers and results in an external action. Second, some respondents may have SLOs that primarily serve internal customers.

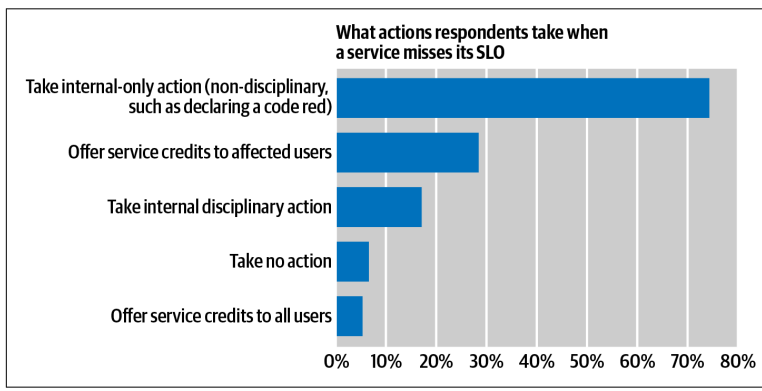


Figure 2-18. What actions respondents take when a service misses its SLO

13 Data calculated based on the 46% of respondents who have SLOs. Data excludes the 10% who have SLOs but did not select one of the options provided. Respondents were able to select more than one answer.

SLO Reviews Are Underutilized by the Majority of Respondents

Once organizations establish their SLOs, the majority (54%) do not review them regularly. Forty-six percent rarely evaluate their SLOs, doing so only as needed or as requested. Nine percent never review their SLOs. Of those who review their SLOs more frequently, 32% do so once a quarter, and 14% do so monthly or more frequently.[14] SLOs should not be static. They should evolve as your services and business evolve to ensure that you are measuring the most important aspects of your service.

What prompts reviews? Figure 2-19 shows that scheduled periodic reviews is the top reason why respondents evaluate their SLOs. One-quarter of the respondents say they review SLOs when management demands it.

Although organizations should develop customer-centric (internal or external) SLOs, user behavior drives only 20% of SLO reviews. Error budgets also influence when respondents assess their SLOs. Twelve percent review their SLOs after exceeding their error budgets.[15]

Figure 2-19. Reasons prompting organizations to review SLOs

14 Data in this section represents only those respondents who answered that they use SLOs.

15 Eight percent of the respondents selected "Other" as a reason for what prompts a review of their SLOs.

Availability Is the Top SLI Measurement

We recommend that organizations base their SLO targets on SLI measurements, so we also asked respondents what SLIs they use.[16] Availability is the top answer, with 87% using it as an SLI, followed by request latency (69%) and error rate (65%). Half of the respondents use system throughput as an SLI metric. Durability (the likelihood that data will be retained over a long period of time) is the least-used SLI type, with only 20% using it to measure their system.

We looked at whether the length of time an organization has had SLOs correlates with the SLIs it uses. Figure 2-20 shows the length of time organizations have used SLOs for all respondents who chose that option. Interestingly, those in the first year using SLOs are more likely to adopt each of the SLI options than the other groups are.

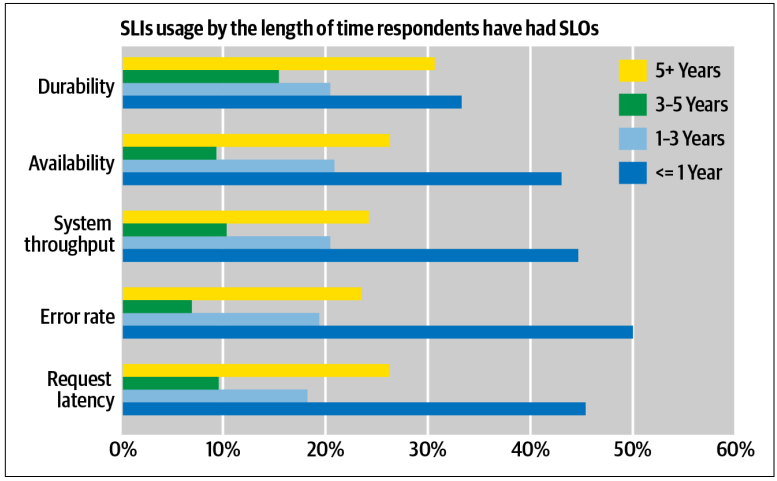


Figure 2-20. SLIs usage by the length of time respondents have had SLOs

16 Data calculated based on the number of respondents who have SLOs. Data excludes the 10% who have SLOs but did not select one of the options provided. Respondents were able to select more than one answer.

Summary

The survey data suggests that reliability is becoming a priority for many organizations. Many enterprises are just beginning to integrate SRE into their practices and are implementing SLOs to measure their services. Service reliability must be in the forefront of your mind; otherwise you risk losing users and customers, or you risk the reputational impact of a very public outage. SRE practices will make your service more reliable, and SLOs are the measurement tool that will help you move toward the level of reliability that will make your customers happy.

In the following chapters, we will provide guidance on how to implement SLOs and SLIs, and on how to use error budgets to measure availability without sacrificing feature release velocity.

CHAPTER 3
Selecting SLOs

SLOs are well-defined, concrete targets for system availability. They represent a dividing line between user happiness and unhappiness, and they frame all discussions about whether the system is running reliably as perceived by users. The *SLO Adoption and Usage Survey* reveals that, despite the importance of SLOs in ensuring the reliability of services, many organizations have not implemented SLOs. The survey also shows that organizations using SLOs fail to regularly update them as their businesses evolve. These are missed opportunities that can keep organizations from gaining all of the benefits SRE offers.

Although the survey did not delve into the reasons why respondents do not leverage SLOs and SLIs, we speculate that it is because defining SLOs and SLIs is a difficult task and many do not know where to start. In addition, our experience with customers suggests that teams may lack executive support—a critical component in SLO definition, alignment, and success. The remainder of this report provides a step-by-step guide for building SLOs and SLIs, and describes how to apply them to error budgets so that your organization can use this data to make business decisions that drive feature release velocity that is in balance with the availability appropriate for your business and customer needs.

This chapter and Chapter 4 describe how to set SLOs using the following three steps:

1. Define desired objectives or the services you want to cover with SLOs
2. Determine how to measure SLOs using SLIs
3. Once you have SLI measurements in place, set the precise numerical thresholds for SLOs

This chapter first reviews the importance of SLOs and describes how you can use them to make business decisions. Then, it will cover the characteristics of impactful SLOs and the basic principles underlying SLOs, which will help you determine which SLOs to establish.

Do Not Let Perfect Be the Enemy of Good

Before we dive into SLOs and SLIs, we want to be straightforward and tell you that it's unlikely that you will get your SLOs and SLIs right on the first try. But, that's okay! Developing SLIs and SLOs is not science. There is no one-size-fits-all approach that you can apply to your organization or service. That's why you have to use the guidance in this report, the *SRE Book*, the *SRE Workbook*, and other resources[1] to establish the best initial set of measurements you can. From there, iterate and adapt your SLIs and SLOs as you learn more about your customer experience and system performance. Your SLOs and SLIs will scale as your service and SRE practices scale.

It is very easy to get caught up in the intricacies of your service and prolong the implementation of your SLOs and SLIs. Rather than dragging the process out, just start somewhere. Pick one aspect of your service that is critical to the user journey and begin to measure it. Get the feedback loop started and go back and improve your SLOs and SLIs. And if (when) you get it wrong, don't worry; your users will let you know. Ultimately, user feedback will make your next SLO iteration stronger.

When the process becomes cumbersome or complicated, simply ask three questions:

1 Special thank you to the authors of many CRE Life Lessons blog posts and the developers of our online course, Site Reliability Engineering: Measuring and Managing Reliability, that informed much of the content in this chapter and in the next two chapters on SLIs and error budgets.

- Who are my users?
- What do they want/expect from the system?
- What level of reliability will they be happy with?

Come back to these three questions if or when you find yourself going down a rabbit hole of SLO/SLI possibilities or arguments.

SLOs: What They Are and Why We Have Them

Let's quickly recap how SLOs facilitate business decision making.

SLOs are numerical thresholds for system availability and reliability. With SLOs, reliability is no longer a fuzzy concept defined differently by product, engineering, and business teams. If you meet your SLOs, users should be happy with your service. If your service drops below the SLO, the user experience suffers.

SLOs should reflect your business goals so that you can use SLO data to measure progress toward your goals. The data informs conversations about service design, architecture, and feature releases. Well-thought-out SLOs help businesses answer the following three important questions that ultimately drive business outcomes:

- How do we prioritize reliability versus other features?
- Can we release new features and risk breaking the system without significantly impacting the user's experience?
- How do we weigh operational versus project work?

How Do We Prioritize Reliability Versus Other Features?

Because SLOs are precise numerical values, there is no longer any ambiguity about whether your system is reliable. SLOs become a common language and shared understanding among various parts of your organization. The conversation becomes focused on concrete data rather than a vague definition of *reliability*.

Most businesses acquire users and enhance profitability by releasing new features. However, you must balance feature release velocity with maintaining the reliability of your system. If your system is not available, users leave and revenue (or other impact measures, like engagement) declines. SRE methodology looks at reliability (as

measured by an SLO) as a feature of the service. Because reliability and the impact of an unreliable system on user happiness is now quantifiable, executives and product teams can prioritize reliability as they would any other feature. SLOs are a mechanism for balancing reliability and innovation, and they guide the business strategy for application development and operation.

Can We Release New Features and Risk Breaking the System Without Significantly Impacting the User's Experience?

SLOs help you mitigate risk and keep current customers while allowing more rapid feature releases that attract new customers. As exciting as new feature releases can be, they often take a toll on system reliability. By measuring SLOs, you have an indication of the reliability cost of new features. Understanding this cost allows you to balance the potential threat to reliability that a change to your system may incur with your organization's business goals/directives to release new features.

Using SLOs allows you to take an informed and strategic view toward feature delivery versus reliability work. When you are meeting your SLOs, you can increase feature release velocity without impacting user happiness. The opposite is also true. If you continuously break the SLOs, all stakeholders must agree to curb nonreliability feature releases, while you focus on building reliability features and pay down technical debt. As your SLOs and SLIs mature, you can proactively predict and manage how changes will adversely impact your system.

How Do We Weigh Operational Versus Project Work?

Often, operational overload (e.g., firefighting, incident response, upkeep tasks) and reactive responses to outages occur because product managers or leadership do not have two key pieces of information that would convince them to prioritize engineering work:

- Hard evidence (data) that gives them a reason to slow or stop feature release and fix reliability issues
- A well-defined and readily apparent cost associated with unreliability (i.e., the revenue implications associated with user unhappiness)

SLOs draw a clear line in the sand that defines the right level of reliability for the system—either you're performing above, at, or below them. When measuring performance against your SLOs, deciding when to invest in operational responses versus project work becomes a data-driven decision.

To factor SLOs into decision making, you must be sure that you are measuring the most essential aspects of your system as perceived by your users. The next section gives you an overview of what constitutes a solid SLO.

Characteristics of Meaningful SLOs

SLOs will be unique to each organization because they reflect their own users' experiences and their business goals. Each organization must select and calibrate their targets based on the unique characteristics of their system and objectives. Although we cannot tell you what your SLOs should look like, we can tell you about common features we see in impactful SLOs. Knowing the features of strong SLOs will help you determine the parts of your service that are best to measure with an SLO.

User-Centric

You should have enough SLOs to measure the attributes of your system that are critical to your user experience. Remember, SLOs are a target level of reliability for your users. User happiness matters the most because happy users will continue to use your service, which drives revenue for the business. As such, define SLOs for aspects of your service that are critical to your users' journey.

Challenging but Not Too Challenging

SLOs can drive the daily tasks and priorities of your teams. They clear up uncertainty about whether you need to work on an issue. If you're missing the SLOs and exhausting your error budget, you must take action. If not, the team can work on other tasks.

Selecting targets that protect reliability but that are not any higher than necessary protects your flexibility to change things in the future, including trade-offs against reliability (i.e., development velocity).

Specific yet Simple

Explicitly state the scope of your SLOs, what they cover (i.e., what queries or data objects), and the conditions under which they are valid. Consider whether you will include invalid user requests as errors, and think about what will happen when a single client spams you with many requests.

At the same time, keep SLOs as simple as possible. Your system is undoubtedly complex, and you can spin your wheels debating the merits of various SLOs. Remember, focus SLOs, especially your first few, on critical operations. Gain experience with just a few SLOs, measure them, and modify them over time.

Shared Sense of SLO Ownership

In organizations with successful SLO cultures (which develop engineering processes that lead to a reliably better customer experience), developers believe they have a shared responsibility to make the service reliable, and operations teams feel that they have a responsibility to get new features out to users as quickly as possible.

Both teams must agree on the SLOs, and they must recognize that the SLOs are valuable, are an accurate measure of user experience, and need to be defended through trade-offs.[2] If there is

2 Robert van Gent and Stephen Thorne, "Building Good SLOs: CRE Life Lessons," Google Cloud Platform, October 23, 2017, *https://cloud.google.com/blog/products/gcp/building-good-slos-cre-life-lessons*.

disagreement on these points, you will lose the benefits of using SLOs as a primary driver for decision making.

Best Practices for SLO Selection

SLOs have product and business implications, so selecting targets is not just a technical activity. Product teams, Site Reliability Engineers, and developers will all weigh in on the services that should have SLOs. Consider the following tips and guidelines for selecting services for your initial SLOs.

Avoid 100% Targets and Absolutes

It is unrealistic and undesirable to set SLOs at 100%. Doing so will slow innovation. The number one reason for outages is changes to the system, such as pushing new features. If you set a 100% target for reliability, you will never be able to update your service—either with new features or with essential updates like security patches. Maintaining unnecessarily high reliability is expensive, and, in many cases, it will not increase user happiness.

SLOs that contain absolutes are also unrealistic targets. For example, we would not set an SLO for a system that requires it to always be available while loading infinitely without any latency increase. Building such systems takes a long time, and they are expensive, if not impossible, to operate.

Base SLOs on Current Performance if You Have Nothing Else

Typically, we advise that you do not choose SLOs based on current performance (unless your current state perfectly meets all of your customer and business goals). Basing SLOs on current performance typically means that you are just adopting the values of your system in its current state rather than constructing SLOs that measure user happiness and reflect and align with your business goals.

However, if you do not have any other information, it is okay for your starter SLOs to be based on current performance. Just be sure to have a plan to revisit, evaluate, and improve SLOs once you begin collecting measurements.

Group SLOs by User Experience

Rather than grouping SLO queries by product elements or by internal implementation details, categorize them by user experience. For example, group direct responses to similar user actions, such as all HTTP GET requests, into a single SLO for static content, regardless of URI. Create another SLO for background or ancillary responses. You could also establish an SLO for "read" operations and a different SLO for lower volume but more important "write" operations. In these cases, the "read" and "write" SLOs will likely have different availability and latency targets.

Develop More Than One Target for Some Services

SLOs should state how they are measured and the conditions under which they are valid. For maximum clarity, you may need to define multiple grades of SLOs for some types of SLIs. It is absolutely appropriate to set multiple targets for a latency SLO to capture the distribution more effectively. It's common to set one target for the bulk of your users and another for the long tail to make sure that it doesn't get too long. Having two different targets allows you to assess the shape of performance curves.

For example:

- 90% of GET RPC calls will complete in less than 1 ms over a rolling 30-day window
- 99% of GET RPC calls will complete in less than 10 ms over a rolling 30-day window

Some organizations may need to define separate objectives for different classes of workload. If your service involves a bulk processing pipeline that cares about throughput and an interactive client that cares about latency, you may have the following SLOs:

- 95% of throughput clients' SetRPC calls will complete in <1 s.
- 99% of latency clients' SetRPC calls with payloads <1 kB will complete in <10 ms.

Give Yourself a Buffer

Set a tighter internal SLO than what you promise to users. This buffer acts as an early warning system, giving you time to address an issue before it impacts users. A safety margin is important, especially if your service level agreements (SLAs) state that you will give customers free credits or services if you miss the SLO listed in the SLA.

Advertising a less restrictive SLO to users also gives you room to accommodate reimplementations that trade performance for other attributes, such as cost or ease of maintenance, without disappointing users.

Have a Plan to Iterate

Your SLOs (and SLIs) are not static measurements of your system that never change. Your first SLIs and SLOs will look very different from the ones you have six months, one year, or five years from now. Because your business goals inform SLOs, you should update reliability targets as your business objectives and constraints change. Do not get stuck trying to make your first (or fifth!) attempt perfect. Instead, think of SLOs and SLIs as a constantly evolving lens through which to view your system.

Summary

There's no question that defining SLOs is a large undertaking. It would certainly be a lot easier if you could simply plug in a ready-made set of goals and targets, but then SLOs would lose their value. Only after you match your SLO approach to your business can you apply formulaic methods for SLOs and SLIs. We hope that after reading this chapter you see the benefits of developing even rudimentary SLOs and updating them as your organization gains more experience with SRE practices.

The next chapter details how to identify and measure SLIs and how to use SLIs to derive the numerical value you will set as an SLO.

CHAPTER 4
Constructing SLIs to Inform SLOs

Once you choose the service(s) you want to measure, you can then think about the SLIs you will use to measure users' common tasks and critical activities. In our experience, choosing SLIs that represent the customer's experience and obtaining accurate SLI measurements are two of the most difficult tasks that organizations undertake on their SRE journeys.

The *SLO Adoption and Usage Survey* found that most organizations (87%) use availability as an SLI. Although availability is an important SLI, it should not be the only SLI you use to measure the reliability of your service. Request latency and error rate are also important metrics indicative of system health. Depending on your service, durability and system throughput should also be considered as metrics.

We are encouraged to see that organizations that adopted SLOs within the last year were the most likely to implement all types of SLI metrics. The industry has long relied on "uptime" as the measure of reliability when, in fact, measuring only time "up" or "down" obscures many important details. Relying on uptime as the measure of reliability is particularly problematic in distributed and cloud computing environments, where systems are usually not binary "up" or "down," and outages are the result of partial degradation with symptoms that are more diverse than "not working."

We hope that the information provided in this chapter will help organizations select the best SLIs for their services. We will discuss what SLIs are, how they measure user happiness, how to build SLIs,

considerations for choosing measurement strategies, and, finally, how to use SLIs to set SLO targets.

Although the process for identifying, measuring, and monitoring SLIs may seem daunting, keep in mind that having an imperfect SLI is better than no SLI. As your SLI and SLO practices mature, you can build more sophisticated SLIs that more closely correlate with end-user problems.

Defining SLIs

SLIs are quantifiable metrics that measure an approximation of a customer's experience using your service. Common SLIs include the following:

Availability
: For what fraction of events is the service usable by end users?

Request latency
: How long does it take to respond to a request?

Error rate
: How often does an error occur (typically displayed as a fraction of all requests received)?

Throughput
: How much information can the system process (often measured in requests or bytes per second)?

Durability
: How likely is it that your system will retain data over a period of time?

SLIs tell you whether you are in or out of compliance with your SLO targets and are therefore in danger of making users unhappy. For example, an SLO target may be that 99.95% of requests will be served within 400 ms in the previous four weeks. The SLI measures your performance against the SLO. If your SLI shows that only 95% of requests were served within 400 ms in the past four weeks, then you missed your SLO. If you continue to miss your SLO, your user experience suffers, and you know that you must take action to bring the SLI back into compliance with the SLO.

SLIs Are Metrics to Deliver User Happiness

How do you quantify user happiness? It's not easy to measure directly in our systems, but we can look for signals in the user journey. You may experience an outage or other problem that internally seems relatively small, but your users take to Twitter in droves and express their displeasure. Or, you may have a catastrophic event but receive few or no complaints from end users. It is impossible to get inside your users' heads and see whether they are happy or not while using your service. SLIs specify, measure, and track user journey success.

The key to selecting meaningful SLIs is to measure reliability from the user's perspective, not your perspective. For example, if your website loads slowly for users, they do not care whether your database went down or your load balancer sent requests to bad backends. All the user thinks is, "I am not happy because the website loads too slowly." SLIs quantify the user's complaint that the website is slow. When you understand at what point "the website loads too slowly" impacts user happiness, you can use the data to enhance the customer's experience.

Specific SLIs that closely represent end-user issues will identify where you should improve the user experience. An ideal SLI is a close to real-time metric expressed as a percentage from 0% to 100%.

With this SLI framework in place, a well-designed SLI should do the following:

- Increase when customers become happier
- Decrease when your customers become displeased
- Show low variance during normal operations
- Demonstrate very different measurements during outages versus normal operations[1]

This predictable and linear relationship between your SLIs and user happiness is critical because you will use these indicators to make

1 Adrian Hilton and Yaniv Aknin, "Tune Up Your SLI Metrics: CRE Life Lessons," Google Cloud Platform, January 30, 2019, *https://cloud.google.com/blog/products/management-tools/tune-up-your-sli-metrics-cre-life-lessons*.

engineering decisions. If your SLI value falls below your target level for a specified period of time, you know user happiness is suffering, and you should likely dedicate resources to restoring reliability to your service.

Common SLI Types

Choosing SLIs that successfully quantify aspects of the user journey can seem complex, but we have found that most users' interactions can be collapsed down and mapped to recommended SLI types. Refer to these high-level guidelines as you start thinking about how to measure different aspects of the user's journey.

Requests and Response

If your service responds to a user's request, then you should measure how quickly the response occurs and how many responses are successful. If your service relieves excess load by downgrading the quality of the response, you should also measure how often that occurs.

Data Processing

If your service processes data, then users probably have expectations regarding the time it takes to crunch the data. They probably also count on the accuracy of the data returned. SLIs that quantify these interactions include freshness and correctness of the processed data, and the coverage and throughput of the pipeline performing the processing.

Storage

If your service stores data for users, then they expect that they can access the data. To quantify this action, measure the durability of your storage layer.

Table 4-1 outlines the SLIs that you will likely want to measure for various user journeys.

Table 4-1. The SLI menu

Service	SLI type
Request/response	Availability Latency Quality
Data processing	Freshness Coverage Correctness Throughput
Storage	Durability

SLI Structure

Although many numbers can function as an SLI, we like to structure SLIs as a percentage of good events versus bad events. The following is the SLI equation:

$$\text{SLI} = (\text{Good events/valid events}) \times 100\%$$

The SLI equation requires that you use only valid events (not all events). When developing SLIs and determining how to collect and measure them, you may want to identify and exclude certain events so that they do not consume your error budget. There should be only a few exclusions. For example, if your system services requests over HTTPs, you may determine validity by request parameters (e.g., hostname or request path) to scope the SLI to a set of response handlers that exercise a specific user-critical code path.

Standardize SLIs

We have also found it helpful to standardize the format of indicators using this structure. A consistent SLI format eliminates the process of reasoning out the structure of SLIs each time you create a new one. All stakeholders will also have an easier time understanding SLIs if they follow a consistent format within or across services.

This ratio allows you to express SLIs as a percentage on a scale of 0%–100%. The structure is intuitive—0% means everything is broken, and 100% means everything works. This format is also easy to apply to SLO targets and error budgets.

From a practical standpoint, a uniform style simplifies writing alerting logic because you can use the same inputs: numerator,

denominator, and threshold. Apply the logic to tooling for SLO analysis, error-budget calculations, and reporting.

In addition to standardizing the structure of your SLIs, you can build a set of reusable SLI templates for common metrics. Features that fit into the standard definition templates can be omitted from the specification of an SLI. For example:

- Aggregation intervals: "Averaged over 1 minute"
- Data-access latency: "Time to last byte"
- How frequently measurements are made: "Every 10 seconds"

Aggregate Measurements

Once you have your monitoring strategy set, you must consider how to view the data. We recommend that you aggregate raw measurements. Most SLIs are best expressed as a distribution rather than an average. Consider request latencies. It is possible for most requests to be fast but for a long tail of requests to be very slow. Averaging all requests obscures the tail latencies and the changes in those latencies.

Use percentiles for SLIs so that you can see the distribution and its varying characteristics. For example, a high-order percentile (e.g., 99th or 99.9th) shows you a worst-case value. Using the 50th percentile (the median) shows the typical case. Consider our latency example: the higher the variance in response time, the more the typical user experience is affected by long-tail behavior.

Developing SLIs

Now that you understand the purpose and properties of SLIs, you can formulate indicators. The following steps guide the development of your SLIs.

First, select an application for which you want to establish an SLO. You're not setting the SLO target yet. Simply choose an application that should have an SLO.

Clearly define the users of this service and identify the common tasks and critical activities they perform when interacting with your service. These are often "user journeys" defined during product or

feature creation. These are the people and the interactions whose happiness you want to maximize. Refer to the user journeys defined during product or feature creation when taking this step toward defining SLIs.

Draw the architecture of your system, showing key components, request flows, data flows, and critical dependencies. As you abstract your system, consider grouping your components into the following categories:

Request-driven
: The user performs an action and expects a response (e.g., a user interacts with an API for a mobile application).

Pipeline
: A process in which the system takes an input, alters it in some way, and puts the output elsewhere. Pipelines can range from single instances that process in real time to multistage batch processes that take several hours.

Storage
: Systems that receive and store data that users can access again in the future.

With your system mapped out and your components identified and grouped together, the next step is to choose SLIs that will measure aspects of the user's experience. If this is your first time selecting SLIs, pick an SLI that is most relevant to the user experience and is easy to measure. Expect some SLIs to overlap. Choose five or fewer SLI types that measure the most important functions for customers.

Finally, review the diagram and determine the SLIs that would measure the user's experience. As you formulate SLIs, it helps to think of them as having two parts: SLI specification and SLI implementation.

SLI Specifications and SLI Implementations

Breaking SLIs into SLI specifications and SLI implementations is a great way to approach SLI development.

First, articulate the *SLI specification*. This is the assessment of service outcome that you believe users care about. At this point, do not consider how you will measure it. Focus on what users care about.

Refer back to Table 4-1, "The SLI Menu," to figure out what types of SLIs you want to use to measure your journey.

SLI specifications will be fairly high-level and general, for example, the ratio of home page requests that load in < 100 ms.

Then consider the *SLI implementation*—how you will measure the SLI specification. Make SLI implementations very detailed. They should be specific enough that someone can write monitoring configurations or software to measure the SLIs. Well-defined SLIs describe the following in detail:

- What events you are measuring, including any units
- Where you are measuring the SLI specification
- What attributes of the monitoring metrics are included and excluded or any validity restrictions that scope the SLI to a subset of events
- What makes an event good

When considering how to implement your SLI measurements, choose data that is easy to gather. For example, rather than taking weeks to set up probes, use your web server logs if they are readily available.

You may have several possible SLI implementations for an SLI specification. You will have to weigh how well they reflect the customer's experience (quality) versus how many customers' experiences they encompass (coverage) versus cost.

Infrastructure considerations for SLI implementations

After deciding on your SLI implementations, assess how your infrastructure serves users' interactions with your service. SLIs should have a close, predictable relationship with your user's experience, so choose SLIs that directly measure the performance of your service against the user's expectations, or in as close proximity to these expectations as possible. Often, you will have to measure a proxy because the most direct or relevant measure is hard to gather or interpret. For example, it is often difficult to measure client-side latency even though it is the most direct SLI. You may be able to measure only latency from the server side, but, if possible, measure related latencies in many locations to help surface problems in the request chain.

Also consider how that infrastructure could fail and how those failures will affect your implementations. Identify failure modes that your SLIs will not capture, and document them. Revise your SLI implementation if it will not capture high-probability or high-risk failures. You may have to change your measurement strategy or supplement it with a second one. For more details on measurement strategies, see Ways of measuring SLIs.

When determining SLI implementations, you will have to weigh the pros and cons of the many options you will have to choose from. The following section details what you may want to consider when selecting SLI types for tracking reliability.

Tracking Reliability with SLIs

Let's look at how we determine SLIs for availability, latency, and quality to track the reliability of a request response interaction in a user journey.

Availability

Availability is a critical SLI for systems that serve interactive requests from users. If your system does not successfully respond to requests, it is likely that users are not happy with the level of service. To measure reliability in this case, your SLI specification should be the proportion of valid requests served successfully.

Creating the SLI implementation is more difficult. You must decide the following:

- Which of the requests the system serves are valid for the SLI?
- What makes a response successful?

The role of the system and the method you choose to measure availability will inform your definition of success. As you map availability for an entire user journey, identify and measure the ways in which users may voluntarily end the journey before completion.

Measuring availability applies in many other circumstances. For example, the SLI specification to measure availability of a virtual machine may be the proportion of minutes that it was booted and accessible via SSH. In this case, creating the SLI implementation will require you to write complex logic as code and export a

Boolean-available measure to your SLO monitoring system. You can also define a similar SLI to the virtual machine based on the proportion of minutes the system was available.

Latency

Users will not be happy if a system serving interactive requests does not send timely responses. The SLI specification for a request response latency is the proportion of valid requests served faster than a threshold.

To develop the SLI implementation, you must decide the following:

- Which of the requests this system serves are valid for the SLI
- When the timer for measuring latency starts and stops

Selecting a target for what constitutes "fast enough" depends on how well your measured latency captures the user experience. Many organizations use an SLO that measures the long tail. For example, 90% of requests < 450 ms and 99% of requests < 900 ms. The relationship between latency and user happiness tends to be an S curve, so you can better quantify user happiness by setting other thresholds that target latency for 75–90%.

Latency is also an important reliability measure for tracking data processing. For example, if your batch processing pipeline runs daily, it probably should take less than a day to complete. The SLI implementation should reflect the time it takes to complete a task that a user queued because that directly affects their experience.

Quality

Creating quality SLIs is important if your system trades off quality of responses returned to users with another aspect of the service, such as memory utilization. The SLI specification for request response quality is the proportion of valid requests served without degrading quality.

To build the SLI implementation, you must decide the following:

- Which requests served by the system are valid for the SLI
- How you determine whether the response was served with degraded quality

Most systems have the ability to mark responses as degraded or to count such instances. This ability makes it easy to represent the SLI in terms of bad events rather than good events. If quality degradation is on a spectrum, set SLO targets at more than one point on the spectrum.

Ways to Measure SLIs

In addition to considering how well a potential SLI measures user happiness, you need to determine how and where you will measure it. We cover five strategies here for measuring SLIs. Each strategy has its own pros and cons that you must weigh when deciding how or whether to implement it. Because you want to choose SLIs that measure the user experience as closely as possible, we will cover the methods in order of proximity to the user.

Gather SLI metrics from processing server-side *request logs*. Using server-side request logs and data has several advantages. They can monitor the reliability of complicated user journeys that entail a considerable amount of request response interactions during long running sessions. Request logs are also well-suited for organizations that are establishing SLOs for the first time. You can often process request logs retroactively to backfill SLI data. Use the historical performance data to determine a baseline level of performance from which you can derive an SLI.

If SLIs require complicated logic to discern between good and bad events, you can write the logic into the code of your logs and processing jobs and export the number of good and bad events as a simple good-events counter. Counters provide the most accurate telemetry, but the downside of this approach is that building something to process logs reliably will necessitate engineering effort.

The downside to request logs is that processing will result in significant latency between an event occurring and the SLI observing it. This latency can make a log-based SLI a poor fit for triggering emergency responses. Log-based SLIs also will not observe requests that do not make it to your application servers. The same observability issue exists when you export metrics from your application services.

Although exporting metrics from stateless servers does not allow you to measure complicated, multirequest user journeys, it is easy to add *application-level metrics* (also known as whitebox metrics) that

capture the performance of individual requests. These metrics do not result in measurement latency.

Also consider your or your cloud provider's *frontend load balancing infrastructure*. This measurement takes you up a level in the stack to measure the interactions that involve users making requests that your service responds to. For most services, this will be the closest in proximity that you will come to the user experience while it is still within your control.

The upside of using frontend infrastructure metrics is that implementing should require little engineering work because your cloud provider should already have metrics and historical data easily available. Unfortunately, because load balancers are stateless, there is no way to track sessions, and cloud providers typically don't provide response data. In this case, you must ensure that metadata in the response envelope is set accurately to determine whether responses were good.

Another factor to consider is the inherent conflict of interest present—your application server exports metrics for response content, and it is responsible for generating those responses. It may not know that its responses are not good.

That's when you may choose *synthetic clients* to measure SLIs. They mimic users' interactions with your system, possibly from a point outside of your infrastructure. You can verify whether a user's journey can be completed in its entirety and whether the responses received were good. The downside is that synthetic clients are not an exact replication of user behavior. Users are often unpredictable, and accounting for outlier cases in complicated user journeys can require substantial engineering work. We recommend that you do not use synthetic clients as your only measurement.

Another drawback to this approach is that, in our experience, synthetic probes tend to be finicky, flaky beasts. Synthetic probes sometimes end up sending invalid requests due to neglect or drift from real user behavior, or, without careful tuning, they can trigger content delivery network (CDN) rate limits. One workaround is to remove synthetic outlier requests from error-budget data.

The last measurement strategy to consider is *client-side instrumentation*. Because the data comes directly from the client, it is the most accurate measure of their experience. It also allows you to gain

insights into the reliability of third parties (e.g., payment providers) in users' interactions. Although client data is the most accurate, it can create the same issues with latency as logs processing, which makes client data unsuitable for triggering a short-term operational response. You may be able to collect client-side metrics in real time via tooling such as StatsD (a popular client metrics collection tool). Collecting client-side metrics in real time may also lower the signal-to-noise ratio of the prospective SLI because this method captures many factors outside your control, such as browser or public network variations.

Use SLIs to Define SLOs

So far, we have discussed SLO fundamentals and have covered several topics related to choosing SLIs, including what makes a good SLI metric, common SLIs, and how to develop SLIs. Now, we can finally talk about how to use SLIs to set SLOs.

SLOs are a target value or range of values for a service level that is measured by an SLI, measured over a specific period of time. As discussed in the Introduction, we typically structure SLOs in the following way:

$$\text{SLI} \leq \text{target}$$

or

$$\text{Lower bound} \leq \text{SLI} \leq \text{Upper bound}$$

Your SLOs can fall into two broad categories based on how you determine them: achievable SLOs and aspirational SLOs.

Achievable SLOs

Achievable SLOs are determined by historical data. They are considered achievable because you have enough data to inform a target that you will likely meet most of the time. If you used existing metrics to build SLIs, use the historical data to select a target you will likely meet in the medium and long term.

Underlying the development of achievable SLOs is the idea that your service's past performance creates your user's current expectations. You cannot directly measure user happiness, but if your users are not complaining on social media or to customer support, chances are that your reliability target is correct. If performance levels

decline, you will miss your SLOs and will have to dedicate engineers to fix the problem.

Because achievable SLOs assume that users are happy with current performance, organizations that implement achievable SLOs must be vigilant in revisiting and reevaluating their targets in the future.[2]

Aspirational SLOs

Not all organizations have historical data on which they can base reliability targets. Other organizations may know that users are not happy with their current or past performance. Or, the opposite may be true. Your service is more reliable than users expect, affording you the opportunity to establish a less strict target and increase development velocity without impacting user happiness. In these cases, you can establish an aspirational SLO. Business requirements and goals drive the creation of aspirational SLOs.

If you have no historical data, collaborate with your product team to develop a best guess about what will maintain user happiness. Begin measuring your SLIs and gather performance data over a few measurement windows before setting your initial targets. You can also estimate SLOs based on your business needs and existing indicators of user happiness. Taking an educated guess at a reasonable target, measuring it, and refining it over time is better than getting it right the first time.

Because aspirational SLOs are based on something you are trying to achieve, expect to miss them at first and to redefine them as you gather data.

Determine a Time Window for Measuring SLOs

You must apply a time interval to your SLOs. There are two types of time windows: rolling windows (e.g., 30 days from a certain delivery date) and a calendar window (e.g., January 1–31).

2 For more information on this topic, refer to Theo Schlossnagle (@postwait), "If I could share just one thing about SLOs to frame your appreciation for them as well as your discipline around them, it would be the content on this slide," Twitter, January 9, 2020, 11:50 a.m., *https://twitter.com/postwait/status/1215345069668085761*.

Choosing a time window for measuring your SLOs can be difficult because there are many factors to consider. Let's look at each time interval in more detail.

Rolling windows better align with the experience of your users. If implementing a rolling window, define the period as an integral number of weeks so that it always contains the same number of weekends. Otherwise, your SLIs may vary for unimportant reasons if traffic on the weekend differs greatly from traffic during the week.

Calendar windows align better with business planning. To choose the correct measurement interval, you must consider whether you want to use the data to make decisions more quickly (shorter time frames) or to make strategic decisions that benefit from data collected over a longer period of time.

In our experience, four-week rolling windows work well as a general purpose window. To accommodate quick decision making, we send out weekly summaries that help to prioritize tasks. We then roll up the reports into quarterly summaries, which management uses for strategic project planning.

We recommend defining an SLO with only one time window, rather than different audiences having their own view; this encourages harmony between your development and operations teams. But, you might use that same data recalcuated for different time horizons to derive additional metrics useful for certain stakeholders, like on-call engineers (five minutes for on-call response) or executives (quarterly during a business review).

SLO Examples for Availability and Latency

Most organizations set an SLO for both availability and latency, so let's look at examples of SLOs you may set for these SLIs.

Availability SLOs answer the question, Was the service available to our user? To formulate the SLO, tally the failures and known missed requests and record errors from the first point in your control. Report the measurements as a percentage. The following is an example of an availability SLO:

> *Availability:* Node.js will respond with a non-500 response code for browser pageviews for at least 99.95% of requests in the month.

or

> *Availability:* Node.js will respond with a non-503 for mobile API calls for at least 99.9% of requests in the month.

Requests that take longer than 30 seconds (or 60 seconds for mobile) count against your availability SLO because the service may have been down.

A latency SLO measures how quickly a service performed for users. To calculate a latency SLO, count the number of queries slower than a threshold and report them as a percentage of total queries. The following is an example of a latency SLO example:

> *Latency:* Node.js will respond within 250 ms for at least 50% of requests in the month and within 3000 ms for at least 99% of requests in the month.

Iterating and Improving SLOs

SLOs should evolve as your system or user journeys evolve. Over time, your system changes, and your current SLOs may not cover new features or new user expectations. Organizations should plan to review their SLO and SLI definitions after a few months and modify them to reflect the current status of your system and user experience.

The *SLO Adoption and Usage Survey* finds that the majority (54%) of respondents with SLOs do not regularly reevaluate them and that 9% never review them. This is a major oversight by organizations. Frequent reviews are especially important when starting your SLO journey. As we have mentioned several times, the best way to implement SLOs is to start using them and to iterate rather than getting caught up in being perfect. You will find it easier to discover areas you're not covering, and gaps between SLOs and users if you have something to review.

At the start of your SLO journey, review SLOs as often as once every 1–3 months. Once you establish that the SLO is appropriate, you can reduce reviews to once every 6–12 months. Another option is to align SLO reviews with objectives and key results (OKR) goal setting. Consider all customer groups (i.e., mobile, desktop, or different geographies). Reviews should include an assessment of the SLI and all of the details of the customer groups.

Because SLOs help you maintain reliability while pursuing maximum change velocity, improving the quality of your SLOs lets you

judge this balance better, which will help you guide the strategy of your application development and operation.

Summary

SLIs are metrics that measure service performance and help you detect when your users' experience starts to suffer. Tracking SLIs gives you measurable insights that ultimately help you improve the customer's experience.

When selecting SLIs, engineers must consider when and how to measure aspects of their service that are critical to user journeys. Enhance accuracy of SLI measurements by selecting implementations that are as close to the customer as possible. You may consider the following five common strategies for measuring SLIs:

- Gathering SLI metrics from processing server-side request logs
- Using application-level metrics that capture the performance of individual requests
- Looking at your frontend load balancing infrastructure to measure interactions involving user requests to which your server responds
- Using synthetic clients to measure SLIs
- Implementing client-side instrumentation

With your SLIs identified, you can set achievable and aspirational SLOs, a target level of performance for an aspect of your service. When the SLI is above the SLO threshold, you know customers are happy. If it falls below the target, your customers are typically unhappy.

Now that you have well-defined SLOs and SLIs, you are ready for the next phase of the SRE journey—applying SLOs to error budgets. Using an SLO and error-budget approach to managing your service will unlock the full benefits of SRE methodology.

CHAPTER 5
Using Error Budgets to Manage a Service

Error budgets establish a framework within which product owners can manage innovation and product reliability. They provide an objective metric that tells you how unreliable your service can be in a given time period. By not striving for perfection or 100% reliability, there is an expectation of failure that gives product teams the room and SRE teams the ability to embrace risk.

Error budgets operationalize this concept. They incentivize both teams on finding the right balance between achieving both innovation and reliability, goals that seem to be at odds, but that aren't actually at odds in the highest performing organizations.[1] When product and SRE teams agree on the error budget, conversations about velocity versus development become strategic, collaborative, and data driven. The less ambiguous and more data-based decisions can be, the better.

The *SLO Adoption and Usage Survey* did not explore error-budget usage or practices because not all teams use them. In our experience working with organizations, we often find that many organizations implement error budgets after setting and monitoring SLOs. Our survey data seems to support this: nearly 40% of the respondents who use SLOs implemented them just in the past year, so it's likely

1 *2019 Accelerate State of DevOps Report*, DORA, Google Cloud Platform, https://cloud.google.com/devops/state-of-devops.

that many organizations are only now beginning to explore how to use error budgets to manage their services. At this level, the benefits of instituting the SRE framework become even more obvious and tangible to all stakeholders because you are now applying your metrics (SLIs) and goals (SLOs) to how you run your business.

In this chapter we explore the concept of error budgets and describe how product and SRE teams can use them to align and jointly make decisions about reliability and development velocity. We'll also discuss the importance of stakeholder buy-in and agreement as well as the steps you should take to establish an enforceable error-budget policy.

The Relationship Between SLOs and Error Budgets

As discussed in the previous section, SLOs tell you how much downtime a service can experience without significantly hurting user happiness. That allowable downtime is your error budget. In other words, the error budget is the inverse of your SLO:

$$\text{Error budget} = 1 - \text{SLO}$$

For example, your SLO may say that a service may experience 22 minutes of downtime every 30 days in order to be available 99.95% of the time. The 22 minutes is your downtime allowance, or your error budget. You can spend it any way you like over the measurement window. You just cannot overspend it.

At Google, our process for establishing an error budget is the following:

- Product management, developers, and SREs set an SLO that specifies the reliability the service should have
- The monitoring system measures the actual reliability
- The difference between the SLO and the actual reliability is the error budget or the amount of unreliability the service is allowed to have

As you can see, SLO thresholds sit between two binary states: developing new features when you have an error budget to spare and

improving service reliability when you have no error budget to spare.[2]

As long as the service performs above the SLO (i.e., there is error budget remaining), the product owners can spend the error budget on new feature releases or anything else they wish. But, if they exceed or come close to exceeding the error budget, they need to halt or slow feature releases and take action to restore reliability to the system.

When developing SLOs, leadership, SRE, and product teams must set a reasonable target that gives enough error budget to protect reliability while also accounting for the risk the business is willing to take in pursuing development velocity. For example, if your SLO says that 99.9% of requests should be successful in a given quarter, your error budget is 0.1%. If you look at this in terms of time, this equates to 43 minutes of downtime per month. This is just enough time for monitoring systems to surface issues and for a human to address them, allowing for just one incident per month, roughly. That's not a lot of time, so you may consider relaxing the SLO.

The next section discusses how to use error budgets to strike a balance between service reliability and engineering practices.

Negotiating Technical Work Versus Development Velocity

Because SRE and product development jointly own and manage the error budget, they resolve tensions between making technical improvements that increase system reliability and building or releasing new features.

Discussions about release velocity and engineering work to ensure availability focus on concrete numbers—the SLOs and the remaining error budget. For example, if the development team wants to skip testing to push new code more quickly and SREs are resistant, the error budget guides the decision.

2 Alex Bramley, "Consequences of SLO Violations: CRE Life Lessons," Google Cloud Platform, January 3, 2018, *https://cloud.google.com/blog/products/gcp/consequences-of-slo-violations-cre-life-lessons*.

Most organizations use a straightforward control loop for managing release velocity: releases can continue if the system meets the SLOs. If the system breaks the SLOs and spends down the error budget, feature releases are temporarily slowed or suspended. Resources are directed to testing and improving the system.

Product teams can manage risks by deciding what to spend the error budget on. When there is a substantial error budget, they can deliver more features, thereby taking more risks. When the error budget is low, they are likely to be more conservative and ask for additional testing and oversight because they do not want to risk exceeding the error budget and delaying the launch.

Error budgets encourage SREs to set realistic reliability goals that benefit the end user while allowing for innovation. For example, you may find that increasing reliability incrementally will cost you 15 times more in effort than the previous increment. You must consider such factors when setting your SLOs. The difference between 99.9%, 99.99%, and 99.999% targets can be negligible to users but significant in engineering work and other costs. In such a case, you can pursue development velocity rather than service reliability.

There are times when the error budget is close to being maxed out or the SREs believe a certain change to the system will consume a significant portion of the error budget. In these cases, SREs and product managers must collaborate and engage in trade-offs to protect both reliability and the rate of releases. For example, a trade-off may include implementing better integration tests and an automated canary analysis and rollback in order to keep the error-budget burn within the SLO when trying to push more quickly.

The error budget may also surface SLOs that are too strict and, as a result, slow the pace of innovation. In this case, the team may weigh increasing the error budget by setting a less-restrictive SLO with the impact on the user.

Error budgets align product teams and SREs on incentives, which removes tensions between teams and makes it easier to decide the rate of releases versus production risk. However, you will realize this alignment on incentives only if all teams and leadership support an error-budget culture. The following section discusses how to gather buy-in on your obligations to meet SLOs and enforce your error budget.

Stakeholder Buy-in and Establishing an Error-Budget Policy

The examples in the previous section illustrate how you can use SLOs and error budgets to make business decisions regarding reliability and development velocity. To have effective discussions about risk versus reliability, you must have buy-in from all levels of the organization to use SLOs and error budgets to drive decision making.

The idea of comanaging the error budget is critical. Every team must feel a sense of shared ownership in managing the error budget, and leadership must play a role in informing the error budget according to what's important from a business perspective. The operations team must have the authority to stop feature launches when there is no error budget left to spend. SREs need to feel like they have a stake and role in facilitating feature release and innovation, which is what the product team wants. Developers need to feel like they have a role in making the service reliable, which is what the SRE team wants.

To establish a culture in which all of these seemingly disparate goals align, the following must be in place:

- The organization has SLOs that all stakeholders agree are thresholds for ensuring user happiness
- The individuals responsible for defining the SLOs agree that it is possible to meet the SLO under normal circumstances, and not by burning teams out
- The organization is committed to using the error budget when making decisions and prioritizing work
- There is a process in place for refining the SLO
- There is a documented process for how to make decisions when the error budget is exhausted
- The management team or executives must support all of these points

Formalize the use of error budgets and emphasize commitment to these principles by establishing an error-budget policy (see a Sample Error-Budget Policy). A written error-budget policy guides enforcement decisions and should cover the specific actions that will be

taken when a service has consumed its entire error budget. It should also indicate who will take these actions. Common actions include stopping feature launches until the service falls within the SLO again or dedicating engineers to work on reliability-related bugs.

As an added benefit, developing the policy and getting approval for the error budget from all stakeholders—product managers, development team, and SREs—is a good test for the efficacy of your SLOs. You will need to adapt your SLOs and SLIs if all three parties cannot agree to enforce the error-budget policy.

If SREs believe the only way to defend the SLO is with heroic efforts and lots of toil, they can make the case for loosening the SLO or acquiring additional resources. If the development team and product manager feel that the resources required to fix reliability issues will slow release velocity to an unacceptable level, they can also ask to relax the SLO. If the product manager thinks the SLO will lead to a negative experience for a significant number of users before the error budget triggers a fix, they can argue that the SLO is not tight enough.

If the service exhausts the entirety of the error budget, but not all stakeholders agree that activating the stipulations in the error-budget policy is appropriate, then you must go back and review and possibly get reapproval on your policy to ensure all parties will uphold it.

Your published error-budget policy should list policy authors, reviewers, and approvers and the date on which it was approved. To ensure you're constantly improving your SLO program, also include the date of the next policy review. Give the reader context by providing a description of the service, and if the reader is not familiar with SRE concepts, also include an overview of error budgets. Specifically outline the actions that should be taken if the error budget is overspent, and provide a clear escalation path if there is a disagreement about the calculation or about whether to take the agreed-upon actions.

Summary

Error budgets are integral to the SRE model. They close the gap between product development and operations teams by giving them a clear decision framework for balancing reliability with other engineering work. Error budgets eliminate debates over work prioritization and development decisions. Instead, the discussion becomes straightforward and simple:

- If there is room in the error budget, the product teams can release new features.
- If the error budget is exceeded (or is close to being exceeded), feature release slows or stops completely and focus shifts to work that restores reliability to the service.

Using an error budget allows you to make data-driven decisions to manage the reliability of your services while also enabling the high engineering velocity seen in well-performing organizations. This balance ultimately results in a better experience for your customers.

CHAPTER 6

SLO Implementation Case Studies

Google's Customer Reliability Engineering (CRE) team helps Google Cloud Platform customers implement the SRE model of service management. Underlying all SRE practices is the concept that organizations should establish reliability metrics that align with business objectives. The CRE team—comprised of experienced SREs—partners with customers to develop SLOs that reflect the customers' goals and effectively measure the reliability of their services.

SLOs and SLIs are powerful business tools because they give organizations a framework for quantifying reliability. The insights revealed by SLOs and SLIs allow organizations to make data-driven decisions about their services. In this chapter, we tell the story of how two very different companies—Schlumberger Limited and Evernote—implemented SLOs and used the insights gained to better manage their businesses and serve their customers.

Schlumberger's SLO Journey

Schlumberger Limited is a leading provider of technology for the oil and gas industry. The company provides customers with an innovative collaboration environment—called the DELFI cognitive exploration and production environment—that combines domain expertise and digital technologies to enable a new approach to planning and operating assets across exploration, development, production, and midstream. Schlumberger delivers DELFI via a SaaS subscription model.

The DELFI environment leverages digital technologies, including security, analytics, machine learning, high-performance computing (HPC), and Internet of Things (IoT), to improve operational efficiency and deliver optimized production at the lowest cost per barrel for customers. The openness and extensibility of the DELFI environment enable Schlumberger customers and software partners to add their own intellectual property and workflows in the environment.

The Schlumberger Software Integrated Solutions (SIS) team introduced SLOs when the company transitioned its software delivery model from packaged software on CD/DVD to a cloud-based SaaS model in May 2018. As a part of the strategic plan to become a SaaS provider, the SIS Cloud Operations organization sought a new way of building and managing DELFI software solutions. It introduced the SRE model and a central SRE team[1] to continuously improve the efficiency, scalability, and reliability of DELFI operations in its new digital delivery environment.

Why Schlumberger Implemented SRE

The oil and gas industry is rapidly embracing digital transformation. To support and serve the needs of its software customers in the fast-paced, always-on digital business environment, Schlumberger moved from a traditional packaged software model to a cloud-based solution for software delivery.

Schlumberger's new cloud infrastructure allows its product solution teams to release features and fixes at a much higher velocity than they could in the traditional pull model (via CD/DVD or software download). As the teams moved to a continuous integration/continuous delivery (CI/CD) pipeline for delivering code, they had to give careful consideration to how they would maintain their high-quality standards (i.e., service availability) in a cloud ecosystem.

1 Thank you to the following members of Schlumberger, who provided input to this case study: Shivanand Misir (SRE Manager), Kamal Jansen (VP Services and Central Delivery), Alejandro Zabala (Central Cloud Operations Manager), Fred Xu (SRE), Karan Jain (SRE), Kevin Haritmonds (SRE), Marcelo Purger (SRE), Naman Bairagi (SRE), Pardeep Sandhu (SRE), Willian Maja (SRE), Marc Valderrama (former SRE Manager), and Stephen Whitley (VP Open Data Ecosystem).

Adopting SRE methodology and implementing SLOs, SLIs, and error budgets allows solutions teams to leverage cloud technology and rapidly release changes while maintaining high levels of service reliability.

Optimizing the feedback loop

The cloud environment gives Schlumberger's executive management, product managers, and development, operations, and SRE engineers greater telemetry data for their applications. The continuous customer feedback loop is a significant advantage over the traditional packaged software model because now a significant proportion of feedback is enabled directly from the application with the potential for direct client interaction.

The game changer is Schlumberger's ability to respond rapidly with a DevOps approach to software development. It can now release in a timescale measured in a few weeks, compared to many months or sometimes years due to the previous annual release cycle necessitated by packaged software. Over time, Schlumberger expects SaaS to enable more interactive and "smart" feedback mechanisms (meaning that feedback is provided within the app, and analytics can be employed on the feedback) to be leveraged by development teams.

With SLO and SLI metrics in place, all stakeholders have a framework for gathering and monitoring performance data on the new features that Schlumberger's solutions team pushes. SLOs that cover newly released features provide a standardized definition and target for user happiness. SLIs indicate how well certain features penetrate the market. For example, the availability SLIs measure how many of the requests to the system are successful as well as how customers interact with the system. The SRE engineers and solutions teams correlate this information with new feature releases and see whether there is an uptake in the requests to those new services.

The data gleaned by monitoring SLIs and SLO performance has visibility up the management chain, giving the business more data to support its software development and enhancement decisions.

Maximizing feature release and reliability

Moving to the cloud also means that solutions teams have the capability to engage in CI/CD and release new features, fixes, and solutions more quickly than they could in the desktop model of software

delivery. Currently, many solutions teams deliver new features or enhancements directly to customers in two-week sprints after rigorous operational testing. Before transitioning to the cloud, they deployed major releases once a year and minor releases typically every three to six months.

The SRE and development teams use SLOs, SLIs, and error budgets to mitigate outages while pursuing rapid release of features or fixes, thereby maintaining high standards of software delivery quality to customers.

When services meet their SLOs, they continue to release in two-week sprints. When an SLI indicates that the SLO has not been met and the service exhausts or comes close to exhausting the error budget, the team implements a feature freeze for 30 days. In those 30 days, the solutions team has specific policies that delineate whether the feature freeze applies to the entire service or just to a particular component. Solutions teams must dedicate resources to determine the cause of the issue and decide how to improve reliability. Only reliability fixes are promoted to production and turned out in those 30 days. The 30-day feature freeze policy allows the team to address additional reliability issues from the backlog to avoid future freezes. This backlog can include fixes to software, increased telemetry (SLIs) to improve serviceability, or improvements to automation to reduce recovery time.

The SLOs, SLIs, and error budget provide a data-driven change management strategy that allows Schlumberger to pursue its goals of releasing new features and software solutions while protecting the reliability of the service.

Bridging the DevOps divide

As Schlumberger became a SaaS provider, realizing the potential of DevOps was critical to protecting the reliability of its services. SLOs and SLIs align development and operations teams on common and shared targets for site reliability. No longer could development throw code over the proverbial wall and expect operations to make it work. That dynamic increases the likelihood of outages and bugs, which causes traditional operations teams to avoid changes in order to keep the system stable. As a result, development velocity slows.

With agreed-upon metrics and comanaged error budgets, development teams have the same insights that ops teams have. The SLOs

and SLIs facilitate data-driven discussions between development and operations teams about how pushes to the system affect the customer experience.

Implementing the First SLOs for a Nonnative Cloud Application

When Schlumberger started this journey nearly 18 months ago, cloud technology and SRE were novel to Schlumberger's product teams. Schlumberger engaged the Google CRE team to learn SRE principles and how to apply them to its service management strategies.

Schlumberger's initial endeavor to define user journeys and establish SLOs presented unique challenges compared to cloud-native applications. The software solutions to be delivered on the new cloud-based DELFI environment were not new. In many cases, customers had used the software for a decade or more. The business teams already had a deep understanding of how customers used the products, but they had to discover how customers would use the software on the cloud.

Additionally, teams did not have reporting or insights on availability and latency that cloud-native applications can typically use to establish a baseline level of performance and set SLIs. Nor did they have status codes and error codes to help define how the user interacts with the system.

With these challenges in mind, the teams started by building a few straightforward SLOs and SLIs, knowing that they would iterate and develop more in the future as both existing "on-prem" and new "cloud-native" solutions became available in the cloud. These SLOs and SLIs were elaborated and tuned through the incorporation of customer feedback.

Schlumberger's SRE team engaged the business and portfolio teams to identify failure scenarios that would unquestionably lead to customer dissatisfaction. It selected the customer's user journey when accessing the DELFI Portal because it represents every customer's first experience in the DELFI environment. Based on feedback from the respective business and portfolio teams, the SRE team mapped the specific user journey for the DELFI Portal related to customer login. The teams broke the user journey down into SLIs for

request/response and data processing, and measured the availability, latency, and correctness for the DELFI Portal landing page load, user login, and user logout experience.

Definition of the SLO

A monthly uptime percentage measure that is based on customer's authorized users being able to connect to the DELFI Portal landing page, sign in (where they have appropriate permission), and then log out.

What to measure and how to measure

For each of the steps identified in the user journey, where each individual step must be successful for the overall transaction to be considered a success, the SREs instrumented both a black-box prober as well as live monitoring. The black-box prober was completely independent from the DELFI environment and simulated the user journey from various external probes across the world. Live monitoring instrumentation captured actual user metrics and experience. Both types of instrumentation measured the availability, latency, and correctness along this user journey, from which the SREs calculated the monthly uptime percentage.

Starting small provided the experience and framework for the SRE team to educate other development and operations teams in markets around the globe on how to develop SLIs and SLOs for their own services. Now that Schlumberger has migrated more solutions to the DELFI environment, it has more data that it can use to establish additional SLIs for availability, latency, throughput, and so on. The next section discusses how Schlumberger currently determines SLOs.

Adapting initial SLOs

The SRE team now works with individual solutions teams to set and establish SLOs for their services. Since establishing its first SLOs 18 months ago, Schlumberger has developed new native solutions and transitioned existing software to now have more than 20 specialist solutions available in the DELFI environment. With services running on the cloud, the SRE team iterated its existing SLOs and the way in which it determines SLIs and SLOs.

Schlumberger built tooling to monitor SLI performance against its SLOs. With this data it can look at its existing performance graphics, observe the behavior of the SLIs, and establish SLO targets that the developers and management of the application can meet.

For example, the teams improved and tightened the SLO for the entry point and the authentication point for the DELFI Portal after collecting user data and measuring the data against SLOs. It is now the strictest SLO at Schlumberger at 99.55%.

Establishing and Evolving SLOs for Products That Are Not Yet Live

As part of the transition to the SaaS model, solutions teams put services through a rigorous operational readiness framework to ensure that DELFI applications and services are ready to be made generally available to the public.

As part of completing the operational readiness process, solutions teams, in collaboration with the portfolio or business teams, define their user journeys and associated SLIs and SLOs. As the solution moves from precommercial to generally available, the solutions team and SREs continuously monitor and evaluate the status of the SLIs in relation to the SLOs over time. This includes gathering and taking into account actual user feedback.

Schlumberger engages customers who are willing to use the software in a controlled environment. During this learning phase, the solution and SRE teams determine whether the SLOs and SLIs are set at the optimal levels and reflect the user experience. They also test whether the monitoring and alerting tools work correctly and are helpful.

The teams typically set SLOs that are less strict than what they will achieve once they make the solution available to all customers. As it monitors system reliability and customer usage, the SRE team identifies aspects of the application that the development team needs to improve. As they improve the reliability of important operational aspects of the application, the SRE and product teams evaluate the SLOs monthly or every few months, depending on the work required, and often tighten them.

Example: How Having SLOs Led to a Better Customer Experience

The SRE team engaged with the business team to define the key user journeys of one of Schlumberger's solutions that runs evaluations on oil field development. As part of running the study, the user uploads a new Excel file containing all the necessary field information. Once uploaded, the software runs a simulation and gives the customer the results.

During engagements with early adopters, the team set up monitoring against several key points in the user journey, including uploading the file, running the simulation, and presenting the results. They quickly saw that the user was able to upload the file successfully only 20–30% of the time, which was much lower than their SLO target.

This wasn't a complete surprise to the development and product management teams. The product teams were anecdotally aware that customers experienced these failures because they were sitting with customers and learning how they use the new app. However, the teams never had the data to quantify the extent of the problem. The SRE team shared the performance of the SLI against the SLO, and the business agreed that it was a critical problem. It dedicated resources to fix the issue, and after some time the success rate improved and exceeded the objective. The development team is now motivated to ensure that the success rate is above the SLO at all times. The data gave the team the ammunition to prioritize this work and the ability to see the impact on customers.

Monitoring and Alerting

With SRE and solutions teams identifying SLIs and establishing SLOs, one of the objectives of the SRE team was to streamline the definition, instrumentation, collection, monitoring, and alerting of SLIs. This ensures consistency and continuity across the organization.

When Schlumberger's SRE journey started 18 months ago, the teams evaluated the tools available on the market and realized that none of them sufficiently captured the complete end-to-end workflow or had the ability to monitor and visualize the status of SLIs and SLOs over time.

As a result, the SRE team built an SRE framework that summarized the requirements at that time. All solutions teams use Stackdriver to collect metrics, and the SRE framework exports the SLI metrics to a persistent data storage medium, which calculates SLIs for the rolling 30-day average. It also includes dashboards showing SLI status against SLOs.

With collaboration and guidance from the Google CRE team, SREs adopted and evangelized Google's Stackdriver service-level monitoring across the organization. This monitoring enables Schlumberger to communicate the results to the various internal audiences, including the SREs/DevOps as well as management and business stakeholders. Using Stackdriver also provides real-time visibility into the performance of SLIs and SLOs, which facilitates consensus about service performance and provides justification for implementing feature freezes (as needed) in order to guarantee user happiness.

Evangelizing SRE and SLOs

After Cloud Operations/SIS leadership decided to implement SRE methodology as part of the strategy to migrate successfully to a SaaS offering, it needed to introduce these principles to the product and DevOps teams in its tech centers around the world.

Cloud Operations/SIS leadership traveled with SRE engineers to each software technology center and held a series of workshops introducing and educating developers and business teams on SRE concepts and how to develop and use SLOs and SLIs to manage the development of their services in a cloud environment.

After the initial educational workshops, the teams in each tech center continued learning about SRE by engaging in a book club. Every week, a different team member facilitated a discussion on a chapter from Google's *SRE Book* and *The SRE Workbook*.

To further support solutions teams in developing their SLOs and SLIs, the SRE team developed a Wiki site where it has a variety of support materials that explain the expectations for developing SLOs and SLIs as part of its operational readiness framework. It also provides guidance and examples of SLIs that the service may adopt.

What's Next

In the 18 months since Schlumberger started its SRE and SLO journey, it has taken considerable steps to establish a new culture across its teams and products. However, culture change of this magnitude takes time. Although the SRE team is dedicated to espousing and implementing SRE best practices and defending the SLOs, product and business teams are still getting comfortable with the concepts, and with applying them in practice.

The majority of major products in DELFI are commercial. Continuous feedback from customers enables engagement with the business in developing the SLOs and using the data to make decisions about product features and system architecture.

Additionally, most product managers have always operated under the notion that they cannot break the service. The concept of an error budget is a completely new way of looking at feature release versus reliability. The SRE team continues to work with product managers to integrate error budgets into decision making.

Evangelism efforts will continue as more services become available to customers on the cloud. Once the SLOs and SLIs go live, SLO and SLI data will become much more meaningful and rich. The SRE team hopes to share powerful examples of how SLOs and SLIs can guide architecture, feature release, and resource decisions. Seeing relevant examples within Schlumberger will continue to drive the organizational shift to an SRE mentality, and product owners will be more willing to use performance against SLOs to make data-driven decisions about the products that protect user happiness.

Evernote's SLO Journey

The Evernote SLO engineering case study in the *SRE Workbook* describes Evernote's adoption of the SRE model and its first nine months developing and using SLOs and SLIs. The case study in this document expands its story, providing more details on its first SLO attempts. We also pick up the story where the *SRE Workbook* left off and describe the evolution of Evernote's SRE culture and how Evernote uses SLOs as part of new feature and product planning.

Start at the Beginning

Evernote is a cross-platform app that helps individuals and teams create, assemble, and share information. Evernote has 225 million users around the world using the platform to store pieces of information, such as text-based notes, files, and images. In April 2017, Evernote started to implement SLOs.

One of the challenges that led Evernote to begin its SLO journey was conflicting objectives between its development and operations teams, which were affecting Evernote's customer experience and quality of service. Google's CRE team helped move Evernote toward an SLO-centric approach that provides both teams with a common frame of reference that they can integrate into conversations to improve business practices.

Since Evernote began its SLO journey, its use of SLOs has evolved from being a way to increase engineering velocity and maintain quality of service, to now factoring performance against SLOs into product development, resource allocation, and everyday business decisions.

Evernote Today: Transitioning to a Shared Responsibility Model

Currently, Evernote is transitioning more developers into a shared responsibility model across more teams. This is an evolution of the SRE culture.

Evernote saw that it was not possible to scale an operations organization to support the level of new feature and product release velocity it wanted while also having those new features act reliably. It wanted developers to be able to launch code without requiring a team of dedicated operations engineers to be solely responsible for ushering it into production. This required Evernote to embed the expectation in its culture that developers are required to do more than just deliver code; they're expected to provide the whole package, which includes follow-through such as monitoring, releasing, and instrumenting the code.

As a result of launching the shared responsibility model, Evernote now has a formal process for launching new features into production that helps it more easily determine which projects are ready to

be published and which will likely cause problems for the teams operating a service.

Promoting shared responsibility using SLOs

Although this transition to the shared responsibility model encompasses many different components, SLOs and SLIs play a role in promoting shared responsibility, and Evernote incorporates them into its processes for launching new features or products.

Evernote structures its teams using a zone model and aligns zone types by product or service. As part of the service development process, technical and project leads define SLOs and SLIs in their architecture documents. These documents typically also describe other important service information, such as the build and testing process, system and component architecture diagrams, and user interaction sequence diagrams.

When creating these documents, the product launch engineers provide guidance and training materials to engineers to ensure that they define primary SLIs that are already in their monitoring metrics platform, Datadog. The product launch engineers also confirm that the SLIs inform the SLOs. Engineering team leads or architects, product launch engineers, the VP of engineering, the VP of operations, and development teams ultimately sign off on the architecture documents.

Additionally, Evernote monitors performance against SLOs in the stage environment and considers this data when determining whether it's ready to launch the new feature or product. For example, if all of its priority 1 and 2 (P1 and P2) bug tickets are closed but the service does not meet the SLOs in staging, it will likely not do so in production. Chances are that the release will immediately cause customer pain. Using bug count as the sole indicator for launch readiness isn't always the best thing for customers. Evernote can quantify that pain with data and factor it into the discussion and decision about whether to launch.

The importance of supporting engineers in creating effective SLIs and SLOs

At this phase of the shared responsibility model, Evernote's developers are still learning how to manage their outcomes and run a service. Prior to Evernote's transition to the SRE model, developers weren't responsible for metrics and monitoring or for necessarily

knowing whether a program was performing well once their code was passed off. The level of openness and comfortability with SRE concepts continues to vary greatly from team to team and project to project.

Evernote's product launch engineers have been critical in facilitating the successful transition to a shared responsibility model and adoption of SRE methodologies, including SLOs and SLIs. The product launch engineers partner with engineering teams and provide support and guidance as they develop the initial SLIs and SLOs included in the architecture documents.

Reducing Inefficiencies for a Progressive Environment

The *SLO Adoption and Usage Survey* showed that most organizations do not regularly review their SLOs and SLIs. Evernote is an exception to this, and its regular reviews have been instrumental to its success and ability to avoid redundancies.

Choosing tools for reviewing data

To store SLIs and SLOs and analyze their performance against error-budget calculations, Evernote uses a tool called Datadog, which contains a widget to measure uptime/success rate SLOs. The widget feeds information into its service health dashboard, which makes level setting easier. The dashboard also clearly shows leadership how services are performing and whether targets are being met.

Evernote has also leveraged its relationship with the Google CRE team, which helped to set up systems that scrape Evernote's main monitoring SLIs and feed them into Google Stackdriver to calculate SLO burn rate. Both Evernote SRE and Google CRE receive notifications when burn rates approach certain thresholds.

In a win-win scenario, Google Cloud can use Evernote's signals to help drive its signals and health. As described in Evernote's first case study in the *SRE Workbook*, sharing the same performance dashboards with Google CRE has proven to be a very powerful way to drive customer-focused behavior by allowing for customized support from Google, rather than generic notifications.

Reviewing SLOs and SLIs for continual improvement

Initially, Evernote revisited its targets every six months by reviewing a document that contained all of its SLOs. At present, that document has been replaced with a weekly operations review and production readiness review. These processes are still being fine-tuned as more engineering teams apply the concepts in production for services they own.

All engineering managers in the company participate in Evernote's weekly operations review meeting. The scope includes SRE, software development, QA, and security.

The standing agenda includes the following:

- A discussion of Evernote's top-level, service health review dashboard, which shows the most important SLOs the company wants the department to monitor. This dashboard currently represents about a dozen SLOs, including an availability SLO for Evernote's monolithic application, as well as others for search, commerce, and note capture services.
- A summary of incidents and postmortems that impacted any SLO from the week before. Evernote practices a blameless postmortem culture and finds it valuable to share with engineering teams as a learning tool for continuous improvement. An SRE team typically creates the postmortem reports, but product development teams may prepare them in certain circumstances.
- A 15–20 minute presentation from one team that recently launched a new feature or service into production. The team describes its service-specific SLO/SLI dashboards, alerts, troubleshooting runbooks, and any recent incidents. Because this team shares its SLOs and experience, the other engineering teams learn from its work and provide feedback.

Using the review process to clean up the dark corners of the infrastructure

Evernote uses SLOs to represent customer happiness as a function of its reliability and prides itself on having a high-level focus on the experience of its customers. One of the challenges of running a monolithic application is that many of the user-facing experiences, such as user login and registration, still depend on this infrastructure. Evernote's goal is to reduce the scope of the monolithic system

by replacing functionality with new microservices on a more reliable platform running on Kubernetes.

To this aim, Evernote uses the production readiness review process to take a close look at how SLOs are implemented for these new microservices. The engineering teams are responsible for having SLOs ready prior to launching into production. After the service is live, these teams will present at the weekly operations review meeting to demonstrate their SLOs and show how successful their service has been running. By continually reviewing the status of SLOs across the service, once dark corners of the infrastructure now receive the visibility and attention required to help drive improvement to the customer product experience.

Taking the Mystery Out of Resource Allocation

By developing data-driven SLIs and SLOs to align its different teams behind common business goals, Evernote has been able to increase the accuracy with which its teams identify problems, measure the impact of those problems, and present solutions to leadership. As a result, leadership can quickly make decisions and prioritize resources based on numbers and facts, rather than nebulous employee complaints or frustrations. This alignment of goals and priorities has led to improvements in major infrastructure and overall user experience.

Quantifying the Impact of Outages on Users

Measuring the impact that outages have on users is important to be able to align your SLOs with user expectations. To understand user expectations, Evernote includes user journeys in its architecture documents, which zone teams present during their weekly architecture review forums. In the past, Evernote's product managers created these user journeys, but now they are shifting that responsibility to tech leads and architects.

Allowing user expectations to guide your targets and determine whether they are too low or too high is essential to providing a consistent experience and can help prioritize which areas the team should be focusing on at any given time to keep unreliability to a minimum.

Using SLOs to drive company priorities

Twitter revealed something interesting to Evernote about user expectations. Sometimes, when an outage Evernote considered small occurred, thousands of customers took to Twitter to express their dissatisfaction. However, during other outages that it considered very impactful, only a few users complained on Twitter. This caused Evernote to review its targets and make adjustments. The importance of being able to quantify the impact of an outage on users may be best represented by the story of one of Evernote's primary database systems, the Userstore.

The application experienced a major outage about every six months. The Userstore is a single point of failure in Evernote's infrastructure and nearly all user-facing application server functionality relies on it. As a result, each outage affected all Evernote customers across all Notestores (shards) and their ability to sync, create notes, collaborate, and so on.

During a repeating series of service incidents, every morning (or sometimes as early as 11 p.m. the night prior) the Userstore would stop responding properly to inbound Notestore requests and spent 100% of CPU time churning in kernel space. Restarting the Userstore restored the ability for the Userstore to process Notestore requests, but the service continued to oscillate through periods of instability throughout the morning peak (client) traffic until 10 a.m., at which time Evernote would drop out of peak traffic and issues subsided.

One outage burned 600% of the 30-day error budget of overall uptime for the Evernote service. The lead engineer responsible for that component spent hours restoring the service each time an outage occurred. After several occurrences, the lead engineer for the component sought to fix the problem once and for all to alleviate customer pain as well as internal burnout across teams. To do so, he needed to secure additional engineering time and looked for a quantifiable way to show leadership the impact of the outages.

Using their SLO report tooling, which included reports for error-budget burndowns, the engineer was able to demonstrate to leadership the significant impact of these outages and gain project investment to solve this class of issue once and for all. A few months later, Evernote had successfully implemented a major redesign of its Userstore architecture to incorporate read replicas. Reviewing its

SLO performance the following quarter, Evernote confirmed it had not had a single recurrence of this class of issue and its error budget remained positive throughout. Success!

Implementing initial SLIs and SLOs

As with any step toward improvement, you have to start somewhere. Rather than spending a lot of time discussing the ideal way to measure an aspect of the service and building the perfect tool to do so, Evernote chose to identify which tools and processes it had at the time that could bring its SLOs to fruition quickly. Engineers implementing SLOs decided it was better to measure something than nothing and iterate along the way. Although these tools didn't fully capture exactly what Evernote wanted to measure, it got the ball rolling toward a solution that it continues to refine today.

The first SLI that Evernote developed measured overall aggregate availability of the Evernote service as a black-box measurement from the outside. This measured the number of seconds or minutes that the application was down/up with an SLO of reaching 99.95% availability. The downside to this SLI was that Evernote lost granularity about specific features, functions, and user journeys that it felt were important, but it was a starting point.

Evernote offers different tranches of service to business and nonbusiness (consumer) users within its application by placing these two classes of users in different application-server shard cohorts. Because the availability of business shards has a proportionally higher impact on business revenue, Evernote's first SLI iteration involved differentiating measurements of the two shards. Its SLO (99.95% availability) remained the same for each shard, as well as for the lump sum measurement of the shards combined.

Evernote users are bound to a single shard for the lifetime of their account, thus the impact of an outage on a single shard is just as painful to an end user. To that end, Evernote also developed an SLO specifying that no individual shard would be unavailable for more than 10 minutes. Even with a basic black-box measurement technique for the initial SLO, it provided sufficient evidence to support resourcing the eventual solution to the Userstore outages.

Improving the fidelity of its measurements

Six months after implementing its first SLOs, Evernote decided to reevaluate its existing SLOs, and revisited its desire to increase granularity around the things it measured and reported on. It discussed creating a number of user journey–based SLOs, each using a composite set of API calls that represented the distinct journey.

After deliberation between the SRE and product engineering teams, Evernote decided not to add a new monolith SLO. There were two reasons for this decision. The first was that Evernote had dozens of important user journeys that were a part of its monolith, and managing dozens of SLOs would not have scaled with its team size. Secondly, but equally as important, Evernote realized through its aforementioned Userstore outages that its existing black-box availability SLI did not accurately represent user pain.

Its existing SLI measured availability from an external prober that hit an HTTP endpoint every minute. If it failed, it retried one time and then would return a failure if that second try failed. The problem was that the tool would not detect a failure until a minimum of 75 seconds passed, which was not ideal from a user standpoint. In the case of oscillating service availability, as was the case in some of the Userstore outages, this meant that the error budget might not be affected at all from a major issue!

To remedy this potential blindspot, Evernote revisited its initial SLI, deciding to change the way it was measuring availability. Instead of black-box requests, Evernote would use user requests as observed by its frontend proxies, changing its SLI to be expressed as a ratio of good events over total events for a given time period. The increased fidelity of the SLI enabled Evernote to better understand the impact of incidents both large and small on its users. Twitter rejoiced.

Off the charts

After deploying read replica, Evernote hasn't burned any error budget due to Userstore brownouts, the Userstore's queries-per-second (QPS) load was reduced on the master by upwards of 50%, and the application gained an additional 12–24 months of capacity runway. The SLI fidelity improvements will allow Evernote to be more reactive to perturbations on the Userstore in the future.

Without a way for the engineering teams to quantify reliability to leadership, the Userstore outages would likely still be occurring

today, resulting in internal inefficiencies and negative impacts on revenue as a consequence of frustrated users and cancelled subscriptions. By implementing SLIs and SLOs as well as continually observing, reporting on, and improving them, the lead engineer responsible for the Userstore service was able to show leadership the extremity of the infrastructure problem, create targets that aligned with leadership's goals to improve the problem, and achieve an availability level that went off the burndown chart.

Where Evernote Is Today

At one point in time, Evernote viewed the core disciplines of operations and development as separate professional tracks in which engineers can specialize. After implementing an SRE model, however, Evernote now sees the value of providing a path for software developers to move over from development positions into operations.

It's happy to report that by implementing SLOs that align these two teams by common goals, its former culture is breaking out of a split model and into a more collaborative and integrated SRE model that supports the evolution of a business as a whole by taking the guesswork out of business decisions. This shared responsibility model integrates SLIs and SLOs into business practices to promote improvement among teams, and a culture of continuous improvement.

Summary

The *SLO Adoption and Usage Survey* finds that, although respondents have adopted many SRE best practices, implementing SLOs and SLIs are the SRE practices they are least likely to adopt. The Schlumberger and Evernote case studies highlight how strong, well-thought-out, and continuously refined SLOs benefit SRE teams and the entire business. Both organizations' experiences show how an SLO- and error-budget-based approach can drive operational performance and bridge the gap between development and operations teams.

Evernote's and Schlumberger's stories are very real examples of how it is not only possible but vital to begin your SRE journey with initial SLOs and SLIs and revise them as the organization gains experience using these tools. Both companies believed that having a distinct

framework for measuring their service reliability was an important business goal. They defined user journeys, began measuring critical service aspects with SLIs and SLOs, and refined them as their SRE culture matured.

Additionally, Schlumberger and Evernote demonstrate the importance of continually reviewing and evaluating their SLOs. Only 46% of the *SLO Adoption and Usage Survey* respondents review their SLOs once a quarter or more frequently. Both case studies demonstrate the value of evaluating SLOs to ensure that they measure the features most important to user happiness and evolve as the organization's products and goals evolve.

Lastly, Evernote and Schlumberger have two very different takes on SLOs, SLIs, measurement styles, and implementation, reinforcing the notion that every company's SRE journey is going to look different. We hope that sharing their stories alleviates some of the trepidation your organization may have about moving to an SRE model that is driven by SLOs and error budgets.

CHAPTER 7
Conclusion

SRE is an emerging methodology that is integral to maintaining and defending the reliability of an organization's service. The *SLO Adoption and Usage Survey* shows that many organizations have integrated basic principles of SRE into their operations (e.g., applying software engineering to ops, implementing capacity planning, and holding blameless postmortems) and are ready to advance their SRE practices by implementing an SLO and an error-budget approach to managing reliability.

SLOs and error budgets are powerful business tools that provide executives and product owners, as well as development, operations, and SRE teams, a data-driven framework for measuring the quality of service and balancing two often competing demands: change/feature release velocity and service reliability. Realizing the full benefits of SLOs requires organizations to thoughtfully establish SLIs that inform SLOs and error budgets, and to continually evaluate and improve the quality of their SLOs.

The real magic of SLOs, SLIs, and error budgets is that they align traditionally divided teams (development and operations) on a common goal. SLOs provide precise numerical targets that provide telemetry into the service's performance that allow SRE and product teams to come together and more effectively manage innovation and risk.

Our survey shows that large organizations are more likely to have a longer history of using SLOs than smaller organizations do. While it makes sense that larger organizations have more formalized

processes, the heart of SLOs is one that we think applies to all organizations: knowing whether your system is working and, more importantly, knowing whether your users are happy! With the advancement of code instrumentation and service monitoring tools, meaningful SLOs are readily attainable for even those with nascent SRE or DevOps practices.

Our Evernote and Schlumberger case studies show not only how large organizations are developing new approaches to SRE but also how they implement, measure, and report on SLOs and SLIs. It's exciting for us to see how these companies factor SLOs and error budgets into product-, system-, and business-level discussions and decisions.

To our delight, we also see that smaller firms are embracing SRE methodology and are most likely to implement *all* SRE practices—not just SLOs. Although we have long suggested that organizations implement SRE at a pace that works best for their organization, we are surprised and encouraged that so many small-sized organizations have operationalized multiple best practices. While all organizations surely have a customer or end-user focus, we encourage our readers to avoid the trap of thinking that it is simply easier for smaller organizations to encode and deploy the full range of SRE practices beyond the more formal aspects like SLOs—smaller organizations, too, are dealing with rapidly scaling systems. They may have less process to work through, but they likely also have fewer resources than larger organizations do. We reason that smaller firms may have an increased focus on the end user because their survival demands it. They may be more agile and better able to adopt SRE practices. If this is true, we will note that agility in process adoption and improvement also lends itself to agility in technology adoption and improvement—characteristics that the *State of DevOps Report* has shown to drive developer happiness and organization value. We look forward to learning more about these firms' experiences.

Additionally, SLOs become increasingly important as more firms institute SRE teams. Forty-three percent of the survey respondents indicated that their organizations have SRE teams. We expect this number to only grow in the future, but without SLOs, organizations will not get the most out their SREs. SREs cannot manage their services correctly if they have not defined the behaviors that are most important to the service and customers and have not developed a standardized way to measure those behaviors.

Ultimately, SLOs help companies focus on their customers. Knowing and defining the level of service that causes customer dissatisfaction allows the business to decide where and when to invest in functionality versus reliability. If your customers are happy, chances are they will remain your customers. Because SRE asserts that you should focus on the features or portions of the customer journey important to customers, it ensures that SRE, product, developer, and operations teams, along with executives, understand and measure reliability as it matters to the customer. Understanding user happiness and how well a system meets expectations will inform decisions on whether to invest in making the system faster, more available, and more resilient.

About the Authors

Julie McCoy is an independent consultant who specializes in content development and product marketing for the technology and healthcare industries.

Nicole Forsgren is VP Research and Strategy at GitHub. Previously, she led the DORA (DevOps Research & Assessment) team. She is an expert in software development and delivery and has advised executives and technical experts on how to optimize their work.