cloud.google.com

Defining SLOs | Cloud Architecture Center | Google Cloud

15-19 minutes

This document is Part 1 of two parts that show how teams that operate online services can begin to build and adopt a culture of Site Reliability Engineering (SRE) by using service level objectives (SLOs). An SLO is a target level of reliability for a service.

In software as a service (SaaS), a natural tension exists between the velocity of product development and operational stability. The more you change your system, the more likely it will break. Monitoring and observability tools can help you maintain confidence in your operational stability as you increase development velocity. However, although such tools—known also as *application performance management* (APM) tools—are important, one of the most important applications of these tools is in setting SLOs.

If defined correctly, an SLO can help teams make data-driven operational decisions that increase development velocity without sacrificing stability. The SLO can also align development and operations teams around a single agreed-to objective, which can alleviate the natural tension that exists between their objectives—creating and iterating products (development) and maintaining system integrity (operations).

SLOs are described in detail in The SRE Book and The SRE Workbook, alongside other SRE practices. This series attempts to simplify the process of understanding and developing SLOs to help

you more easily adopt them. Once you have read and understood these articles, you can find more in the books.

This series aims to show you a clear path to implementing SLOs in your organization:

- This document reviews what SLOs are and how to define them for your services.
- Adopting SLOs covers different types of SLOs based on workload types, how to measure those SLOs, and how to develop alerts based on them.

This series is intended for SREs, operations teams, DevOps, systems administrators, and others who are responsible for the stability and reliability of an online service. It assumes that you understand how internet services communicate with web browsers and mobile devices, and that you have a basic understanding of how web services are monitored, deployed, and troubleshot.

The State of DevOps reports identified capabilities that drive software delivery performance. This series will help you with the following capabilities:

- Monitoring and observability
- Monitoring systems to inform business decisions
- Proactive failure notification

Terminology

The documents in this series use the following terms and definitions:

- **service level:** a measurement of how well a given service performs its expected work for the user. You can describe this measurement in terms of *user happiness* and measure it by various methods, depending on what the service does and what the user expects it to

do or is told it can do.

Example: "A user expects our service to be available and fast."

- **critical user journey (CUJ):** a set of interactions a user has with a service to achieve a single goal—for example, a single click or a multi-step pipeline.

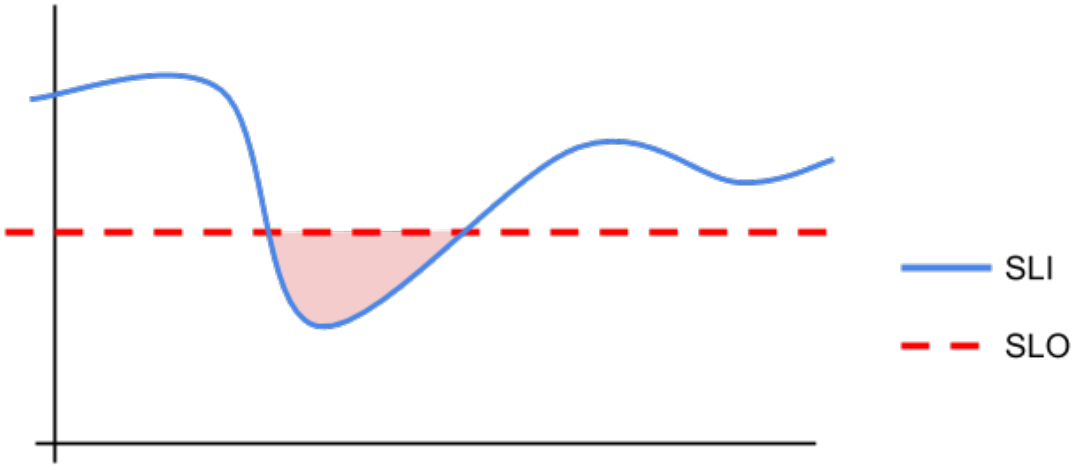
  Example: "A user clicks the checkout button and waits for the response that the cart is processed and a receipt is returned."

- **service level indicator (SLI):** a gauge of user happiness that can be measured quantitatively for a service level. In other words, to measure a service level, you must measure an indicator that represents user happiness with that service level—for example, a service's availability. An SLI can be thought of as a line on a graph that changes over time, as the service improves or degrades.

  Example: "Measure the number of successful requests in the last 10 minutes divided by the number of all valid requests in the last 10 minutes."

- **service level objective (SLO):** the level that you expect a service to achieve most of the time and against which an SLI is measured.

  Example: "Service responses shall be faster than 400 milliseconds (ms) for 95% of all valid requests measured over 14 days."



- **service level agreement (SLA):** a description of what must

happen if an SLO is not met. Generally, an SLA is a legal agreement between providers and customers and might even include terms of compensation. In technical discussions about SRE, this term is often avoided.

Example: "If the service does not provide 99.95% availability over a calendar month, the service provider compensates the customer for every minute out of compliance."

- **error budget:** how much time or how many negative events you can withstand before you violate your SLO. This measurement tells you how many errors your business can expect or tolerate. The error budget is critical in helping you make potentially risky decisions.

  Example: "If our SLO is 99.9% available, we allow 0.1% of our requests to serve errors, either through incidents, accidents, or experimentation."

Why SLOs?

When you build a culture of SRE, why start with SLOs? In short, if you don't define a service level, it's difficult to measure whether your customers are happy with your service. Even if you know that you can improve your service, the lack of a defined service level makes it hard to determine where and how much to invest in improvements.

It can be tempting to develop separate SLOs for every service, user-facing or not. For instance, a common mistake is to measure two or more services *separately*—for example, a frontend service and a backend datastore—when the user relies on *both* services and isn't aware of the distinction. A better approach is to develop SLOs that are based on the product (the collection of services) and focus on the most important interactions that your users have with it.

Therefore, to develop an effective SLO, it's ideal that you understand your users' interactions with your service, which are called *critical user journeys* (CUJs). A CUJ considers the goals of your users, and how your users use your services to accomplish those goals. The CUJ is defined from the perspective of your customer without consideration for service boundaries. If the CUJ is met, the customer is happy, and happy customers are a key measurement of success for a service.

A key aspect to customer happiness with a service is a service's reliability. It doesn't matter what a service does if it's not reliable. Thus, reliability is the most critical feature of any service. A common metric for reliability is *uptime*, which conventionally means the amount of time a system has been up. However, we prefer a more helpful and precise metric: *availability*. *Availability* still answers the question of whether a system is up but in a more precise way than by measuring the time since a system was down. In today's distributed systems, services can be partially down, a factor that uptime doesn't capture well.

Availability is often described in terms of *nines*—such as 99.9% available (three nines), or 99.99% available (four nines). Measuring an availability SLO is one of the best ways to measure your system's reliability.

In addition to helping define operational success, an SLO can help you choose where to invest resources. For example, SRE books often note that each *nine* that you engineer for can result in an incremental cost with marginal utility. It is generally recognized that achieving the next *nine* in availability costs you ten times as much as the preceding one.

Choosing an SLI

To determine if an SLO is met (that is, successful), you need a

measurement. That measurement is called the *service level indicator* (SLI). An SLI measures the level of a particular service that you're delivering to your customer. Ideally, the SLI is tied to an accepted CUJ.

Selecting the best metrics

The first step in developing an SLI is to choose a metric to measure, such as requests per second, errors per second, queue length, the distribution of response codes during a given time period, or the number of bytes transmitted.

Such metrics tend to be of the following types:

- **Counter.** For example, the number of errors that occurred up to a given point of measurement. This type of metric can increase but not decrease.
- **Distribution.** For example, the number of events that populate a particular measurement segment for a given time period. You might measure how many requests take 0-10 ms to complete, how many take 11-30 ms, and how many take 31-100 ms. The result is a count for each bucket—for example, [0-10: 50], [11-30: 220], [31-100: 1103].
- **Gauge.** For example, the actual value of a measurable part of the system (such as queue length). This type of metric can increase or decrease.

  For more information about these types, see the Prometheus project documentation and the Cloud Monitoring metric types `ValueType` and `MetricKind`.

  An important distinction about SLIs is that *not all metrics are SLIs*. In fact, the SRE Workbook states the following (emphasis added):

  > *"...we generally recommend treating the SLI as the **ratio of two numbers**: the number of good events divided by the total*

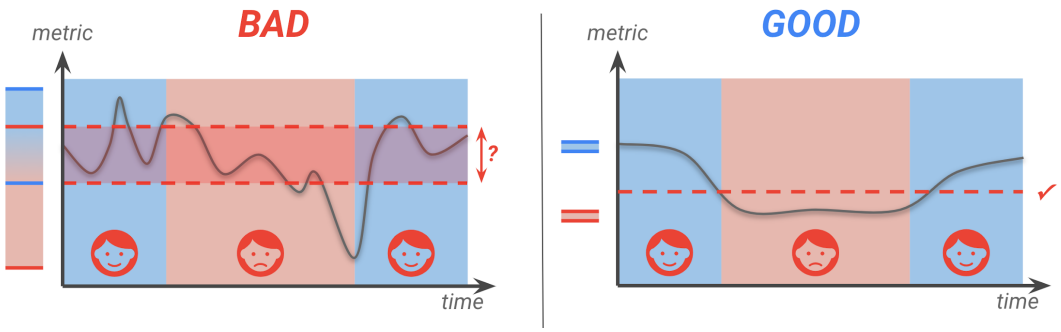
number of events..."

"The SLI ranges from 0% to 100%, where 0% means nothing works, and 100% means nothing is broken. We have found this scale intuitive, and this style lends itself easily to the concept of an error budget."

Many software companies track hundreds or thousands of metrics; only a handful of metrics qualify as SLIs. So apart from being a ratio of good events to total events, what qualifies a metric as a good SLI? A good SLI metric has the following characteristics:

- **The metric directly relates to user happiness.** Generally, users are unhappy if a service does not behave the way they expect it to, fails, or is slow. Any SLOs based on these metrics can be validated by comparing your SLI to other signals of user happiness—for example, the number of customer complaint tickets, support call volume, social media sentiment, or escalations. If your metric doesn't correspond to these other metrics of user happiness, it might not be a good metric to use as an SLI.
- **Metric deterioration correlates with outages.** A metric that looks good during an outage is the wrong metric for an SLI. A metric that looks bad during normal operation is also the wrong metric for an SLI.
- **The metric provides a good signal-to-noise ratio.** Any metric that results in a large number of false negatives or false positives is not a good SLI.
- **The metric scales monotonically, and approximately linearly, with customer happiness.** As the metric improves, customer happiness improves.

  Consider the graphs in the following diagram. Two metrics that might be used as SLIs for a service are graphed over time. The period when a service degrades is highlighted in red, and the period when a service is good is highlighted in blue.



In the case of the bad SLI, the user's unhappiness doesn't correspond directly with a negative event (such as service degradation, slowness, or an outage). Also, the SLI fluctuates independently of user happiness. With the good SLI, the SLI and user happiness correlate, the different happiness levels are clear, and there are far fewer irrelevant fluctuations.

Selecting the right number of metrics

Usually, a single service has multiple SLIs, especially if the service performs different types of work or serves different types of users. For example, separating read requests from write requests is a good idea, as these requests tend to act in different ways. In this case, it is best to select metrics appropriate to each service.

In contrast, many services perform similar types of work across the service, which can be directly comparable. For example, if you have an online marketplace, users might view a homepage, view a subcategory or a top-10 list, view a details page, or search for items. Instead of developing and measuring a separate SLI for each of these actions, you might combine them into a single SLI category—for example, *browse services*.

In reality, the expectations of a user don't change much between actions of a similar category. Their happiness is not dependent on the structure of the data they are browsing, whether the data is derived from a static list of promoted items or is the dynamically generated result of a machine learning-assisted search across a massive dataset. Their happiness is quantifiable by an answer to a

question: "Did I see a full page of items quickly?"

Ideally, you want to use as few SLIs as possible to accurately represent the tolerances of a given service. Typically, a service should have between two and six SLIs. If you have too few SLIs, you can miss valuable signals. If you have too many SLIs, your on-call team has too much to track with only marginal added utility. Remember, SLIs should simplify your understanding of production health and provide a sense of coverage.

Choosing an SLO

An SLO is composed of the following values:

- **An SLI.** For example, the ratio of the number of responses with HTTP code `200` to the total number of responses.
- **A duration.** The time period in which a metric is measured. This period can be calendar-based (for example, from the first day of one month to the first day of the next) or a rolling window (for example, the last 30 days).
- **A target.** For example, a target percentage of good events to total events (such as 99.9%) that you expect to meet for a given duration.

  As you develop an SLO, defining the duration and target can be difficult. One way to begin the process is to identify SLIs and chart them over time. If you can't decide what duration and target to use, remember that your SLO doesn't have to be perfect right away. You likely will iterate on your SLO to ensure that it aligns with customer happiness and meets your business needs. You might try starting with two *nines* (99.0%) for a month.

  As you track SLO compliance during events such as deployments, outages, and daily traffic patterns, you can gain insights on what target is good, bad, or even tolerable. For example, in a

background process, you might define 75% success as adequate. But for mission-critical, user-facing requests, you might aim for something more aggressive, like 99.95%.

Of course, there isn't a single SLO that you can apply for every use case. SLOs depend on several factors:

- customer expectations
- workload type
- infrastructure for serving, execution, and monitoring
- the problem domain

Part 2 in this series, Adopting SLOs, focuses on domain-independent SLOs. Domain-independent SLOs (such as service availability) do not replace high-level indicators (such as widgets sold per minute). However, they can help measure whether a service is working regardless of the business use case.

Domain-independent indicators can often be reduced to a question-for example, "Is the service available?" or "Is the service fast enough?" The answer is most often found in an SLO that accounts for two factors: availability and latency. You might describe an SLO in the following terms, where X = 99.9% and Y = 800 ms:

> *X% of valid requests are successful and succeed faster than Y ms.*

What's next?

- Read Adopting SLOs, which explores these definitions in more detail and introduces other SLIs that might be more appropriate for various workload types.
- Check out other SRE resources:
- The SRE Book