BLAMELESS

BEYOND THE 4 SRE GOLDEN SIGNALS

Table of Contents

Overview

Monitoring has always been essential to reliable solutions. Without knowing the health of your system, you can't know how to improve reliability. So how do you know what to monitor? Your system generates lots of data, but the four golden signals are most essential to getting started and understanding how your service is performing.

The four golden signals emerged as a concept to help engineering teams focus on the most critical aspects of a system's health. They are:

- Latency: the time it takes to serve a request.
- Traffic: the total number of requests across the network.
- Errors: the number of requests that fail.
- Saturation: the load on your network and servers.

Google's SRE book, the first major text to explain site reliability engineering, dedicates entire sections to each of the golden signals. If that's not enough testimony to their importance, Google says plainly, "If you can only measure four metrics of your user-facing system, focus on these four."

In recent years, though,organizations are realizing that the golden signals are just the foundation for a more meaningful understanding of system health. In this eBook, we'll get you caught up with the cutting edge in monitoring for reliability. We'll examine how to get the most out of the golden signals, and show you how to build beyond them.

The Four Golden Signals

Let’s take a look at each golden signal to explore what it measures and how it affects service reliability.

Latency

What is it?

Latency measures how slow your system is. When a user makes a request, how long does it take for the system to return the requested data?

Note that if the system returns an error instead of the requested data, it could arrive much faster than the correct data. Make sure to keep track of these separately.

How does it impact reliability?

A system that is almost always available but almost always slow will still be seen as unreliable by users. If it takes too long for a service to function, users will abandon their requests – not any different than if the service was down!

How do you monitor it?

Log whenever a request is made, then log when the data or response reaches the user. Delineating which data is being requested of which services, and if the response is correct, helps to understand your system's performance in a more nuanced way.

How do you know if it needs improving?

Look for when users are pained by latency. Take a closer look at incidents that caused latency. How usage dropped off if latency increased tells you what level of latency is unacceptable. Compare the use of faster and slower parts of your service for more insights.

How do you improve it?

How to improve latency first depends on the cause of the latency, which you discern from looking at other golden signals. If latency grows with traffic and saturation, then you probably need to provision additional infrastructure resources.

If latency varies by the type of request, then you probably need to optimize the code that processes slower responses.

Traffic

What is it?

Traffic measures how much your system is being used. Depending on the type of service you're measuring, this could be the number of active users or the number of requests that users make.

How do you know if you need to improve it?

Traffic needs further inspection when it causes issues with the service's speed, availability (in other words can you log-in), or error rate. If your service can handle a small number of users with acceptable levels, but strains when traffic increases, you need to focus on better ways to manage. Traffic increases could be seasonal or at specific times of the day. At the same time, you may have over-provisioned resources if your traffic is consistently low. You would need to streamline resources to save operational costs.

How does it impact reliability?

Traffic doesn't necessarily impact your service's reliability. Instead, traffic is potentially a cause of other issues that impact reliability, like saturation and latency. Traffic is an indicator of reliability, as it's often impacted by reduced reliability. If you see your traffic drop off, it could be caused by another reliability issue. Because of these inter-connections, traffic is still an important measurement to factor in when measuring service reliability.

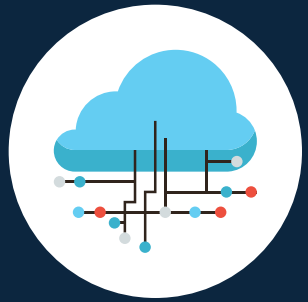
How do you improve it?

Keeping your services functioning with acceptable performance levels consistently while traffic is high requires provisioning more infrastructure resources – claiming more server space. It could also mean optimizing code to more effectively balance traffic.

When traffic is low, it can often be caused by some other reliability issue. Look for things that could be negatively affecting users, and fix them.

How do you monitor it?

Counting the amount of users or requests per time period tells you the volume of traffic.



Errors

What is it?

Errors are when service requests don't return the correct data, or even any response. When every request returns an error, the service is likely having an outage.

How does it impact reliability?

If users receive errors, incorrect data, or no data at all, the service is functionally unavailable to them. This is the biggest factor which leaves users viewing your system as unreliable and maybe not worth revisiting.

How do you monitor it?

Catching errors can be difficult. Sometimes good data and erroneous data can look the same to a monitoring tool. Other times, an outage will both prevent a response and block a tool from noticing that a request was made, making it seem like everything is fine. Using observability tools that are more sophisticated for event-based reports such as end-to-end tracing will track that all requests are completed and all data is verified. This lets you see your true error rates.

How do you know if you need to improve it?

It seems tempting to say you should aim for a 0% error rate, but that isn't feasible: some failure is inevitable. Instead, you should look to see when errors cause users pain. This can depend on where errors occur – if something critical fails once, it can cause a lot more pain than when a less important part of the service fails more frequently. When users complain and quit because of a bad experience you need to immediately address error rates for those critical services.

How do you improve it?

Errors can be caused by different factors, from oversaturation to bad code that isn't working as expected in production. If your error rate is proportional to saturation, you need to devote resources to decrease saturation. If errors suddenly spike up, it's perhaps due to a specific incident. Incidents could be user-driven or generated due to the infrastructure environment. You'll need to use incident response processes to resolve and restore availability.

Saturation

What is it?

Saturation measures how much of your resources are being used versus how much is available, as a percentage. These resources could include server bandwidth, CPU usage, or storage space used and available.

How does it impact reliability?

If saturation on a service ever reaches 100%, it can create many errors or even a total outage. Depending on the resource that's at capacity, the errors generated will be different. No matter what, users will experience pain with errors due to saturation, as it occurs when usage is high. As a result, user trust on your service reliability will quickly drop.

How do you monitor it?

Monitoring saturation involves internal tools that keep track of your own infrastructure, whether on your own servers or the cloud. The cloud provider will provide some level of information. You'll also need a variety of monitoring and observability tools to check in on how your resources are being used.

How do you know if you need to improve it?

When saturation causes your system to slow or generate errors, you need to address it. This is necessary at 100% saturation, but your system may operate slower or generate more errors as it approaches that point. The key is to look at how users change their behavior as saturation increases. Get ahead of the problem before users experience pain.

How do you improve it?

There are two ways to reduce saturation. You can invest in infrastructure: buy larger servers for more capacity to allow for expected growth. This reduces the percentage of your available resources that your traffic takes up. You can also reduce the impact of traffic, so that the same number of users take up less resources. This requires optimizing your service's code.

Monitoring Golden Signals

Now that we've seen how and why you should track the four golden signals, you need to establish how the team will monitor them, including how you'll build dashboards for real-time viewing. This is an up-front time investment. Monitoring tools are key to make your data meaningful and accessible. You'll need to determine which set of tools will work best for you now and as you scale. There are a number of open source monitoring tools to get started if your resources are tight, but implementing them can take more work.

Observability solutions become more challenging as your system grows in complexity. The development team needs to incorporate rich telemetry in new code that ships with new releases. Setting up good monitoring systems is worth it; during incident troubleshooting causes will surface very quickly. You'll be able to identify which parts of your code are impacting your metrics.

Without a good monitoring solution, if you see that a signal is doing poorly, you might not know what aspects of your system are causing the issue. Only with a proper monitoring solution in place can you take action based on your signals.

There are two major categories of monitoring for the golden signals: white box and black box. White box monitoring involves looking at your system from the inside, watching what your code is reporting as it's used in production. Black box monitoring involves looking at your services as a user would, making requests from the outside and logging data about the responses received.

To have a complete picture of monitoring, you need to incorporate both techniques. You want to experience the user's perspective to determine if the system is functioning as a whole, and see the details of how each part or feature of your code is running. Monitoring tools can sort through this data to create meaningful reports of the golden signals.

Going Beyond the Golden Signals

The golden signals are fundamental to understanding the health of your system, and setting yourself up to manage the service as a whole. This is an important part of your reliability journey. However, to get a true picture of reliability, you have to evolve and build beyond the golden signals.

Reliability through users' eyes

When we discuss how the golden signals impact reliability, we always want to focus on how users are impacted. This is because reliability isn't an objective measure of your system, like the golden signals, but a subjective metric based on how users interact with your system.

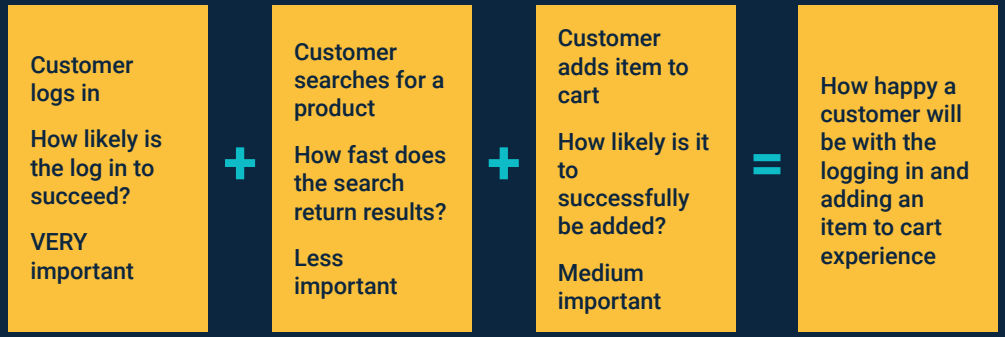
To understand this, consider two services running in your system. One is the log-in service that every user uses. The other is a search filtering capability that most users never use. The log-in service experiences an error once every couple days, and the search filter experiences an error every hour.

By examining these features just using the golden signals, it would seem that the search filter is a bigger issue, and should therefore receive attention first. However, most users won't notice that it's causing errors. In this case, having the search filter not work occasionally is likely a lesser inconvenience than not being able to log in, even if the log in error is much more rare. The perceived reliability and the resulting annoyance isn't just about watching the metrics. It's about the user experience as a context to those metrics.

Building SLIs from user journeys

The above example of two features is simple, but the idea scales up with the increasing complexity when you apply it to the entire system. The way to transform golden signals to something that represents the subjective user experience of reliability is by building user journeys and Service Level Indicators (SLIs).

User journeys represent how a typical user engages with your service. You map what actions they take, what standards they expect for the latency and error rate of that action, and how important that action is for their overall experience. By assigning weights based on how critical each action is, you can make a formula that represents user happiness with a given experience as a single metric.



This is your service level indicator, or SLI. SLIs are a way to view your golden signals in a more meaningful way, as a representation of user happiness. This allows you to effectively prioritize improving the reliability of your service, as there's no greater driver of business value than user happiness.

Using SLOs as brakes and accelerators

Once you have an SLI, you can build a service level objective, or SLO, based on it or a number of SLIs. An SLO is the "acceptable" level for your chosen, related SLI(s). In other words, if an SLI is the thing you measure, SLO is the precise measurement(s) you're aiming for. SREs calibrate the SLO according to the point at which the SLI won't cause any user frustration, such as "99% of requests returned with no errors over a specified timeframe", usually monthly successful logins.

Depending on how important a request is, there's some amount of latency or errors that a user will accept. Past that point, improving the error rate or response speed may go unappreciated, or even unnoticed. Increasing availability requires more and more effort and at a certain point will mean diminishing returns. Finding the right point to set the SLO to achieve user happiness without spending unnecessary effort is a challenge, but worth refining and pursuing.

That's why an SLO is a target you should try to approach from both sides. If you're close to reaching or exceeding the SLO, you know users will be unhappy and you should prioritize changes to improve the metric. If you're comfortably within the SLO, you can feel secure that the reliability of those services don't need resources and effort can be spent elsewhere.

By using the SLO this way, you can gauge user happiness as a way to make decisions for the full engineering team. The buffer you have until reaching or exceeding your SLO is your error budget, which is your reliable guidepost that tells you when to take risks. You can know that even if a new development project makes some services slower, the latency is acceptable and within the SLO. Users are generally happy.

A complete picture of reliability

As you build upon the Golden Signals and contextualize them with user happiness in mind, you'll have a more complete picture of the reliability of your system. Rather than just focus on the latency, traffic, error rate, and saturation of your services, you know if users are having positive experiences.

Blameless helps you put the pieces together. Our SLOs are best-in-class and allow you to monitor increasingly sophisticated SLIs that represent user journeys. We integrate with the most popular observability and monitoring tools, helping you connect how you're performing with your error budget. To see it in action, request a personalized demo.

BLAMELESS

Optimize reliability of software and resilience of teams.

Blameless drives reliability across the entire software lifecycle by operationalizing Site Reliability Engineering (SRE) practices. Teams share a unified context during incident response, efficiently communicate, and resolve faster. Detailed Retrospectives give teams a data-driven approach to learn and Service Level Objectives (SLOs) inform teams where to prioritize work and innovate at velocity. Customers include brands such as Procore, Under Armour, Citrix, Mercari, Fox, and Home Depot who embrace a blameless culture, team resilience, and greater reliability to protect their customers.

Blameless is a 2021 Gartner Cool Vendor recipient and is backed by Accel, Lightspeed, Decibel and Third Point Ventures.
More info: www.blameless.com or LinkedIn, Twitter.