Service Level Indicator (SLI):

An SLI is a quantitative measure of some aspect of the level of service that is provided. It's a carefully defined metric that measures a specific aspect of service level.

Example: The percentage of HTTP requests that return a successful status code (2xx or 3xx).

Service Level Objective (SLO):

An SLO is a target value or range of values for an SLI. It's a goal you set for the level of service you aim to provide.

Example: 99.9% of HTTP requests should return a successful status code over a 30-day period.

Key Differences:

1. **Measurement vs. Goal**: An SLI is what you measure, while an SLO is the goal you set for that measurement.
2. **Specificity**: SLIs are precise metrics, while SLOs are targets based on those metrics.
3. **Usage**: SLIs help you understand your current performance, while SLOs help you set and work towards performance targets.

In this lab, we'll define an SLI (HTTP success rate) and then set an SLO (target success rate) based on that SLI.

**Why This Matters**

HTTP status codes are a fundamental indicator of your service's health. By setting an SLO for successful requests, you can quickly identify when your service is failing to meet user expectations and take action before it becomes a significant problem.

In this exercise you will create an SLO that checks if the http status codes of an application are in the 200s, 99% of the time.

Click on Create SLO

Select the logs-\* data view

Add a Query filter - this allows you to only consider a single service (or host, container, etc.) for the SLI

copy

service.name:"web\_server"

Then set a Good query to define the events that you consider to be a healthy sign.

copy

http.response.status\_code < 400

Scroll down a little, then

* set Duration to 7 days
* set SLO name, for example http status code
* click the Create SLO button at the bottom of the screen

The Power of Logs for SLOs

Log-based SLOs offer unique advantages in your reliability toolkit:

1. **Rich Context**: Logs capture detailed information about errors and system behavior, enabling more nuanced SLOs.
2. **Custom Error Definitions**: Define "errors" based on your specific application logic, going beyond simple status codes.
3. **Cross-Service Visibility**: Track errors across different services, providing a holistic view of your system's health.
4. **Early Warning System**: Identify subtle issues and patterns before they escalate into major problems.
5. **Debugging Aid**: When SLO breaches occur, log data provides invaluable context for faster problem resolution.

By leveraging log data for SLOs, you gain deeper insights into your system's behavior and can craft more meaningful reliability targets.

Log-based error tracking allows you to catch issues that might not be reflected in simple metrics. By grouping errors by service name, you can identify which specific components of your system are contributing to reliability issues, enabling more targeted improvements and faster problem resolution.

In this exercise, you will create an SLO that checks if the logs from your services contain errors, aiming for less than 1% error rate across all services.

Click on Create SLO[A screenshot of a computer

AI-generated content may be incorrect.](https://play.instruqt.com/assets/tracks/559pcbd0ybbd/42297fafdc03dbc025e5335e491f272e/assets/CleanShot%202024-07-05%20at%2009.40.03%402x.png)

Select the logs-\* pattern and set a Good query to define the events that you consider to be healthy. This query considers any log entry without the word "error" as a good event.

copy

not message: error

Also set the Group By to the field:

copy

service.name

[A screenshot of a computer

AI-generated content may be incorrect.](https://play.instruqt.com/assets/tracks/559pcbd0ybbd/b68e1b455464950b90f3253251006bcd/assets/CleanShot%202024-07-03%20at%2010.01.50%402x.png)

Scroll down,

* set the Duration to 7 days,
* set the target to 80%
* give it a name
* click the Create SLO button at the bottom of the screen[A screenshot of a computer

  AI-generated content may be incorrect.](https://play.instruqt.com/assets/tracks/559pcbd0ybbd/58d7881f3756c1f4a086d4038416f710/assets/CleanShot%202024-07-03%20at%2010.42.03%402x.png)Your log-based SLO is now created! The overview page will take a few seconds to update. Click the button at the top of the page to refresh.[A screenshot of a computer

  AI-generated content may be incorrect.](https://play.instruqt.com/assets/tracks/559pcbd0ybbd/1bd9f178d270cf42c284146f5035f7c6/assets/CleanShot%202024-07-05%20at%2009.44.17%402x.png)

You should now see multiple distinct SLOs on your screen, one for each unique service.name.[A screenshot of a computer

AI-generated content may be incorrect.](https://play.instruqt.com/assets/tracks/559pcbd0ybbd/19be174c43a6a2d994e4fdcc7e723cab/assets/CleanShot%202024-07-05%20at%2009.44.59%402x.png)

**Conclusion: The Power of Log-Based SLOs**

You've now created and refined a log-based SLO. Through this process, you've:

1. Leveraged the rich context provided by logs to create more meaningful SLOs
2. Used service-level granularity to automatically create disctinct SLOs

Remember, log-based SLOs offer unique insights that metric-based SLOs might miss. They allow you to define complex "good" and "bad" events, provide more context for troubleshooting, and offer a more detailed view of your system's health.

As you continue working with SLOs, consider how combining log-based and metric-based SLOs can provide a more comprehensive view of your services' reliability and performance.

Quantifying Reliability with Metrics

Metric-based SLOs provide a powerful approach to measuring and managing service reliability:

1. **Real-Time Monitoring**: Metrics offer near-instantaneous visibility into system performance.
2. **Scalability**: Easily track and aggregate data across large, distributed systems. Metric collection and storage is typically less resource-intensive than full log retention.
3. **Performance Focus**: Ideal for monitoring key performance indicators like latency, throughput, and resource utilization.

By implementing metric-based SLOs, you create a quantifiable framework for continuously improving your service's reliability and performance.

Metric-based SLOs are crucial for monitoring quantifiable aspects of your system's performance. They're particularly useful for continuous measurements like latency, throughput, or resource utilization.

Time slice SLOs introduce an important concept:

* **Time Slice SLOs**: These break the overall time window into smaller slices of a defined duration. The SLO is calculated based on the number of "good" slices versus the total number of slices.

Why Time Slice SLOs Matter:

1. **Handling Fluctuations**: They're ideal for metrics that fluctuate frequently, like latency or CPU usage.
2. **Granular Analysis**: Allow you to pinpoint specific time periods where performance degraded.
3. **Flexible Budgeting**: Provides a more nuanced way to manage your error budget over time.

In this lab, we'll use a time slice approach for our database latency SLO.

**Why This Matters**

Database performance is often a critical factor in overall application performance. By setting and monitoring an SLO for database latency:

1. You ensure that database operations meet performance expectations.
2. You can quickly identify when optimizations are needed.
3. You gain insights into how database performance impacts overall system health.

Using a time slice approach for latency allows you to account for normal fluctuations while still maintaining high standards.

In this exercise, you will create an SLO that monitors database query latency, aiming for 85% of time slices to have an average latency under 45ms.

Click on Create SLO[A screenshot of a computer

AI-generated content may be incorrect.](https://play.instruqt.com/assets/tracks/559pcbd0ybbd/c2a77f3b4fd217d87a8cf2dc5d0d7bca/assets/CleanShot%202024-07-05%20at%2009.49.45%402x.png)

Select Timeslice metric as the SLI type

Set the following fields:

1. Index: metrics-\*
2. Timestamp field: @timestamp
3. Query filter:

copy

service.name:database

1. For the Metric definition:
   * Aggregation: Average (default)
   * Field: event.duration
   * Equation: A (default)
   * Comparator: Less than or equal to
   * Threshold: 45000[A screenshot of a computer

     AI-generated content may be incorrect.](https://play.instruqt.com/assets/tracks/559pcbd0ybbd/a335ed96905755536df94df4515681b1/assets/CleanShot%202024-07-05%20at%2009.55.31%402x.png)

Next set the Duration to 7 days

* set the target to 85
* give it a name like database latency
* click Create SLO at the bottom of the screen.

[A screenshot of a computer

AI-generated content may be incorrect.](https://play.instruqt.com/assets/tracks/559pcbd0ybbd/2b66abe8833f68046174a5f95ddd729f/assets/CleanShot%202024-07-05%20at%2009.58.06%402x.png)Your metric-based, time slice SLO for database latency is now created! The overview page will take about one minute to update.[A screenshot of a computer

AI-generated content may be incorrect.](https://play.instruqt.com/assets/tracks/559pcbd0ybbd/0b99b0a8bf13157ec15b078c9ac2b5d5/assets/CleanShot%202024-07-05%20at%2009.59.27%402x.png)Expand the next exercise to continue below.

Let's now click on our newly created SLO to view its details.[A screenshot of a computer

AI-generated content may be incorrect.](https://play.instruqt.com/assets/tracks/559pcbd0ybbd/8b2a2acab7e6aa23749abc585dfdea3e/assets/CleanShot%202024-07-05%20at%2009.59.39%402x.png)

It's clear that we're slowly burning down our error budget, and the SLI is also doing worse over time. To investigate this problem more closely we can simply look at the events by clicking on view events[A screenshot of a computer

AI-generated content may be incorrect.](https://play.instruqt.com/assets/tracks/559pcbd0ybbd/8b672642ac5432050ad2c40b9ee0599b/assets/CleanShot%202024-07-05%20at%2009.59.56%402x.png)Next we'll then click the visualize icon[A screenshot of a computer

AI-generated content may be incorrect.](https://play.instruqt.com/assets/tracks/559pcbd0ybbd/2677f99037d62ab52955425f8efaee91/assets/CleanShot%202024-07-05%20at%2010.04.04%402x.png)

And then search for event.duration on the top left. Now drag and drop this field into the middle of the screen.[A screenshot of a computer

AI-generated content may be incorrect.](https://play.instruqt.com/assets/tracks/559pcbd0ybbd/4188e8a965fc512cc6d874a3a151aca2/assets/CleanShot%202024-07-05%20at%2010.01.46%402x.png)

Finally we can adjust the time picker at the top to last 7 days[A screenshot of a computer

AI-generated content may be incorrect.](https://play.instruqt.com/assets/tracks/559pcbd0ybbd/5b6db3fb32fa3c71b86881ffab408016/assets/CleanShot%202024-07-05%20at%2010.02.06%402x.png)

It's clear now that the database is slowing down over the course of a day, which is causing our bumpy SLI.[A screenshot of a computer

AI-generated content may be incorrect.](https://play.instruqt.com/assets/tracks/559pcbd0ybbd/d571b56aa9221070643cbd4c6b08e5b0/assets/CleanShot%202024-07-05%20at%2010.02.19%402x.png)

**Conclusion: Leveraging Metric-Based and Time Slice SLOs**

You've now created and refined a metric-based, time slice SLO for database latency. Through this process, you've:

1. Learned how to use metrics for creating precise, quantifiable SLOs
2. Understood the value of time slice SLOs for handling fluctuating metrics
3. Gained insights into database performance patterns

Remember, metric-based SLOs like this one provide:

* Clear, objective measures of system performance
* The ability to set precise targets for continuous metrics

Time slice SLOs specifically offer:

* Flexibility in handling natural performance fluctuations
* More nuanced error budgeting
* Detailed insights into performance patterns over time