Service Level Objectives

Online services want to ensure customers have a positive experience in their product. One way they do this is by defining **Service Level Objectives**, or **SLOs**.

An SLO is like a guarantee made to customers. By defining an SLO, an organization is promising to maintain a defined level of service, often related to the performance and availability of their systems. Here are some examples:

- The website landing/home page will take <5 seconds to load 99% of the time over a 7 day period
- Checkout service will operate without error 99% of the time each 30 days
- The landing/home page will be able to successfully process at least 1,000 requests per second, 99% of the time, measured in 90-day increments
- Average response time in the cart service is <300 milliseconds, 98% of the time, measured in 7-day increments

Notice each includes the same components:

- **Scope**: The specific area an SLO relates to, such as "checkout service", or "landing page". This is an area or function that impacts user experience.
- **Target Value**: A measurable threshold for performance, like "1,000 requests per second" or "less than 5 seconds". Data used to measure an SLO is referred to as a **Service Level Indicator (SLI)**, because it *indicates* whether an SLO is met.
- **Target Rate**: A percentage of the time performance will meet the target value.
- **Time Window**: Period for which data is evaluated, such as "over 7 days".

Notice also that each SLO is centered around user experience. SLOs should capture performance and availability levels that, even if *barely* met, would keep the average user satisfied. In the simplest terms:

- **Service *meets* SLO targets → Happy users**
- **Service *misses* SLO targets → Frustrated users**

For this reason, SLOs monitor the performance and availability most likely to impact a user's experience: like load times, error rates, and functionality working in the manner users need and expect.

Service Level Indicators, Objectives, and Agreements

An SLO should be focused on user experience. Think about what makes a positive user experience in a website, app, or online service:

- Load times are reasonable
- Functionality works as expected
- Users are able to complete whatever task or action they visited the service for, for example:
    - On an ecommerce site, you can complete an order
    - On a digital calendar, you can create, edit, and delete calendar events without error

For example, a typical user won't care about the precise integrity of your internal systems. But they *will* care if those systems negatively impact their experience with long load times, frequent errors, and inconsistent functionality.

When considering SLOs, focus on users' goals and experiences. Ask yourself questions like:

- How are your users interacting with your application?
- What is their journey through the application?
- Which parts of your infrastructure do these journeys interact with?
- What are they expecting from your systems? What are they hoping to accomplish?
- What would prevent them from accomplishing their goals?

After identifying what is necessary to keep users content, the next step is locating data to measure each SLO.

Service level indicators (SLIs)

Data used to measure an SLO is called a **Service level indicator (SLI)**, because it *indicates* whether the SLO is met.

For example, the SLO *"The website landing/home page will take <5 seconds to load 99% of the time over a 7 day period"* refers to website load times. To determine whether this SLO is met, you would need latency data capturing average landing page load times for the past 7 days. This latency data would be the SLI for this SLO.

Because SLOs aim to quantify user experience, their associated SLIs are values that correspond to the quality of that user experience. This includes things like:

- The number of requests to an endpoint that complete successfully
- The number of requests to an endpoint that complete within 500ms
- Average load times in specific areas and pages users need

SLIs are often formatted as percentages, representing the rate of "good" events out of all valid events. For example, a valid event could be a user request to an endpoint, regardless of the request's success or failure. A "good" event would be a successful `200 OK` request to that endpoint. The SLI would be the percentage of all valid endpoint requests that were `200 OK` successful.

$$\text{SLI}: \frac{\text{good events}}{\text{valid events}} \times 100\%$$

Service level agreements (SLAs)

Service level agreements, or **SLAs**, address expectations, impacts, and consequences if agreed-upon reliability is not met. This is a contract that outlines what happens if a service meets—or misses—SLO targets. Consequences for missing SLOs are often business decisions or financial repercussions, such as:

- The customer expects a given service to have a 0.05% maximum error rate daily, or they'll receive a rebate.
- The customer expects only 10% of monthly requests to take longer than 500ms to complete, or they'll be reimbursed for the compute overage.

These stipulations are often related to business and financial relationships between the user and service. As such, SLAs are generally outside of the purview of Engineers. Instead, they're often written by other company stakeholders before specific SLOs are quantified.

Engineering teams can derive their SLOs from the SLAs defined by other parts of the company, but SLO targets should generally be more strict than SLA targets.

Time window

In an SLO, the time window refers to the period for which the SLI will be evaluated to determine if it meets SLO targets.

It's best practice to define a rolling time window—such as `the past 30 days` —as opposed to a fixed window, like `the month of August` . This is because a user's perception of a service's reliability is heavily influenced by their recent experience with that service. The average user will not reset their expectations when a new month begins.

Typical SLO time windows are 7, 30, and 90 days.

Target rates & reliability

You may have noticed SLO target rates are often ~99%, as seen in examples from previous lessons. But if SLOs are meant to guarantee positive user experiences, why aren't they aiming for 100%?

In reality, 100% reliability is an impractical target that goes against best practices for many reasons:

- **It's difficult**: Striving for perfection is hard, especially over longer periods of time. It puts undue stress and expectations on engineering teams.
- **Users don't need it**: The difference between 99% and 100% is negligible—even unnoticeable—for most users.
- **Teams must balance time**: Engineers can't spend all their time maintaining perfect reliability and availability. Some of their time must be used to deliver value through developing new features, products, etc.

Aiming for less than 100% reliability means a small amount of undesirable performance/events are allowable. For example, if an SLO has a target of 99% uptime, outages could still occur >=1% of the time without breaching the SLO. This amount of allowable "bad" events is called an **error budget**.

Error budgets

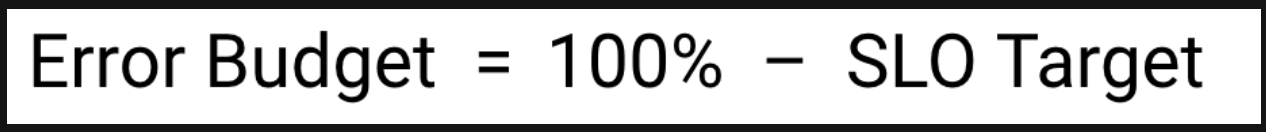
An error budget quantifies an acceptable amount of unreliability. In other words, the amount of "bad" events that can happen without breaching an SLO. They are formatted as 100% minus the SLO target. Consider the following:

```
Average cart service response time is <300 milliseconds, 98% of the time, measured in 7-day increments.
```

This SLO targets a response time of <300ms 98% of the time in a rolling 7-day window. This means >=2% of requests may have *longer* response times, without breaching the SLO. For every 100,000 requests, 2,000 are "allowed" to fail in a 7-day period, because the SLI's total rate of will still be >=98%.

How teams use error budgets

Like a financial budget, an error budget is meant to be spent. When a team knows how much budget is remaining in their SLOs, they can allocate time and resources accordingly.



When an error budget is nearly exhausted (<= 0% remaining), teams prioritize reliability to ensure they don't breach SLOs. This means focusing on lower-risk tasks unlikely to consume budget, such as:

- Prioritizing postmortem items
- Automating deployment pipelines
- Improving monitoring and observability
- Freezing feature releases

When an error budget is readily available (> 0% remaining), teams prioritize velocity and higher-risk tasks, because they have 'room' to encounter errors and issues without breaching SLOs. This includes things like:

- Releasing new features
- Making system changes
- Trying risky (but valuable) experiments

Burn rates

It's also helpful to understand when error budgets are being spent at a higher rate than usual. This is tracked by something called a burn rate. A **burn rate** is a unitless number indicating how fast an error budget is being "spent" relative to its SLO target.

For example, if an SLO with a 30-day window is on track to consume its budget in 30 days, it would have a burn rate of `1`. If it was on track to consume its budget in 15 days it would have a burn rate of `2`, because it is "spending" its error budget at 2x the intended rate.

By tracking burn rates and identifying when rates spike or increase, you can resolve issues and return the rate to baseline before your SLO's error budget is consumed.

SLOs in Practice

2. In your browser, log in to app.datadoghq.com

3. In the main menu, hover over **APM** and click **Traces** to navigate to the Traces Explorer.

4. Locate the facet panel on the left. Expand options for **Env** and **Service**.

> Service Level Objectives—or SLOs—define the level of service you strive to provide users. To identify SLOs, you must understand the user experience your application or product offers. This includes things like:
>
> - What do users achieve in your application?
> - What are the most critical pathways, tasks, or actions?
> - What would prevent them from having a good—or even functional—experience?

5. Navigate to APM > Services > Service Catalog. This is a list of all services associated with your account. Locate the env dropdown menu above the list. This is where you select your environment

6. Hover over the service you'd like to inspect in the list entry to reveal a Full Page button. Click Full Page to open the service overview page

7. Locate the Resources list near the bottom of the page.

8. In the Resources list on the service overview page, click yor desired `route/endpoint` to open its details.

9. Under Resource Summary, locate the Latency graph. Click the View full screen button (four outward arrows) above the graph to enlarge. Notice there are multiple metrics in the legend below the graph. Each metric name is preceded by p + a numerical value (for example, p50, p75, etc). This refers to a distribution percentile. The ideal percentile would be metric p99 this means that 99% of requests have latencies lower than this value.

Creating a Monitor for SLO's (in practice)

> *Some screenshots were taken from a Datadog lab. Just replace with your* `env` , `service` , `route/endpoint` .

Datadog offers two approaches to measuring SLOs:

1. **By Count**: References metric(s) tracked in Datadog to measure the ratio of good events to total events. The SLO target is the percentage of how many events were "good" in a given time period.
    - These are also called **Metric-Based SLOs**.
2. **By Monitor Uptime**: References one or more existing monitors. The SLO target is the percentage of time monitors are in an `OK` state. This is most useful for time-based data.
    - These are also known as **Monitor-Based SLOs**.

Your SLO for homepage load times will be measured by Monitor uptime, because latency is a time-based metric. It will look like this:

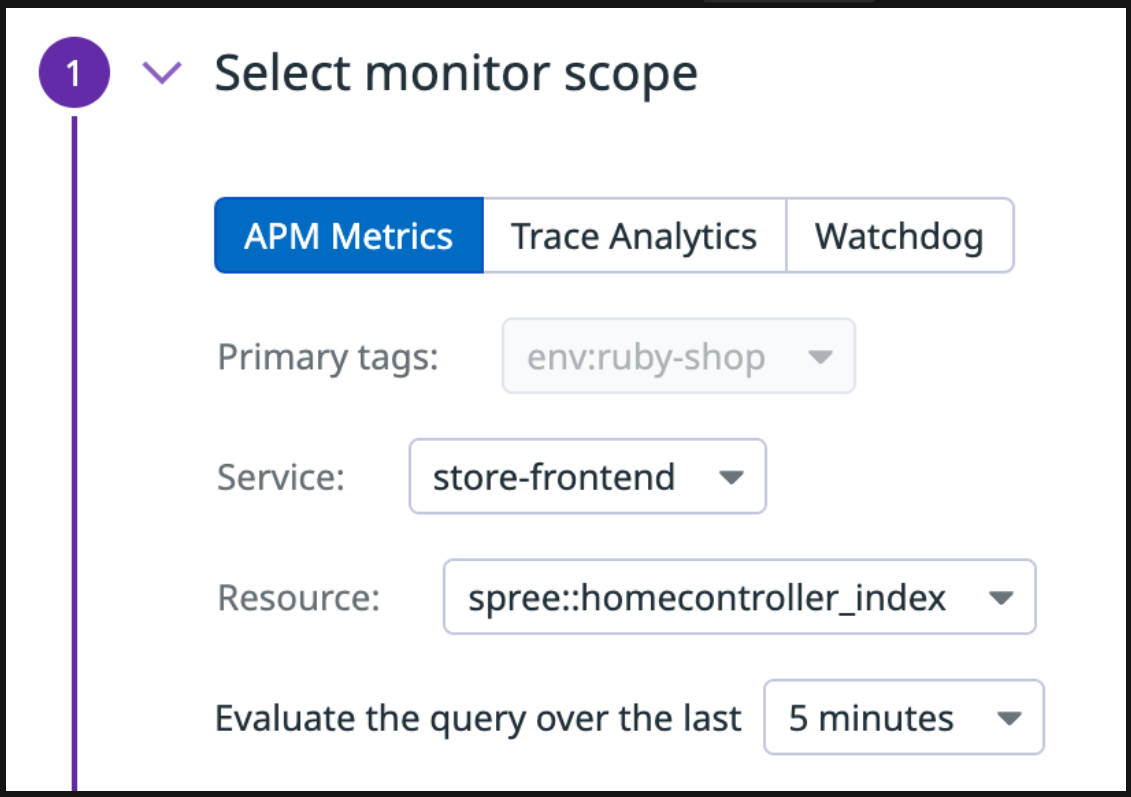
`"Over the past 7 days, 99% of the time the p99 latency of a home page request should be less than 5 seconds."`

That is, for 99% of the past 7 days, 99% of home page requests should experience less than 5 seconds of latency.

Our SLO will be Monitor-based, you'll need to create a new Monitor for homepage latency. The Monitor will then be used to construct your SLO.

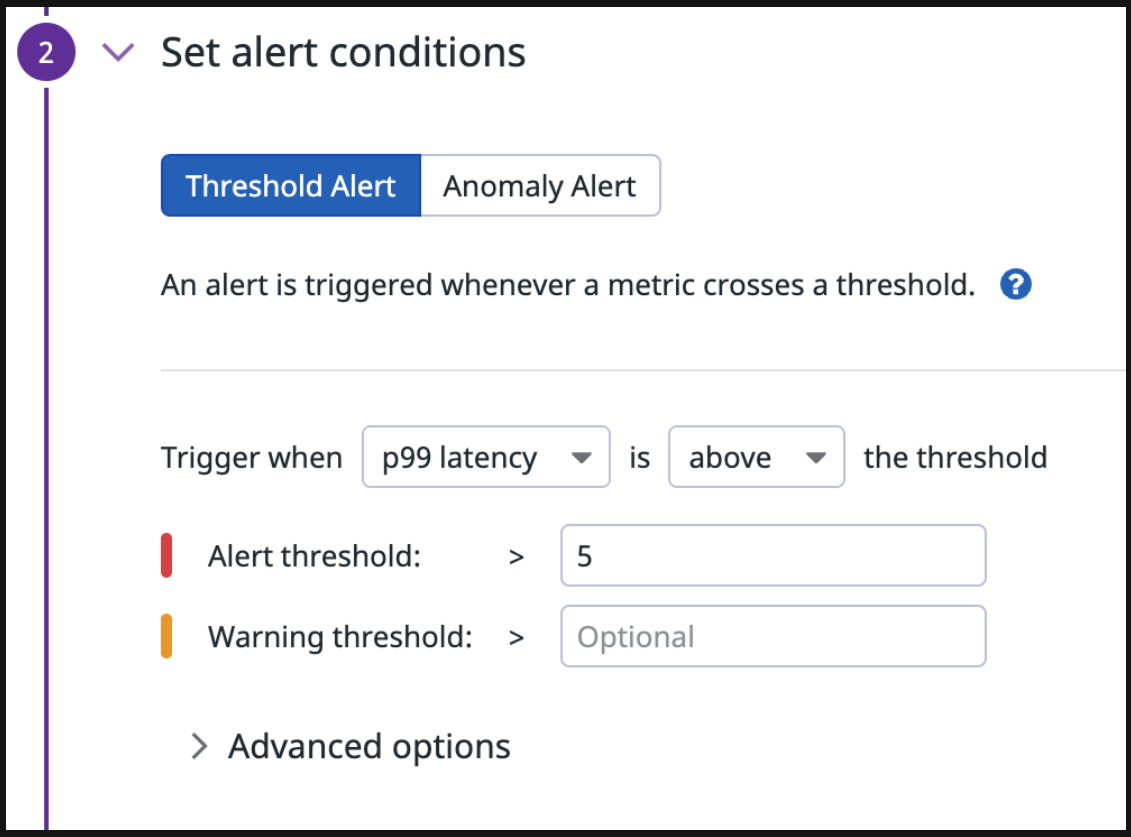
1. In Datadog, navigate to **Monitors > New Monitor**.
2. Select **APM** from the list on the left. This will open a form.
3. Under **Select monitor scope**, select **APM Metrics**.
    - In the **Primary tags** field, select `env:{yourenv}` .
    - For **Service**, select `service-yourservice` .
    - For **Resource**, select `route/endpoint` .

- For **Evaluate the query over the last**, leave the default `5 minutes`.



4. Under **Set alert conditions**, select `Threshold Alert`.

- Set the alert criteria to **Trigger when** `p99 latency` **is** `above` **the threshold**.
- For the **Alert threshold** enter `5`.
- Leave the optional **Warning threshold** blank.



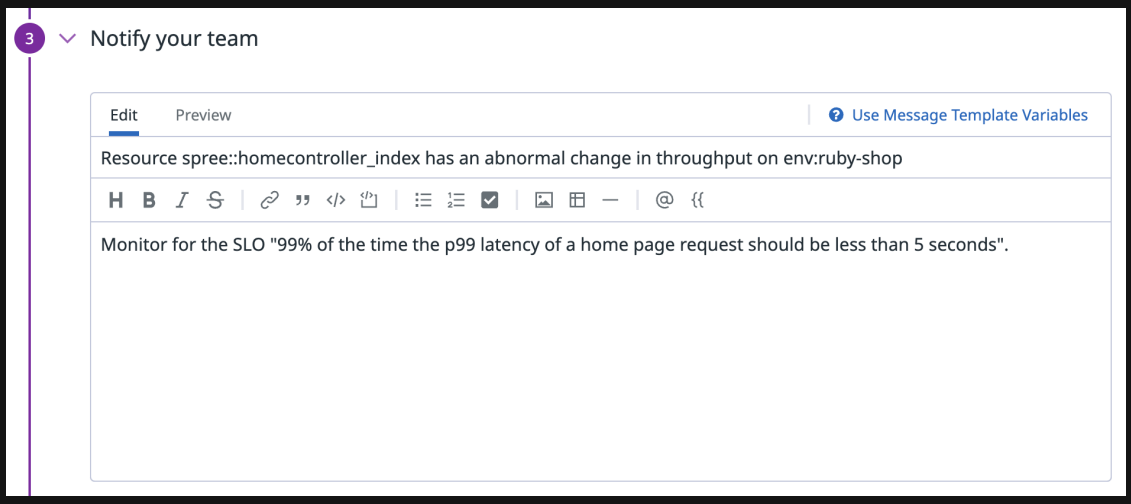
5. In the **Notify your team** section you will see a text editor.

- Pick a Good Name Something like:

" `route/endpoint` *has an abnormal change in throughput on* `env:{yourenv}` "

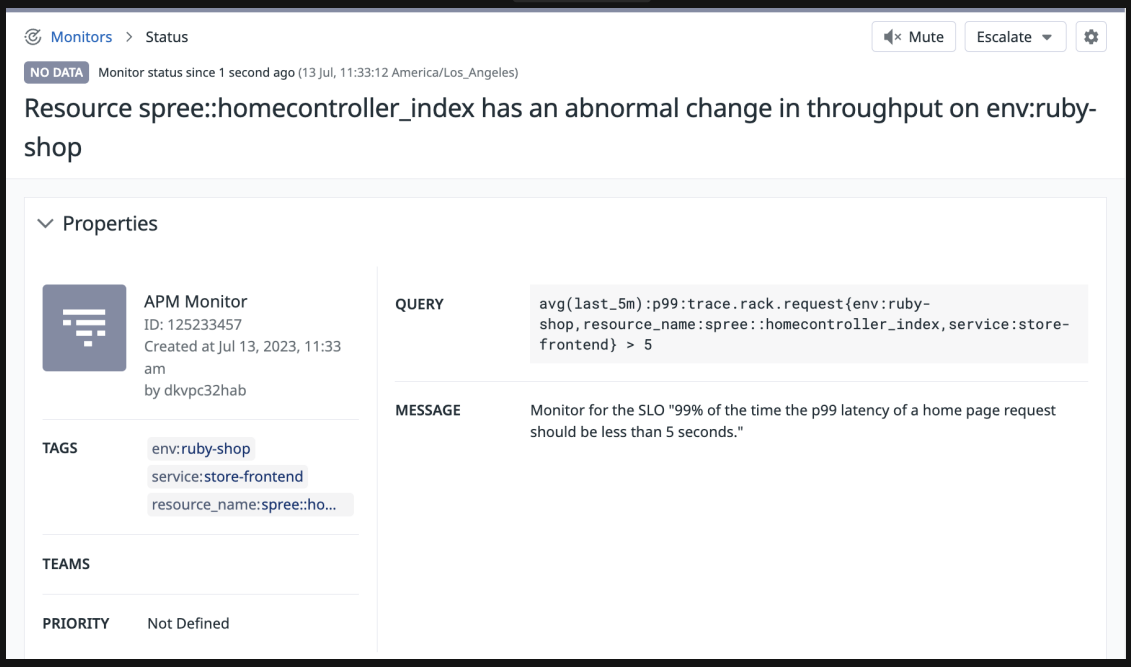
- Replace the message in the body of the text editor with something like:

```
Monitor for the SLO "99% of the time the p99 latency of a home page request should be less than 5 seconds".
```

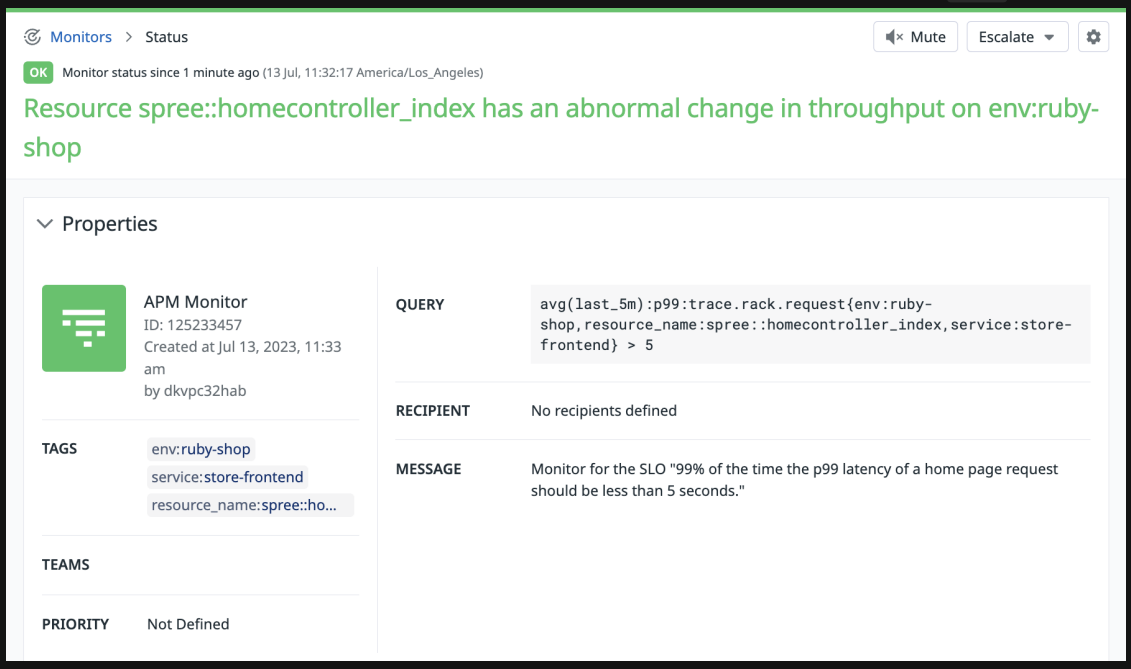


6. Adjust all remaining settings and fields as-needed. Click **Create**. You'll navigate to a detail page for the new monitor.

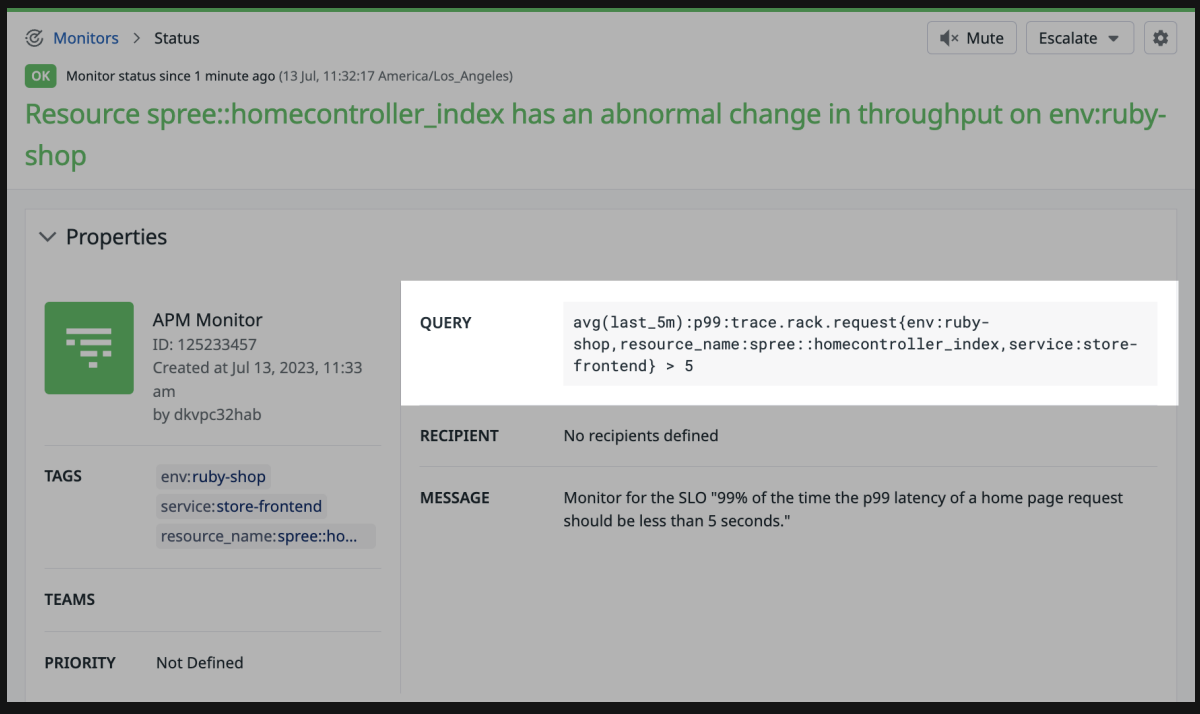
- At first, the Monitor will have a status of `NO DATA` .



- After several moments it should update to an status more than likely it'll be `OK`.



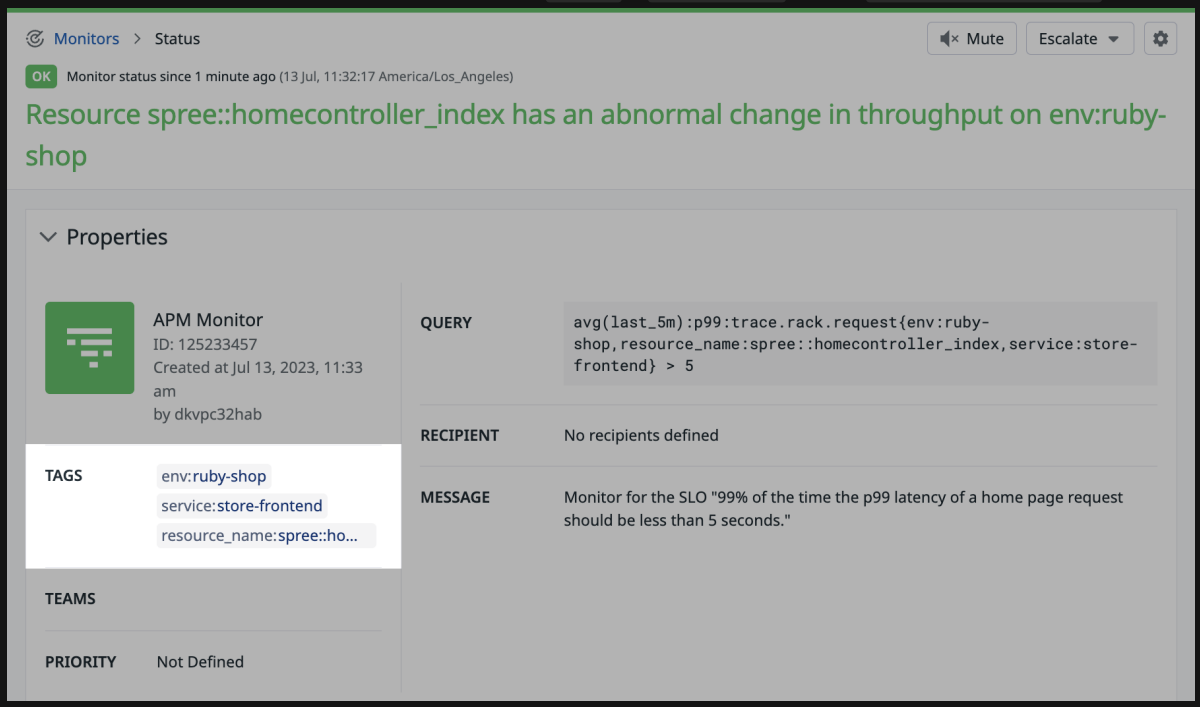
7. On the detail page for your new Monitor, notice the metric query in the **Query** section.



- The `trace.rack.request` metric is scoped to the respective `env`, `service`, and `resource_name`.

> **Note**: For more information on distribution trace metrics, review the DDSketch-based Metrics in APM documentation.

8. Notice that **Tags** have also been added for `env`, `service`, and `resource_name`.



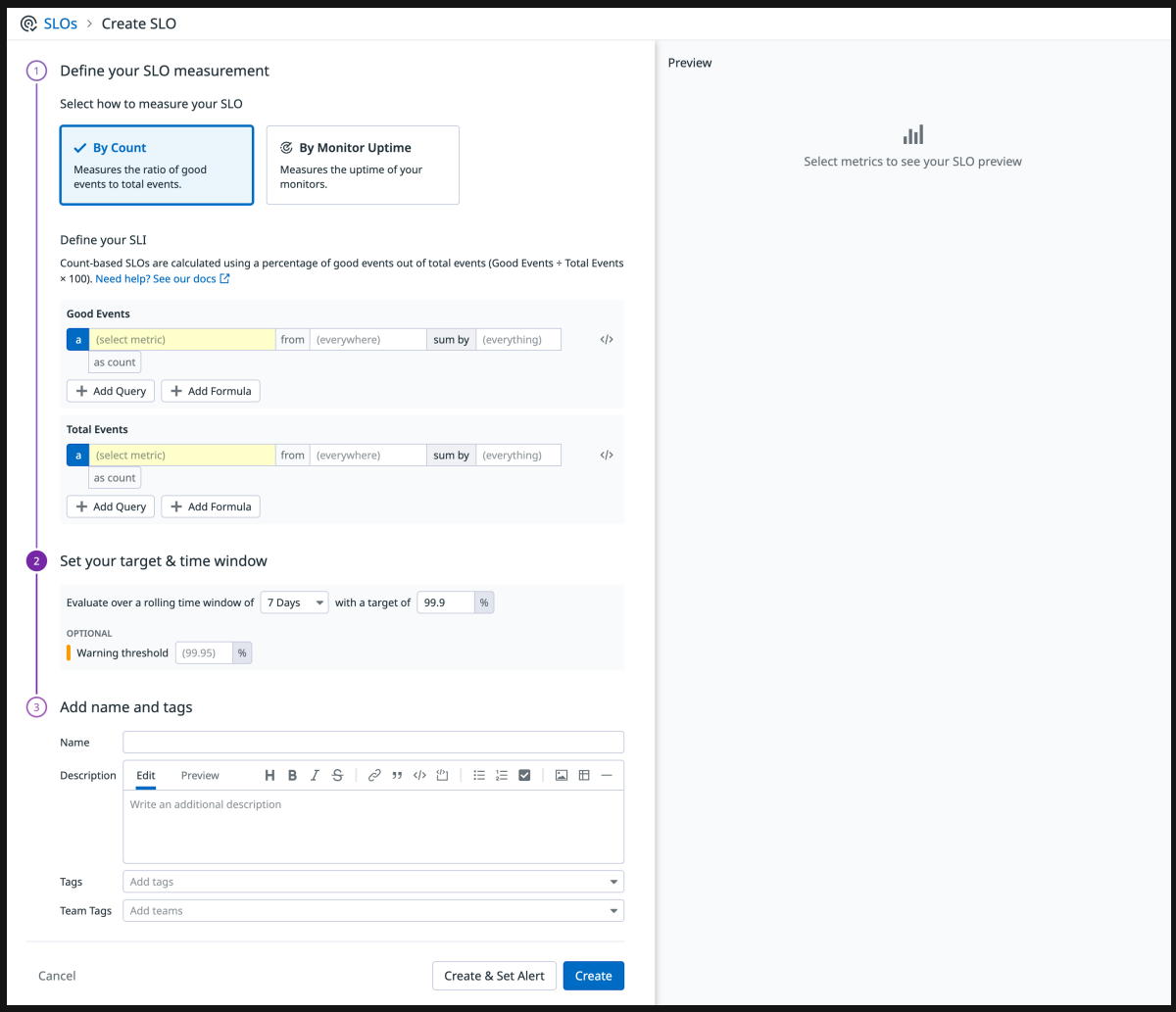
- You can use these tags to search the monitors in the **Manage Monitors** list. They also link the monitor to corresponding APM Service and Resource pages.

With your monitor in place, you can now use it to create your Monitor-based SLO for your page latency issue.

Creating a Monitor based SLO (in practice)

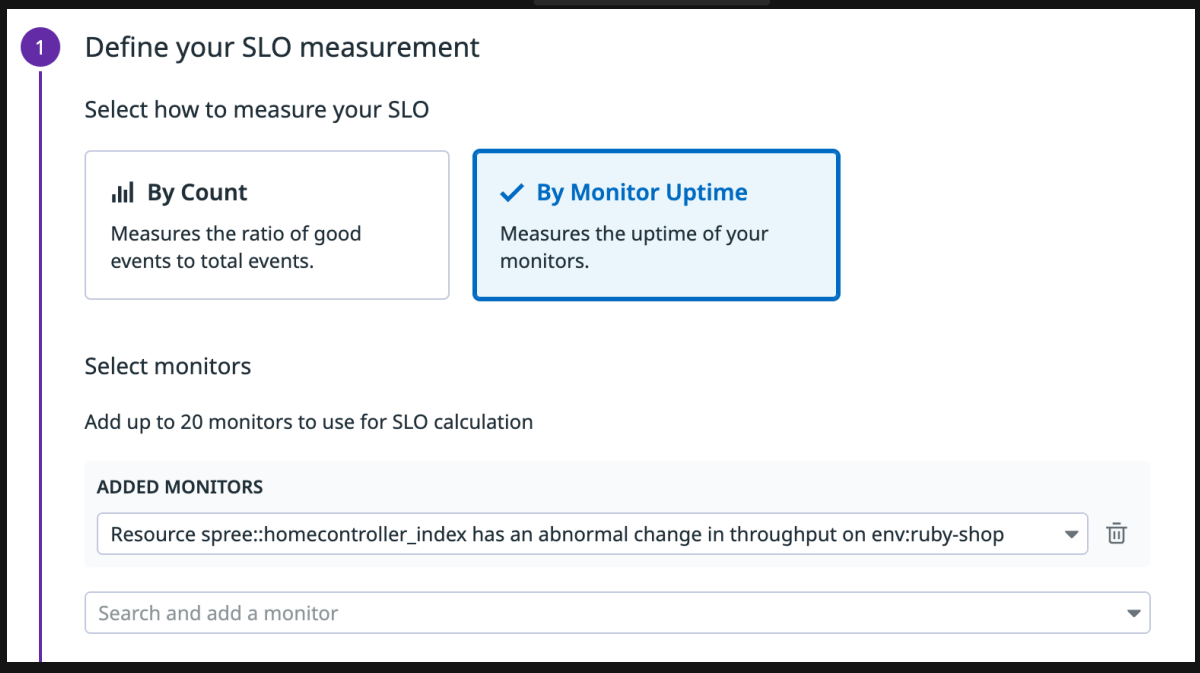
1. In Datadog, navigate to **Service Mgmt > Services > SLOs**.

2. Click **New SLO** in the upper-right. This will open a form.



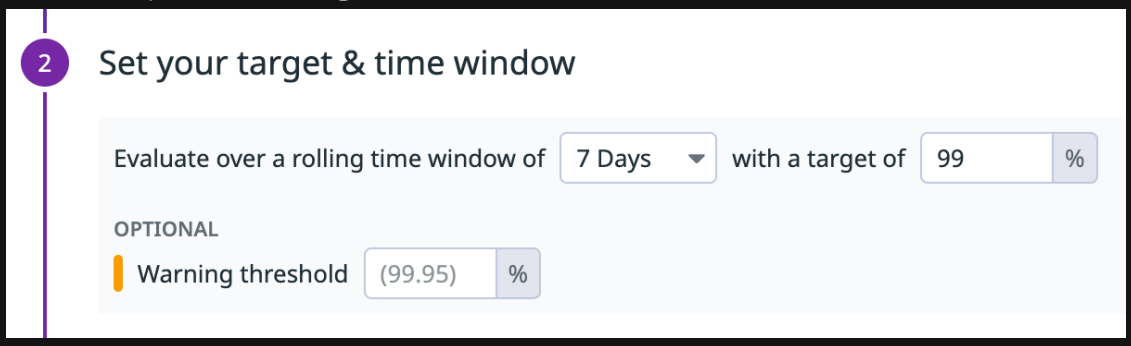
3. Under **Define your SLO measurement**, select **By Monitor Uptime**. This will reveal a **Select monitors** dropdown menu.

4. In the **Select monitors** dropdown, select the monitor you created above: " `route/endpoint` *has an abnormal change in throughput on* `env:{yourenv}` "



5. Under **Set your target & time window**, update the statement to read: **Evaluate over a rolling time window of** `7 days` **with a target of** `99%` .

    - Leave the optional **Warning threshold** blank.



6. Under **Add names and tags**, enter the following **Name**: `Page P99 Latency`

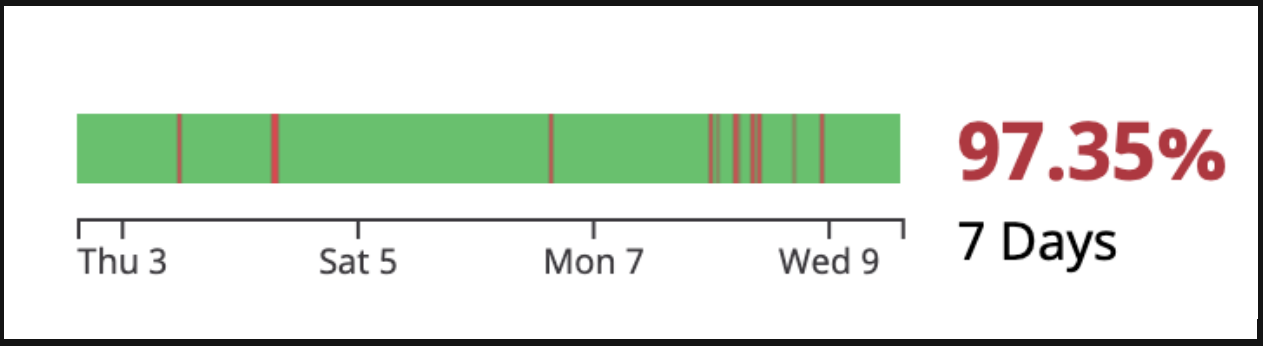
    Under **Tags**, select each of the following:

- `env:`
- `service:yourservice`
- `route/endpoint`

Add **Team Tags** as-needed.

Click **Create**. You will navigate to a detail panel for your new SLO.

The **History** section is might be empty, but this is expected if this is a new `route/endpoint` .



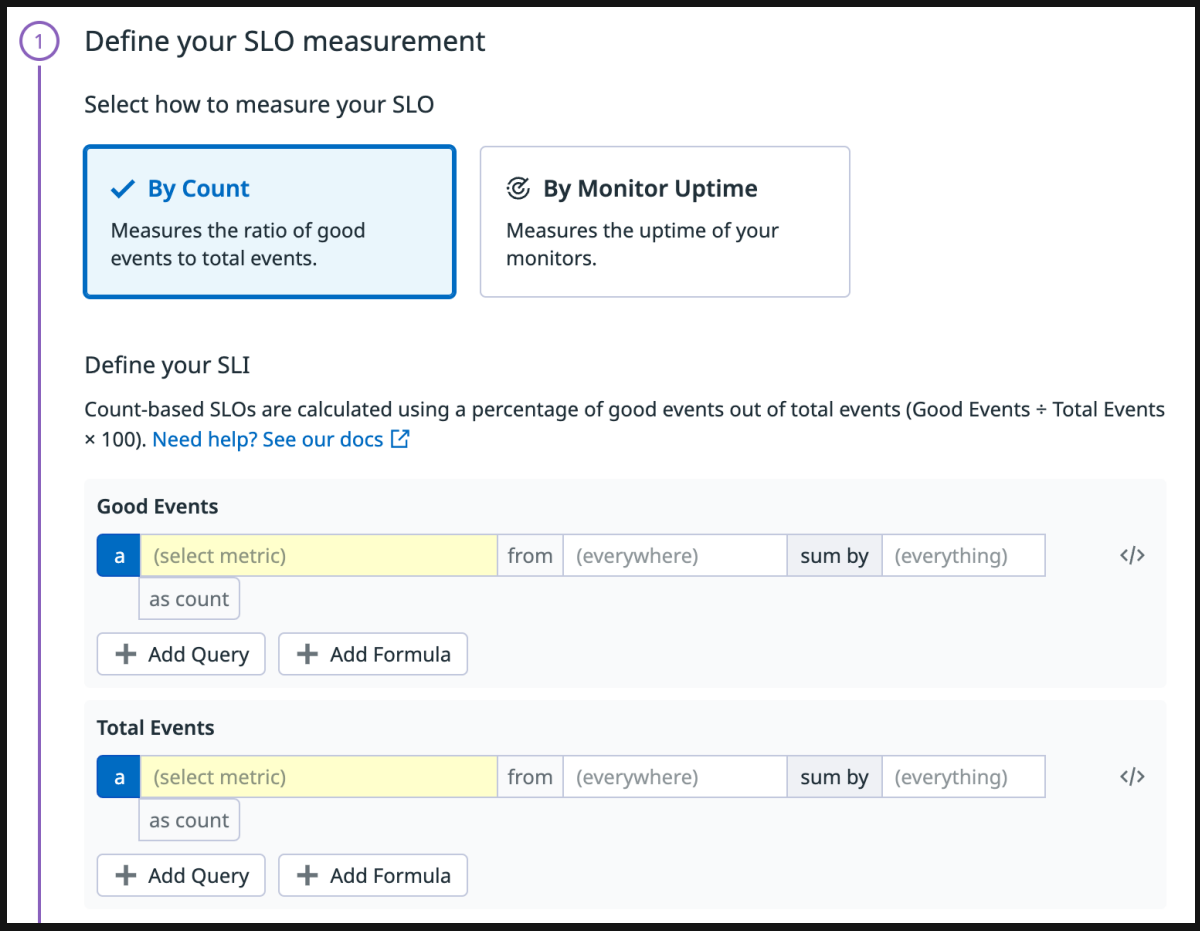
> *SLO replay*: Monitor-based or "By Monitor Uptime" SLOs also have a feature called SLO Replay. SLO Replay backfills SLO statuses with historical data pulled from underlying monitors' metrics and queries. This means if you create or update a Monitor-based SLO it will be populated with relevant historical data when available.

Creating a Metric based SLO (in practice)

As determined earlier, the second Storedog SLO is: `"Over the past 30 days, 99% of requests to the cart will be successful."` You'll create this Metric-based SLO next.

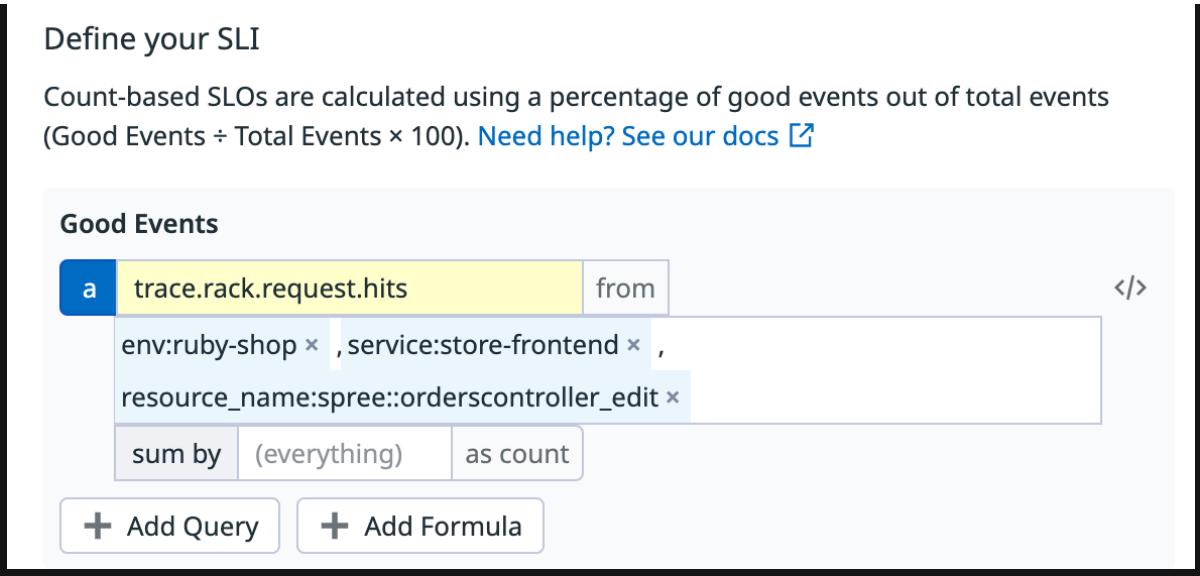
1. In Datadog, return to **Service Mgmt** > **Services** > **SLOs** and click **New SLO** in the upper-right.

2. Under **Define your SLO measurement**, select **By Count**. Below this, notice the **Define your SLI** heading with two sets of form fields: one for **Good events** and another for **Total events**.

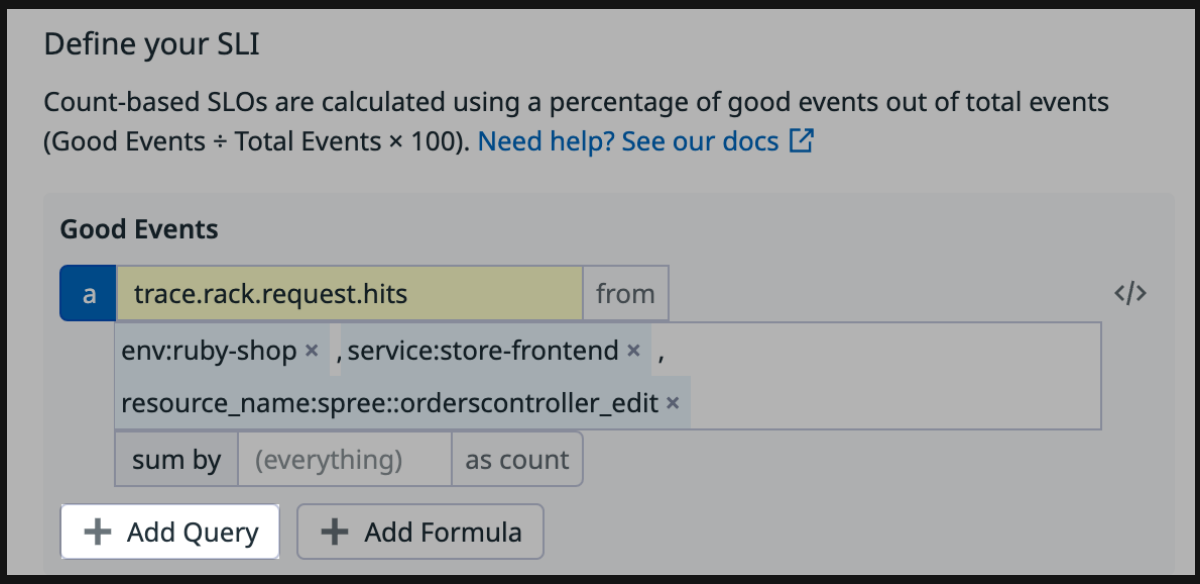


3. Under **Good events**, in the **a** field, insert `trace.rack.request.hits`. Then update each field as follows:

   - In the **from** field, insert each of the following:
     - `env:`
     - `service:your-service`
     - `resource_name`
   - Leave **sum by** blank.

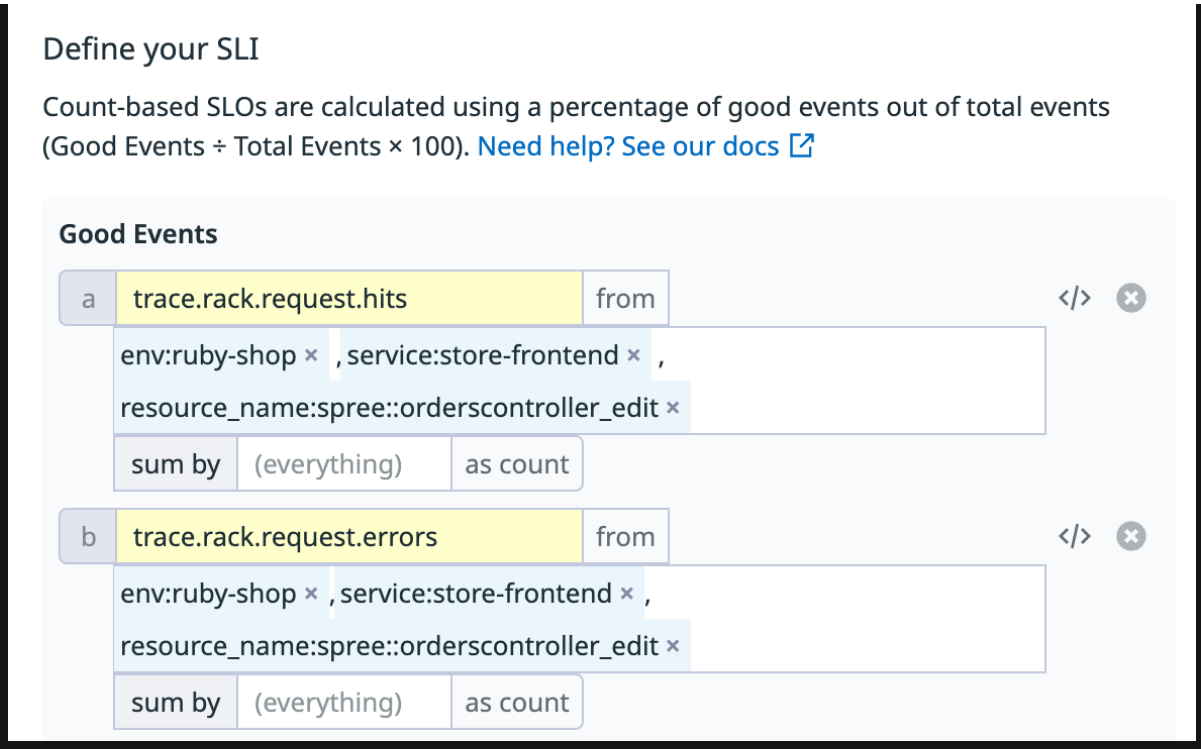


4. Below the fields you just completed, click **Add Query**.

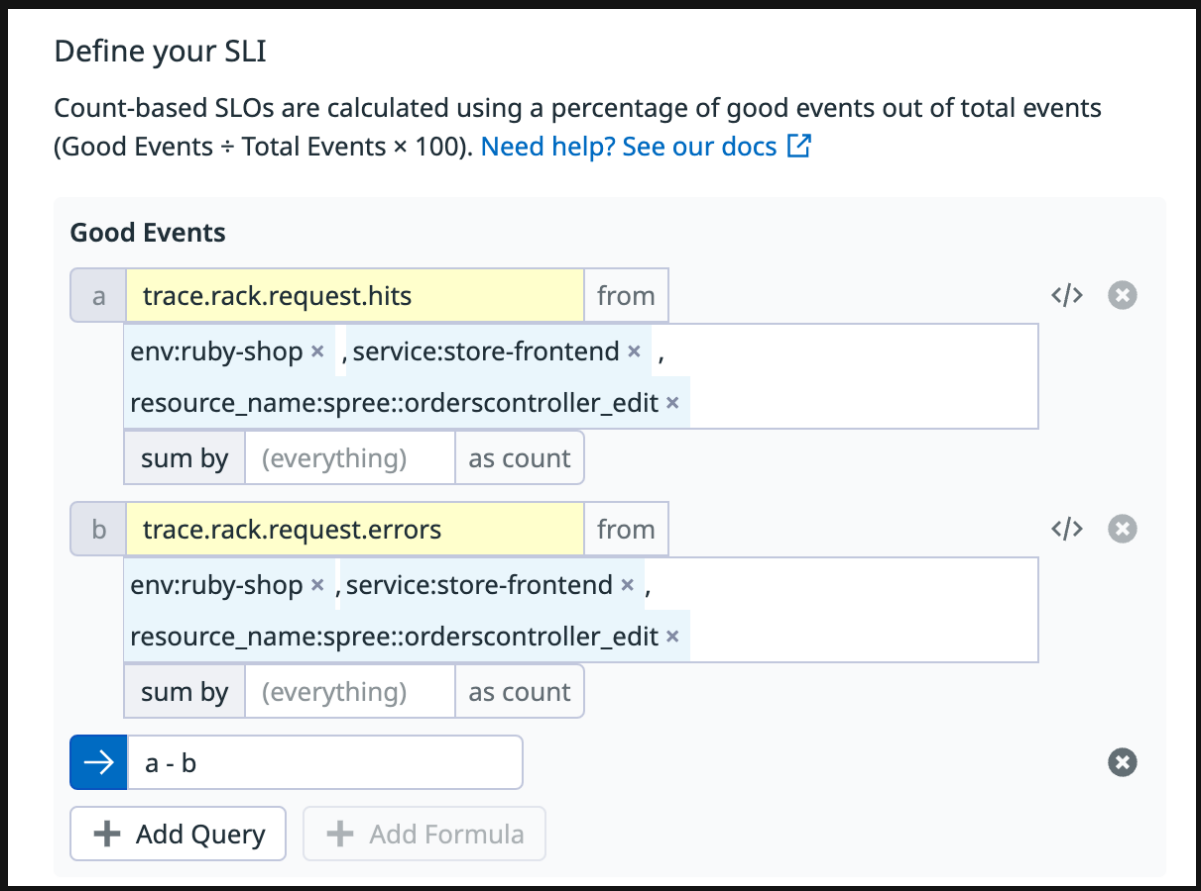


5. Additional fields will appear. Complete them as follows:

   - For **b**, replace the pre-populated value with `trace.rack.request.errors` .
   - In the **from** field, confirm the following are present:
     - `env:`
     - `service:your-service`
     - `resource_name`
   - Leave **sum by** blank.



6. At the bottom of **Good events**, locate the field reading `a + b`. Update this to `a - b`.

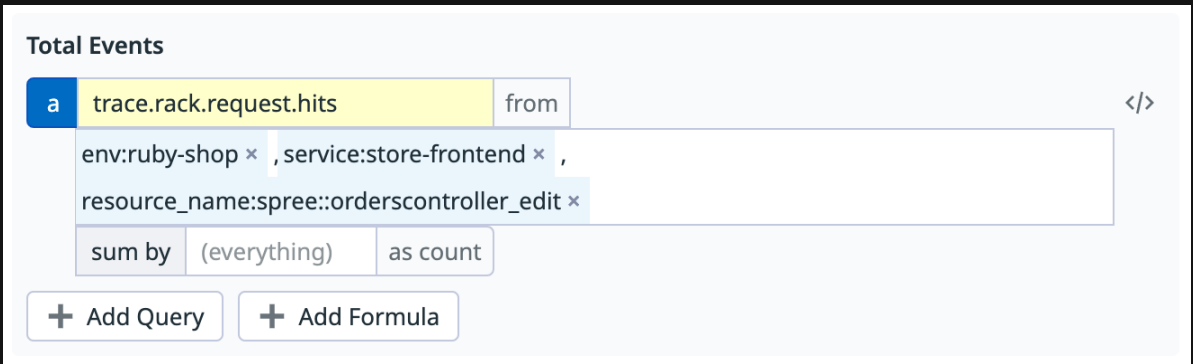


> **Note**: There is no direct metric for successful requests. Instead, there are metrics for total requests (`trace.rack.request.hits`) and for requests with errors (`trace.rack.request.errors`). You can calculate the number of successful requests by subtracting error-ridden requests from total requests. That is,

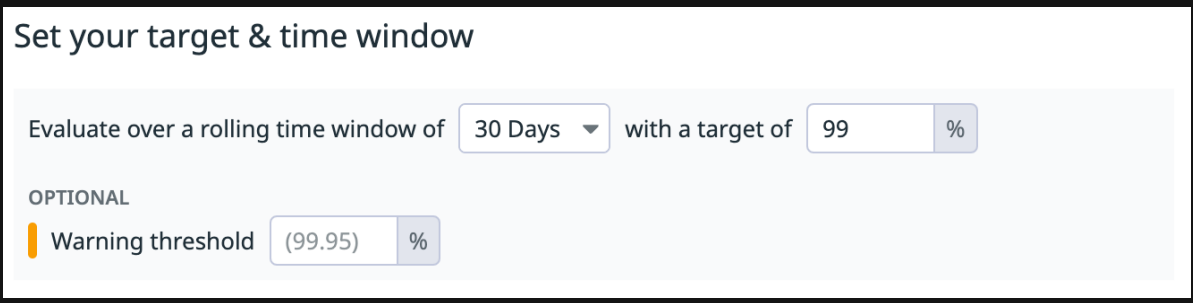
> `trace.rack.request.hits - trace.rack.request.errors`. Or, in this context, `a - b`.

7. Under **Total events**, update fields as follows:

   - **a**: `trace.rack.request.hits`
   - **from**: Add each of the following:
     - `env:`
     - `service:your-service`
     - `resource_name`
   - **sum by**: Leave blank.



8. Under **Set your target & time window**, update the statement to read: **Evaluate over a rolling time window of** `30 days` **with a target of** `99%`**.** Leave the optional **Warning threshold** blank.



9. Under **Add name and tags**, enter the following **Name**:

   `Comparing Requests`

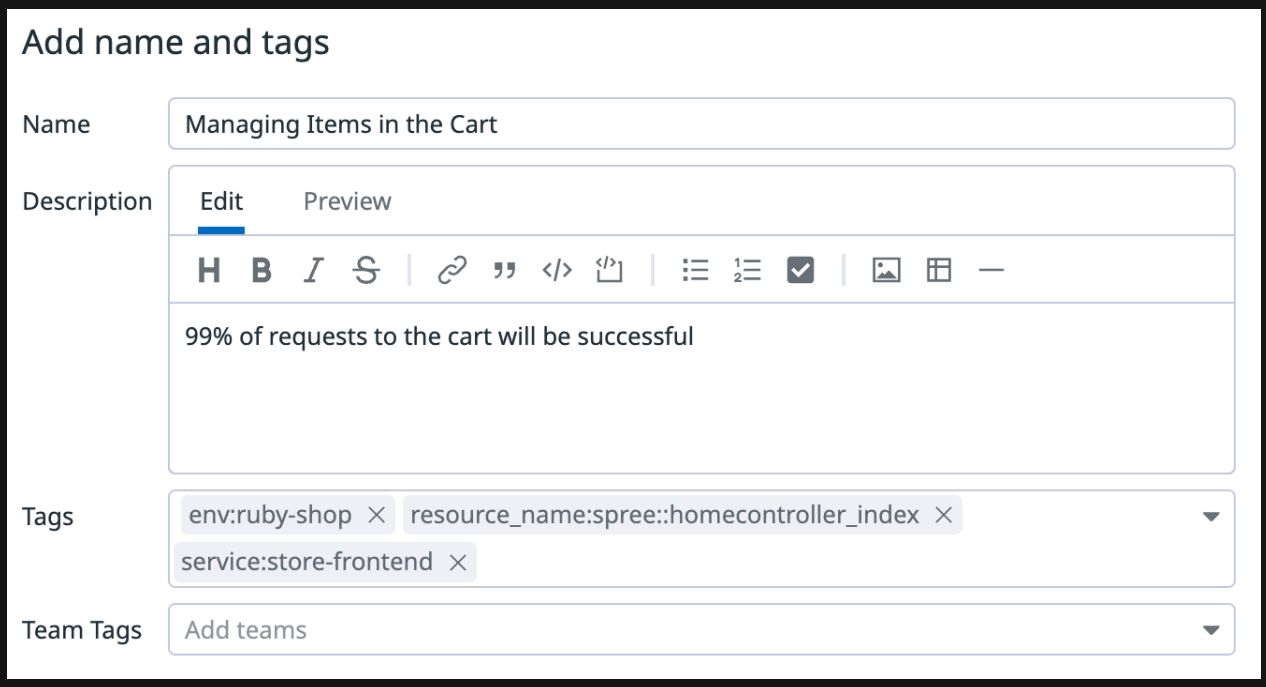
For Description, copy/paste the following:

```
99% of requests will be successful
```

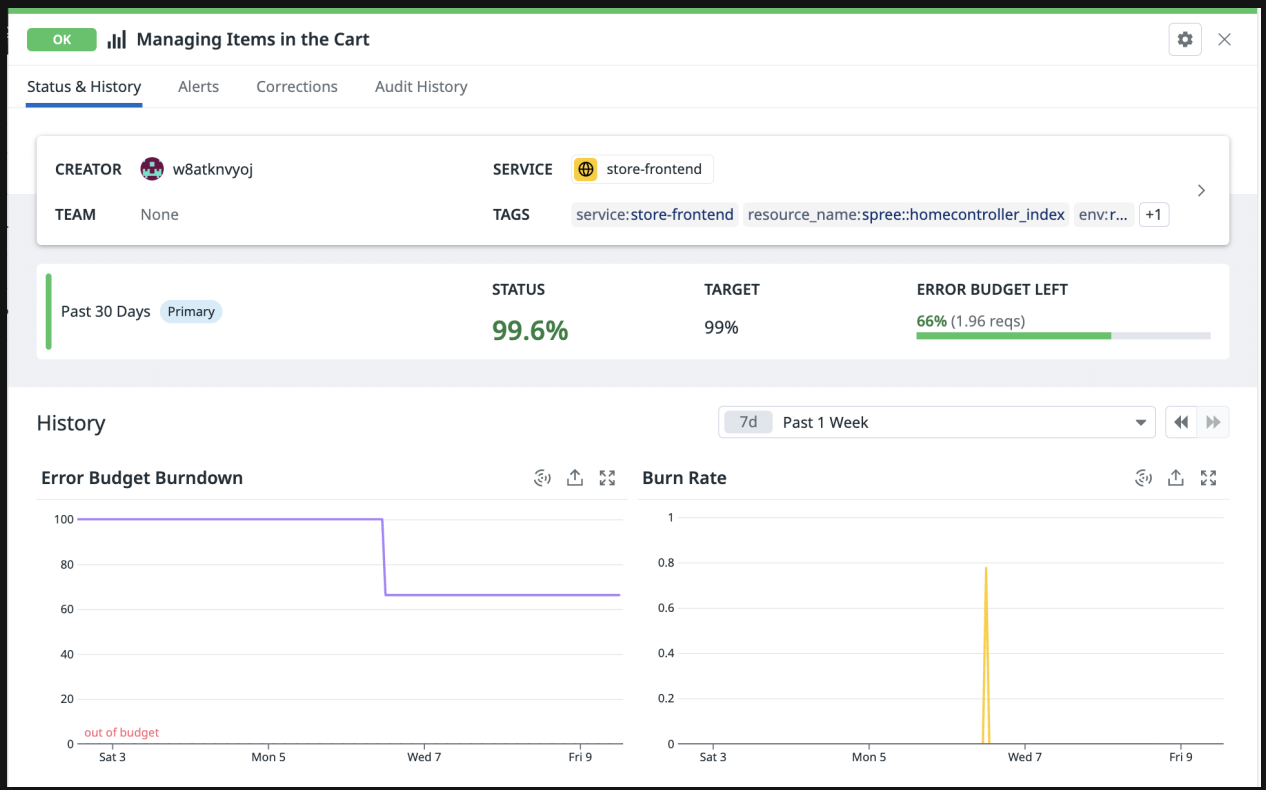
Select each of the following **Tags**:

- `env:`
- `service:your-service`
- `resource_name`

Add **Team Tags** as-needed.



Click **Create**. You will be directed to detail panel for the new SLO.

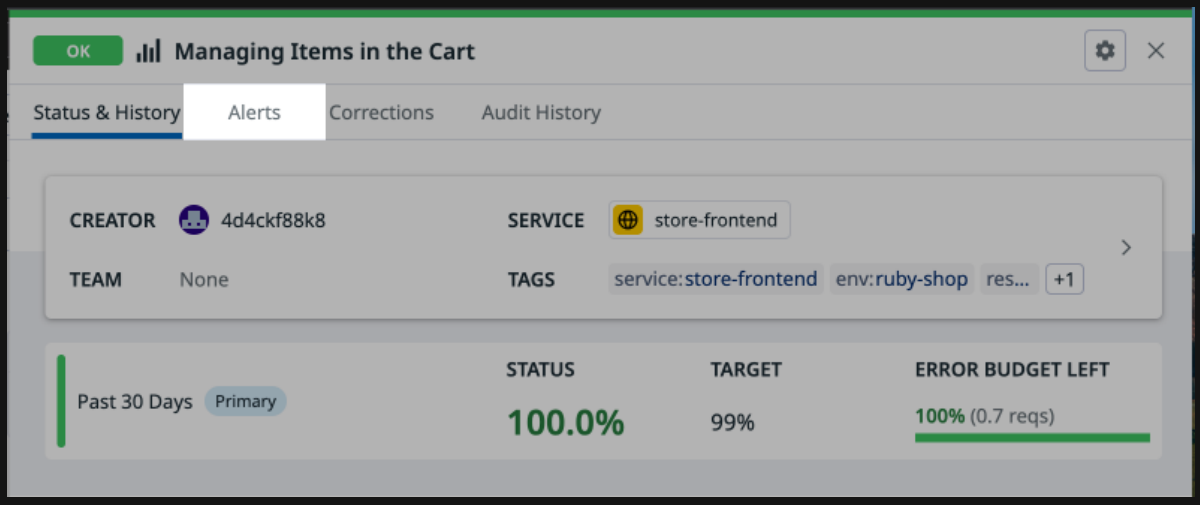


Creating an error budget

Error budget monitors notify you when an SLO's error budget has been consumed to (or past) a point of your choosing.

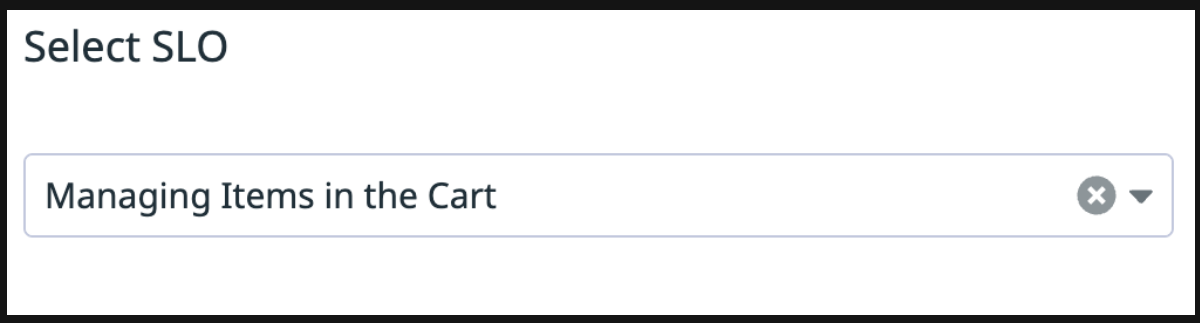
1. In the detail panel for the **SLO you want an error budget for** you just created, click the **Alerts** tab.

   > **Note**: You can access this page by navigating to **Service Mgmt** > **SLOs** and clicking **SLO you want an error budget for**.



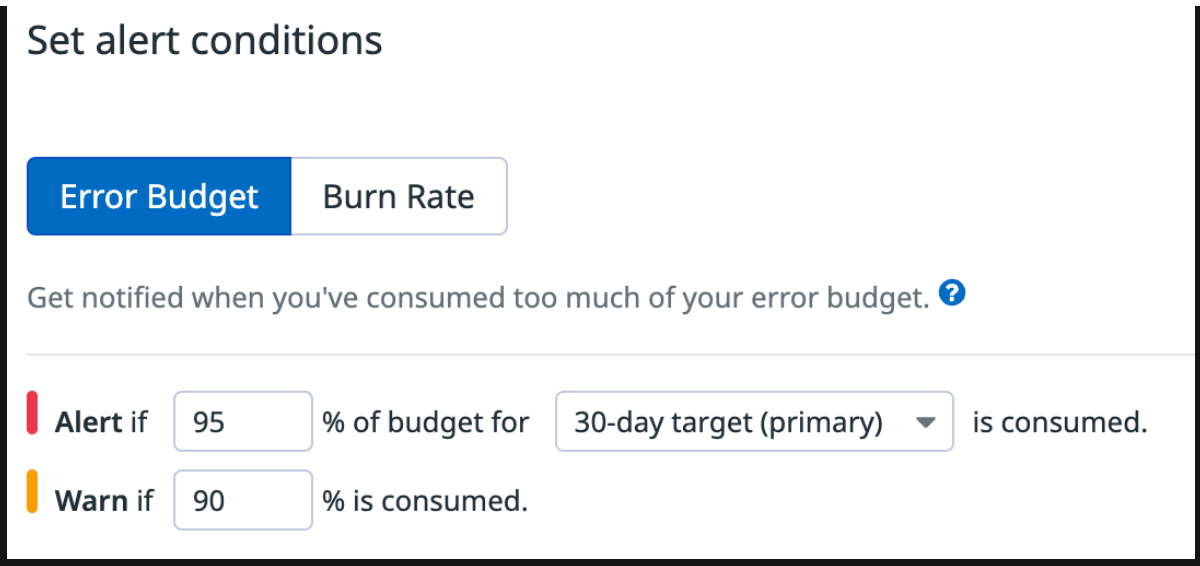
2. In the Alerts tab, click the **New Monitor** button on the right. You will navigate to a form.

3. Under **Select SLO**, confirm your `SLO you want an error budget for` SLO is listed.



4. Under **Set alert conditions**, select **Error Budget**.

   - Update the **Alert if** line to read: `95` **% of budget for** `30-day target (primary)` **is consumed**.
   - Update the error budget **Warn if** line to read: `90` **% is consumed**.



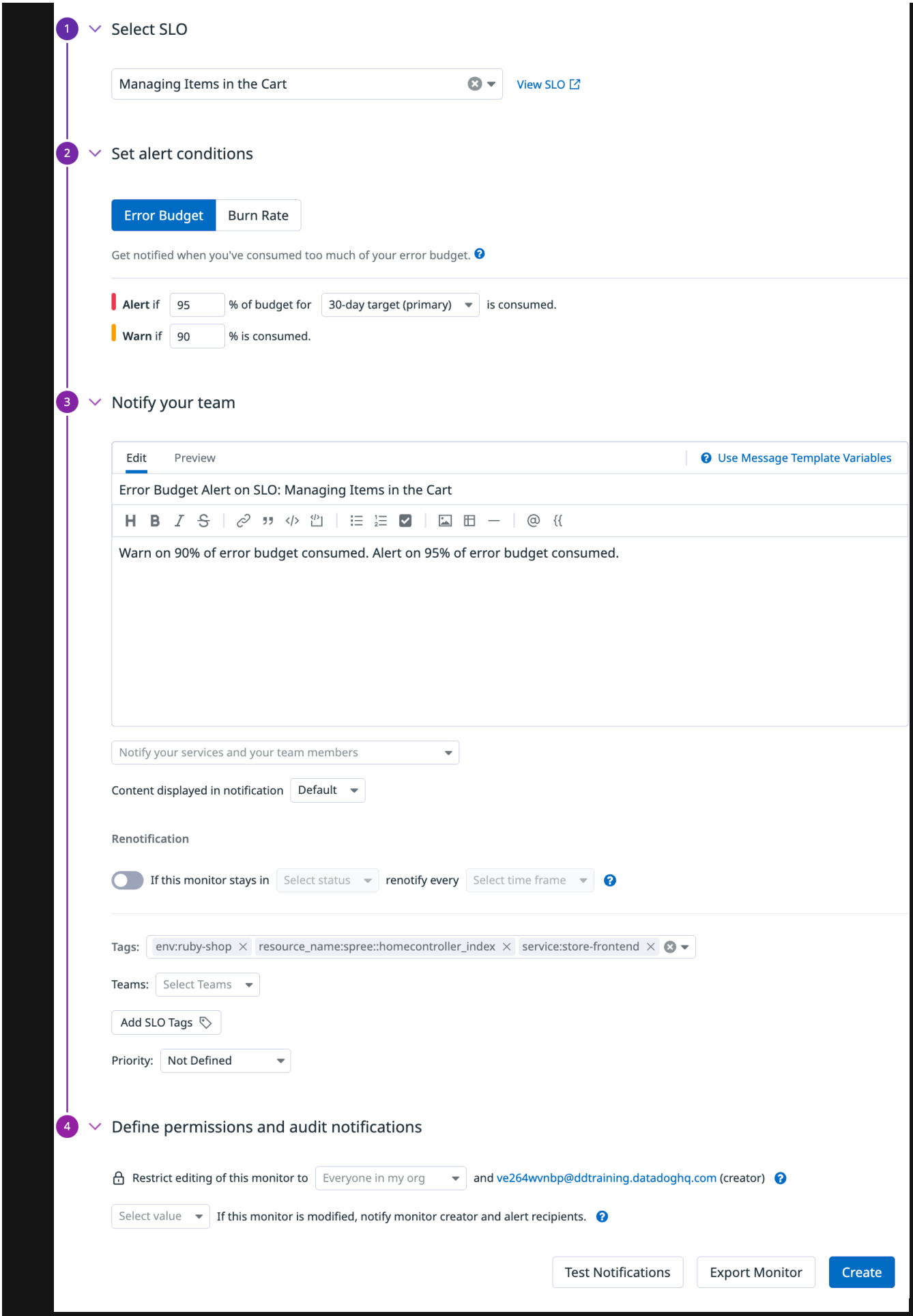
5. Under **Notify your team**, enter the following in the subject field:

```
Error Budget Alert on SLO: SLO you want an error budget for
```

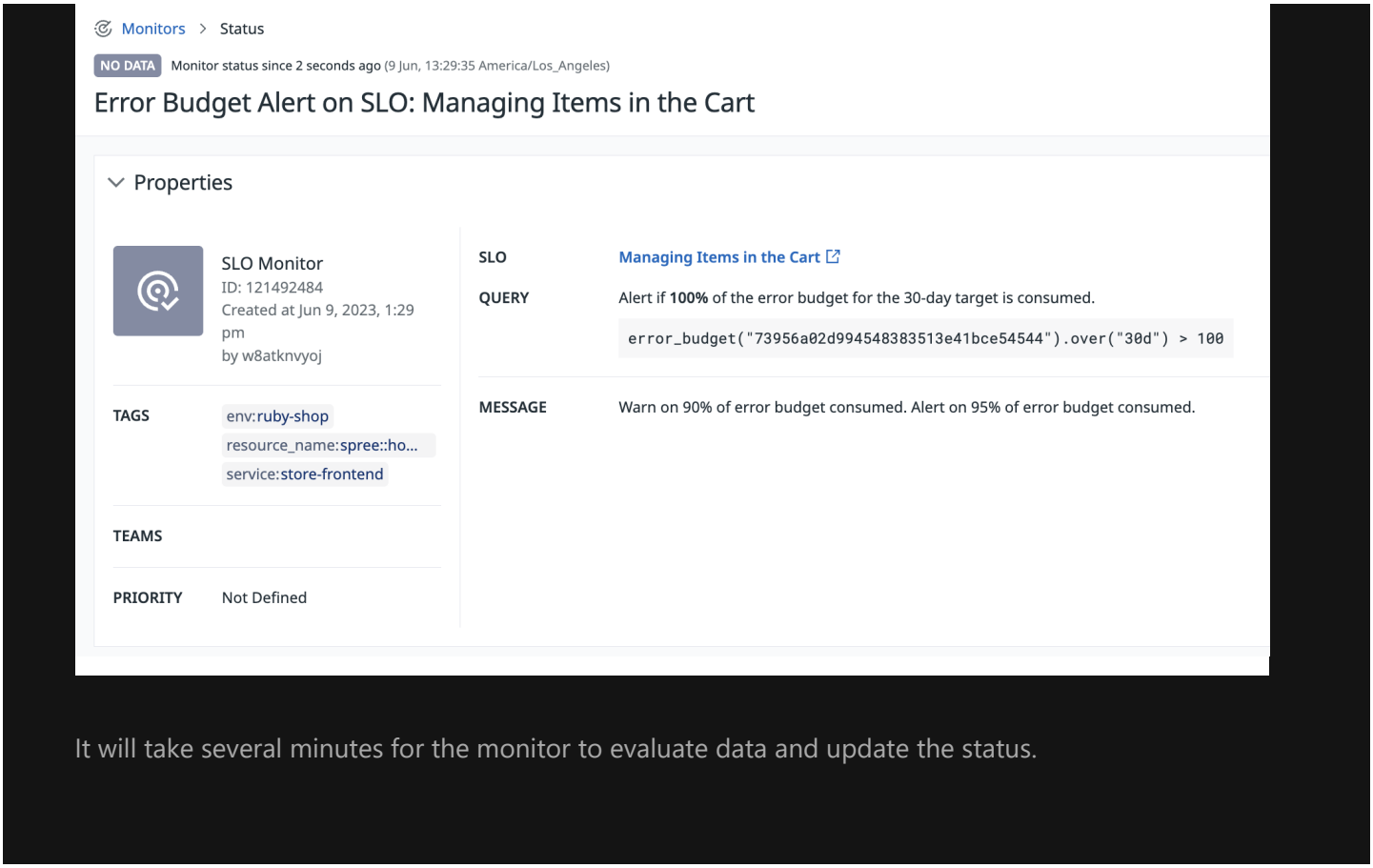
6. Add a message to the text body

7. Adjust default values as-needed in **Notify your services and your team members**, **Content displayed in notification** and **Renotification** fields.

In the **Tags** field (*NOT* the **Add SLO Tags** option), select each of the following: - `env:{yourenv}` - `service:your-service` - `resource_name` Leave remaining fields as-is. The completed form should look like similar this:



Click **Create**. You will navigate to a detail page for your new monitor. The status will initially read `NO DATA`:



It will take several minutes for the monitor to evaluate data and update the status.