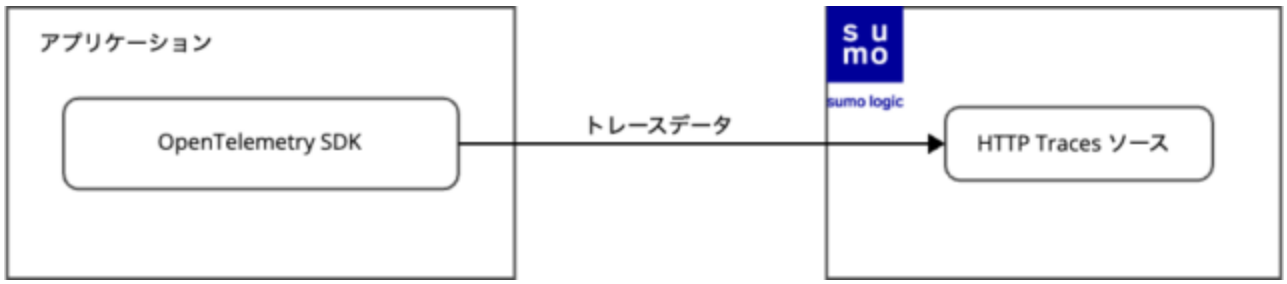
Trying out APM with Sumo Logic

dev.classmethod.jp/articles/sumo-logic-apm

酒井剛 June 18, 2023



Sumo Logic not only collects and analyzes logs, but also collects metrics and traces that can be used to measure application performance (APM).

This tracing allows you to observe the progress of requests and transactions processed by an application, and identify bottlenecks, bugs, and other causes that affect application performance.
In a system consisting of multiple services, such as microservices, a single request is processed across multiple services, making it difficult to understand the entire processing flow of the request.
Tracing allows you to track the flow of individual processes, visualize service dependencies and service latency, and help identify and investigate problem areas.

Sumo Logic Licensing

The licenses that allow tracing are Essentials (5GB/day), Enterprise Operations, and Enterprise Suite.
For PoC purposes, you can also use the Free or Trial licenses with limitations.

License and Feature Table:

Give it a try

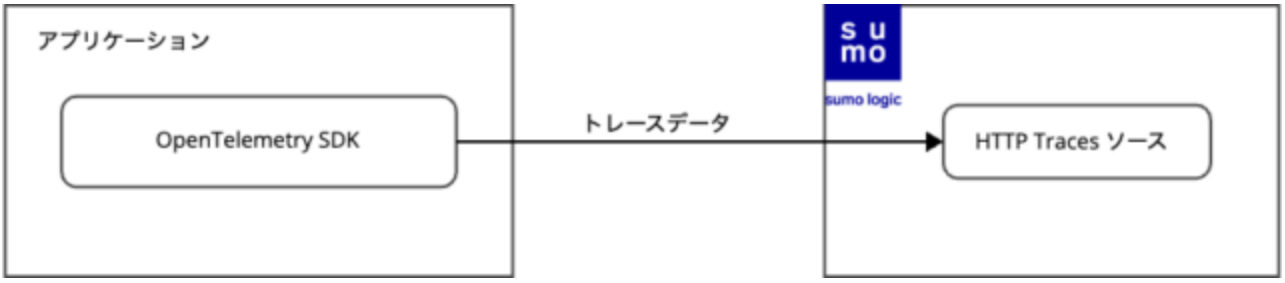
To see what a trace looks like with Sumo Logic, I'd like to first try using a sample application. The
official Sumo Logic documentation provides links to how to configure Sumo Logic and sample applications, so I'll use these as a reference.

This architecture

Sumo Logic uses OpenTelemetry to acquire traces.
There are three implementation methods for OpenTelemetry:
As you go down the list, the tradeoff between ease of implementation and ensuring the integrity of the data being sent, such as missing data.

- No collectors
- Agent type
- Gateway Type

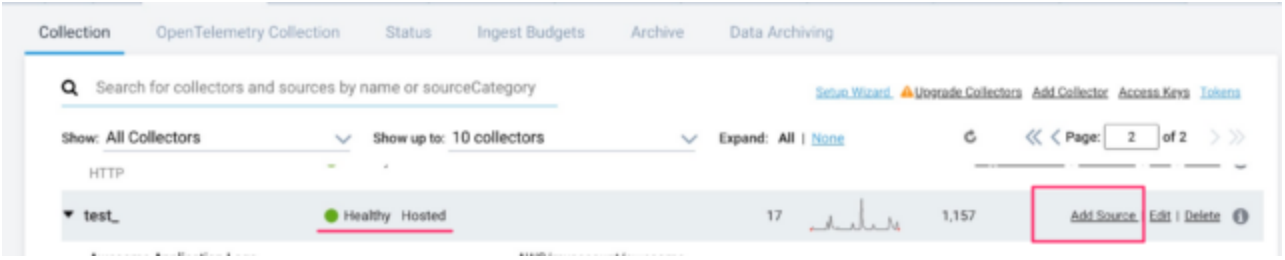
This time we will use the sample Docker image provided by Sumo Logic, which will have the following architecture without a collector.



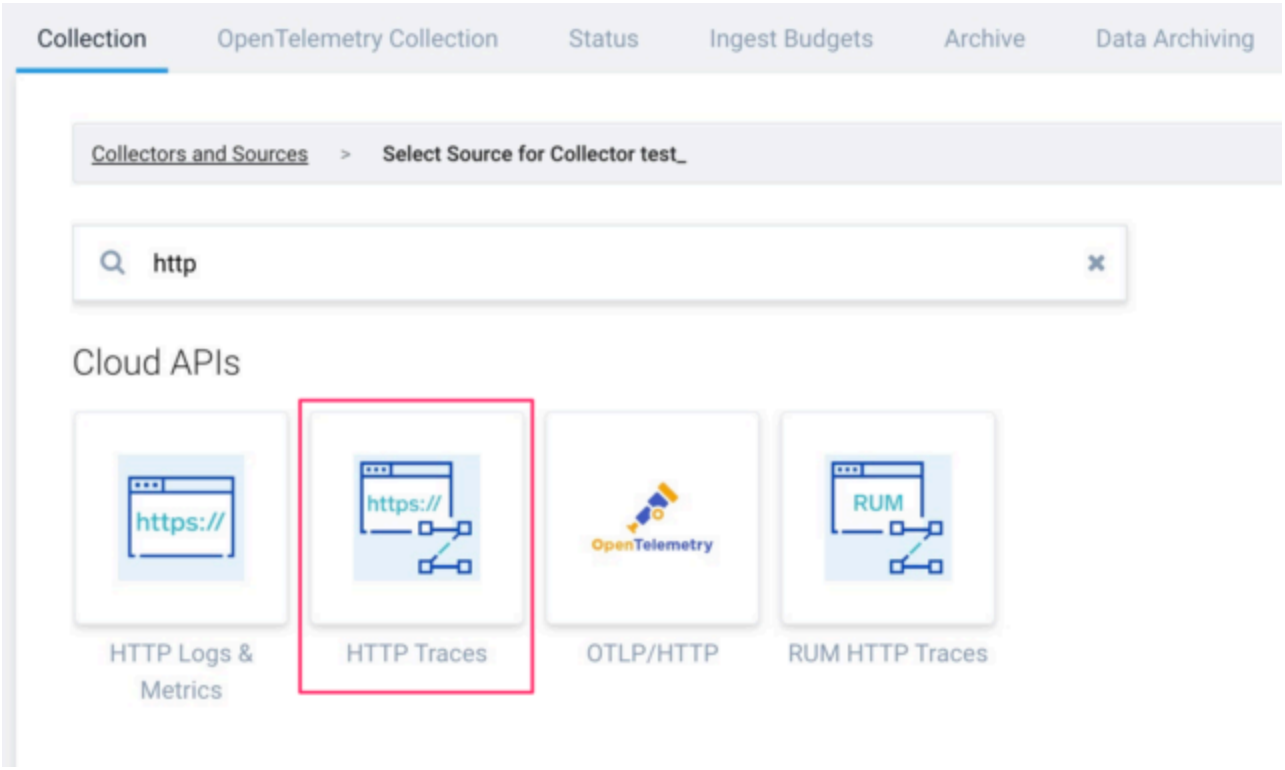
Sumo Logic Source Configuration

First, configure the source.
Configure a Sumo Logic HTTP Traces Source. This source accepts Zipkin JSON v2 or OTLP/HTTP spans and will receive trace data at an HTTP endpoint.

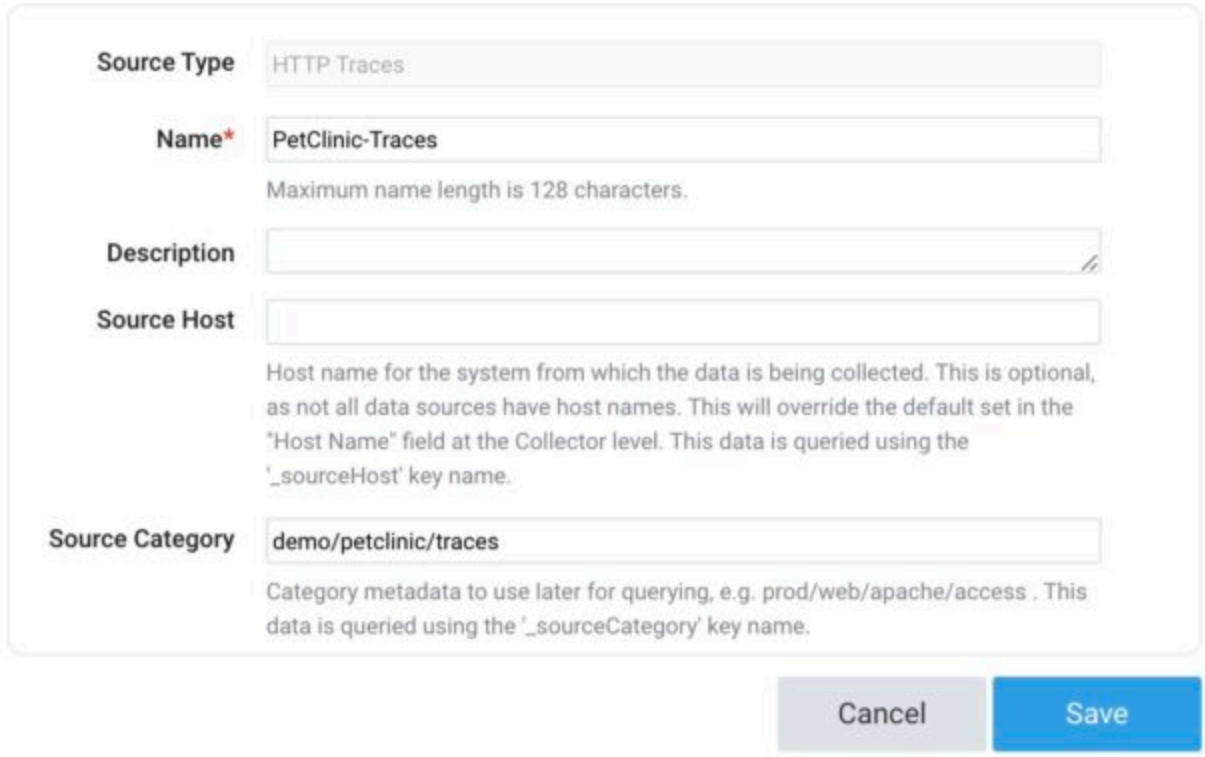
Create a source from a Hosted Collector in the Managed Data Collection. *If you have not created a collector before or if this is your first time using Sumo Logic, you will need to create a collector.



Here, select HTTP Traces.

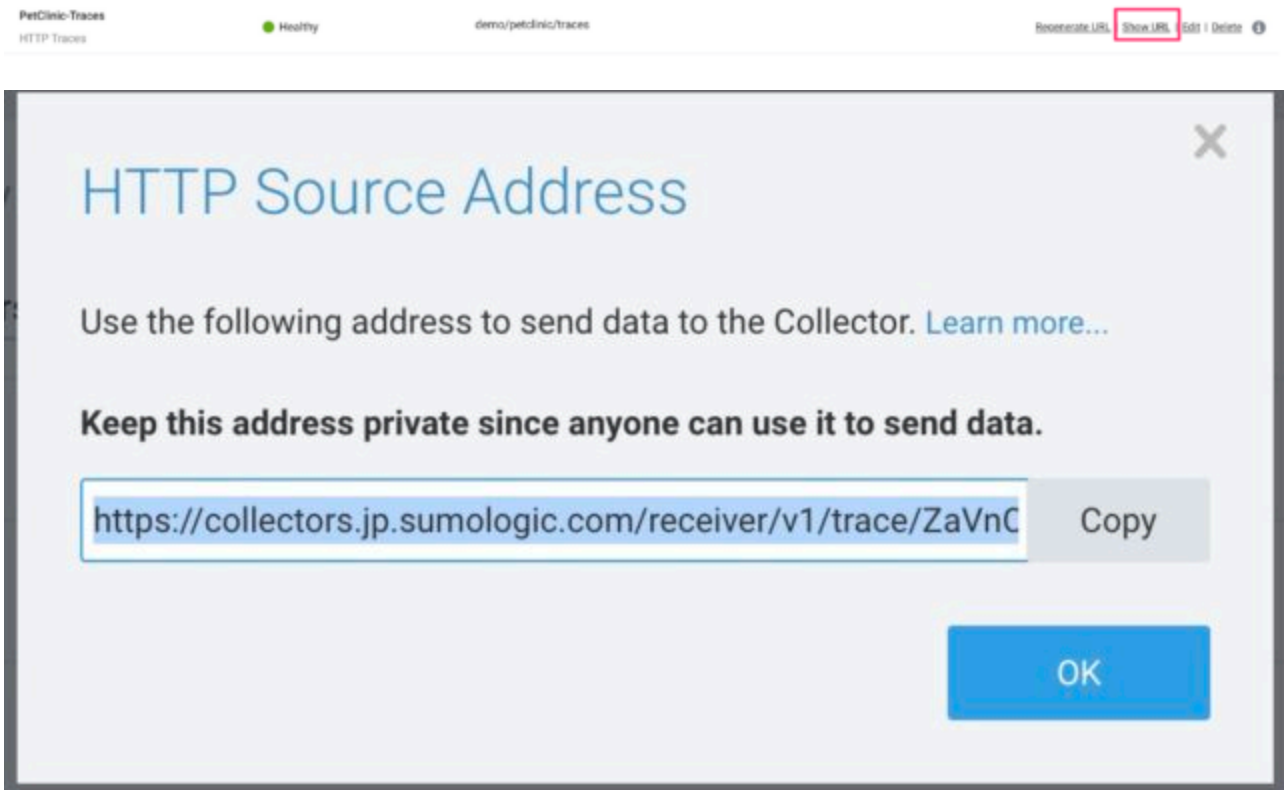


When setting up the source, you can enter an arbitrary source name and an arbitrary source category name.



For more information on source categories, please see:

After saving the settings, copy the HTTP endpoint for receiving the traces that will be displayed in the "Show URL" of the source. This will be used to set the destination for the application to send traces.



Deploying the sample application with Docker

This sample application will be launched with Docker. *The container did not start on a Mac with an M1 processor, so it is launched on a Windows PC with an Intel processor.

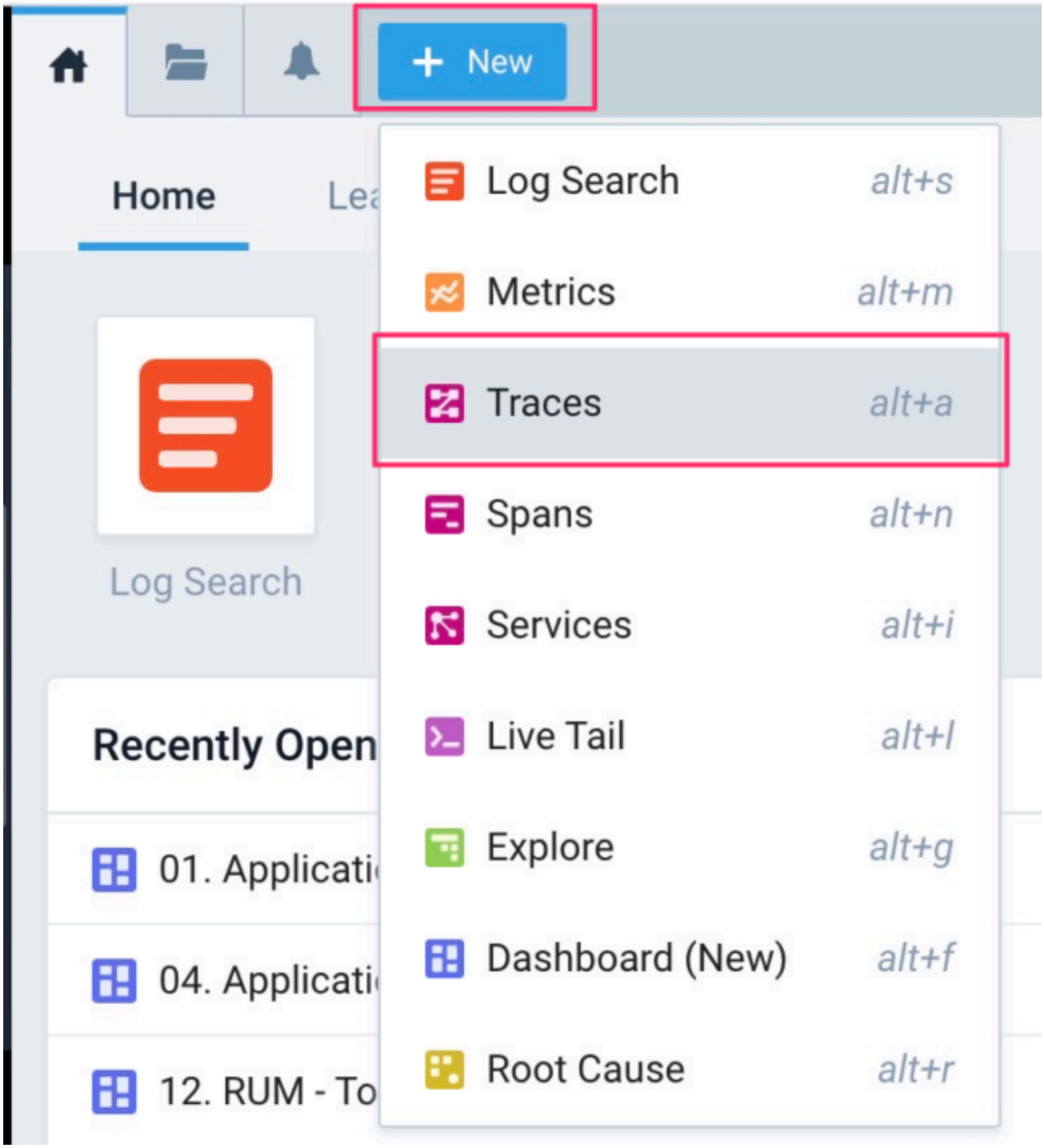
```
export OTLP_ENDPOINT_URL=<さっきコピーしたURL>

docker run --rm --name ot-petclinic -p 8080:8080 \
    --env JAVA_TOOL_OPTIONS="-javaagent:/agent/opentelemetry-javaagent.jar" \
    --env OTEL_SERVICE_NAME=petclinic-svc \
    --env OTEL_RESOURCE_ATTRIBUTES=application=petclinic-app \
    --env OTEL_METRICS_EXPORTER=otlp \
    --env OTEL_TRACES_EXPORTER=otlp \
    --env OTEL_EXPORTER_OTLP_ENDPOINT=${OTLP_ENDPOINT_URL} \
    --env OTEL_EXPORTER_OTLP_PROTOCOL=http/protobuf \
    public.ecr.aws/sumologic/opentelemetry-petclinic:latest
```

The Java application runs on port 8080 of localhost, so I'll try accessing it in various ways.

View trace data in Sumo Logic trace queries and dashboards

Trace Query

Trace data can be viewed from "Traces" under "+New".

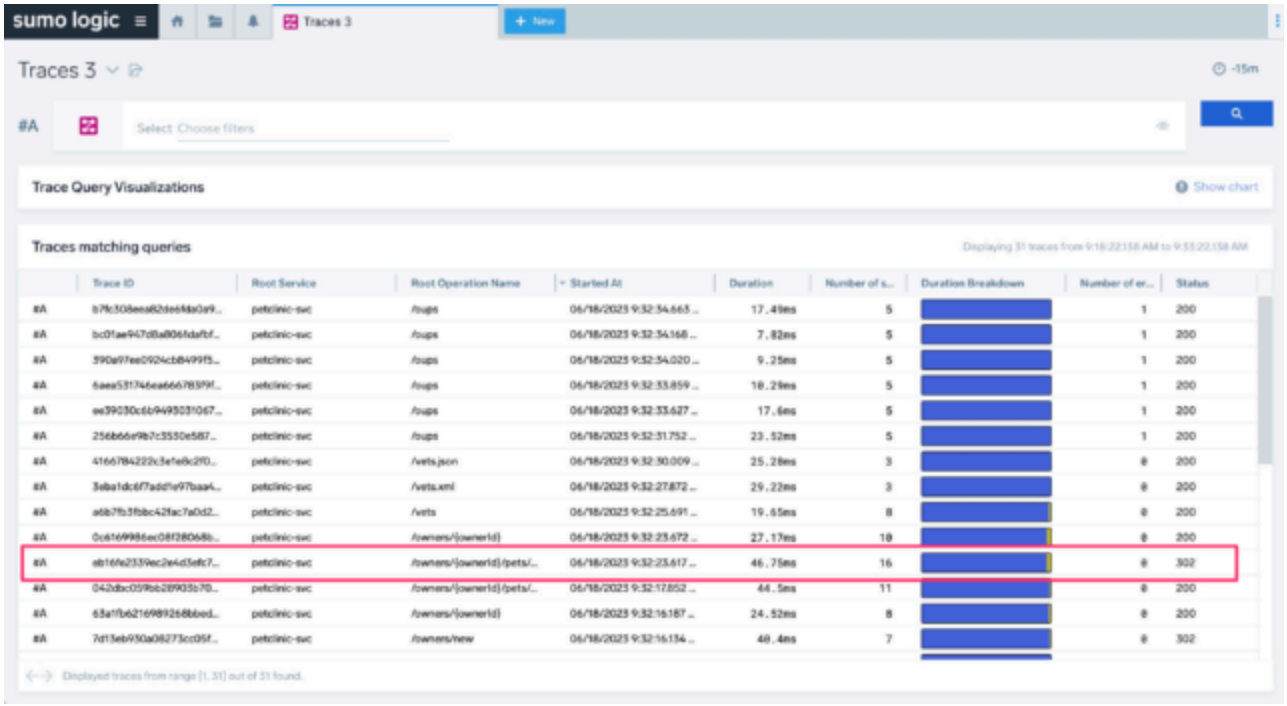


If you cannot find any data in the trace even after expanding the search time, try searching using the following command in the log search. If any errors have occurred, you can check them here.

```
_index=sumologic_system_events AND _sourceCategory=tracingIngest
```

If you see any error output, you can also check the FAQ here .

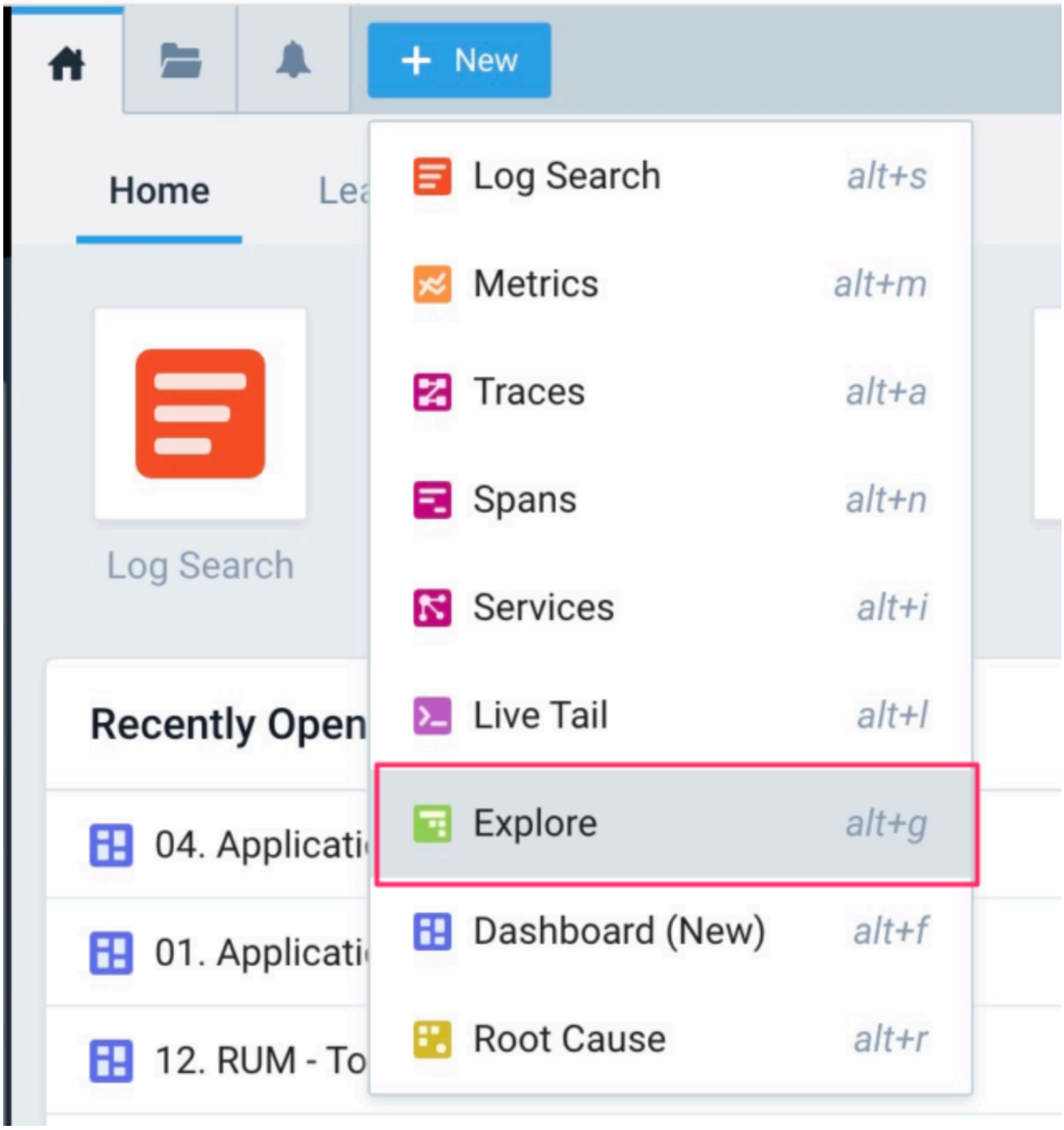
The "Traces" search screen displays a list of internal transactions that were occurring during the operation of the web application in the browser.
Clicking on one of the traces will display its details.



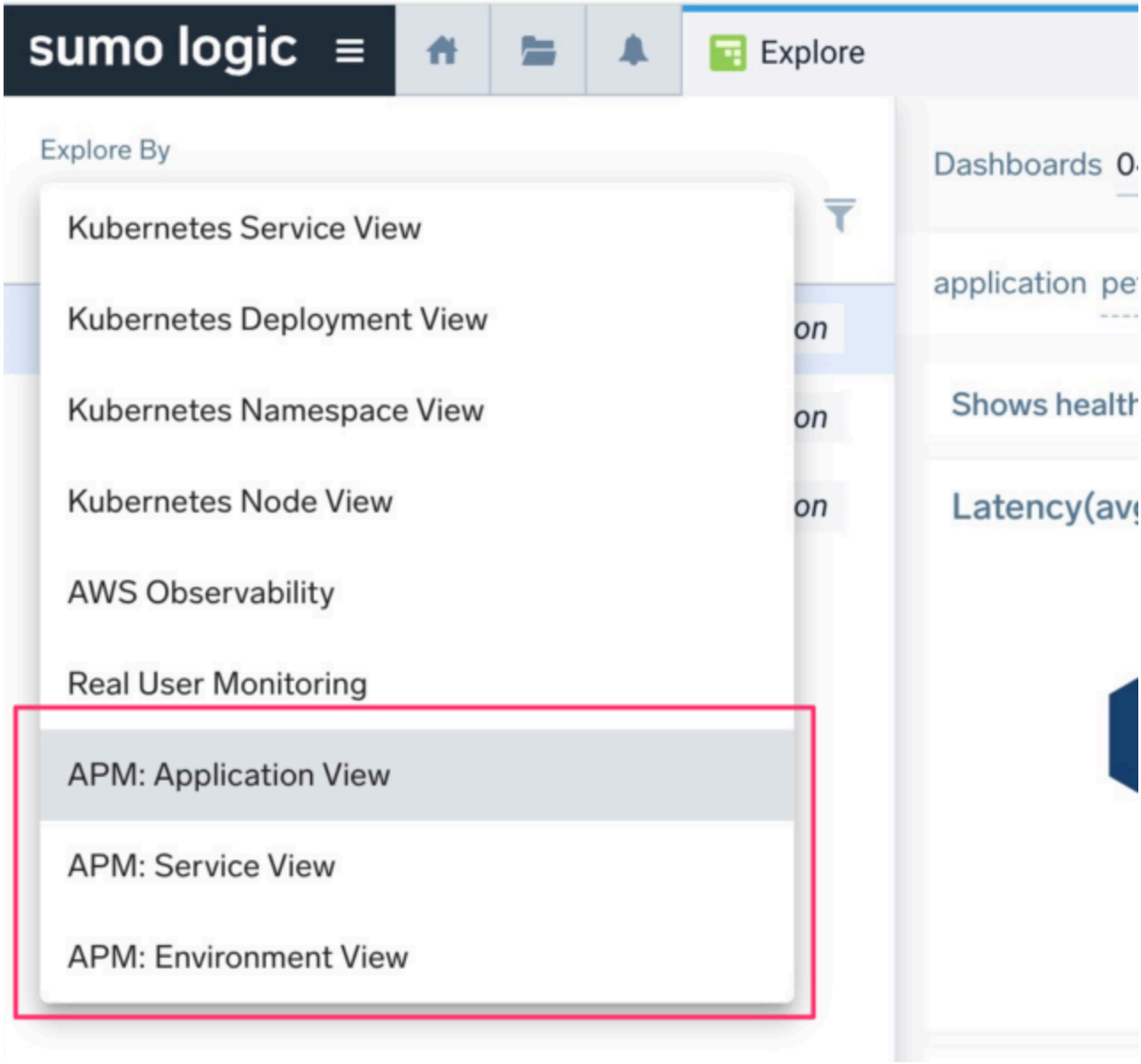
This transaction shows that the entire process took 46.75ms, and that the front-end app and database are connected. You can see how long each process took and how the process transitioned. It's
great to be able to retrieve data on application processing by operation and see the context and actual time taken to process. If a bottleneck has occurred, this alone could be useful for correcting the code and identifying the cause.

APM dashboard

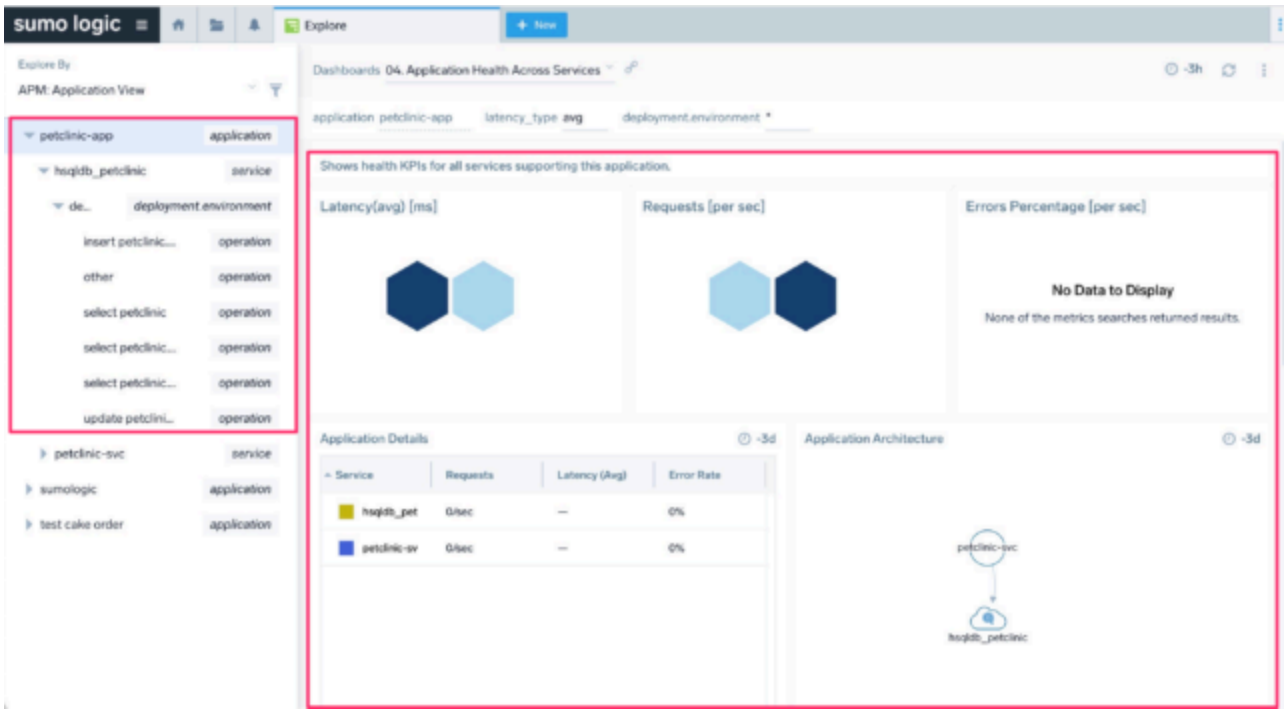
Once trace data is imported into Sumo Logic, the APM dashboard is automatically available.
You can view the APM dashboard from "Explorer" under "+New."



Under "Explore by," you'll see dashboards related to APM.
Sumo Logic can collect trace data using OpenTelemetry. You can switch dashboards by application name, service name, and environment name, which you passed as configuration values when implementing OpenTelemetry in your application.



APM:Application View allows you to understand latency and request status from the application's perspective.
In addition, the left pane displays dashboards hierarchically focused on services within the application, dashboards further focused on the environment, and dashboards focused on operations, allowing you to focus your perspective and examine the causes.



By being able to drill down statistically from a broad perspective, you can quickly identify the problem and hopefully reduce MTTR.

summary

We've tried out trace queries and dashboards to acquire traces with Sumo Logic and analyze them as APM.
While Sumo Logic is a solution that excels in log analysis as a SIEM, it also appears capable of capturing metrics and traces on a single platform, making it a useful APM tool.
Even in the APM realm, the final root cause analysis relies on checking logs, so Sumo Logic's flexibility in log analysis as a SIEM will likely be useful.