

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.

The screenshot shows the 'Collection' tab of the Sumo Logic interface. At the top, there are tabs for 'Collection', 'OpenTelemetry Collection', 'Status', 'Ingest Budgets', 'Archive', and 'Data Archiving'. Below the tabs is a search bar with placeholder text 'Search for collectors and sources by name or sourceCategory'. To the right of the search bar are links for 'Setup Wizard', 'Upgrade Collectors', 'Add Collector', 'Access Keys', and 'Tokens'. Underneath the search bar are dropdown menus for 'Show' (set to 'All Collectors'), 'Show up to: 10 collectors', and 'Expand: All | None'. On the right side of the screen, there are navigation buttons for 'Page: 2 of 2' and a chart showing 17 collectors and 1,157 sources. A red box highlights the 'Add Source' button in the bottom right corner. The main list area shows a single entry: 'test_
Healthy Hosted'.

Here, select HTTP Traces.

[Collectors and Sources](#) > [Select Source for Collector test_](#)

http x

Cloud APIs



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

Source Type

Name*
Maximum name length is 128 characters.

Description

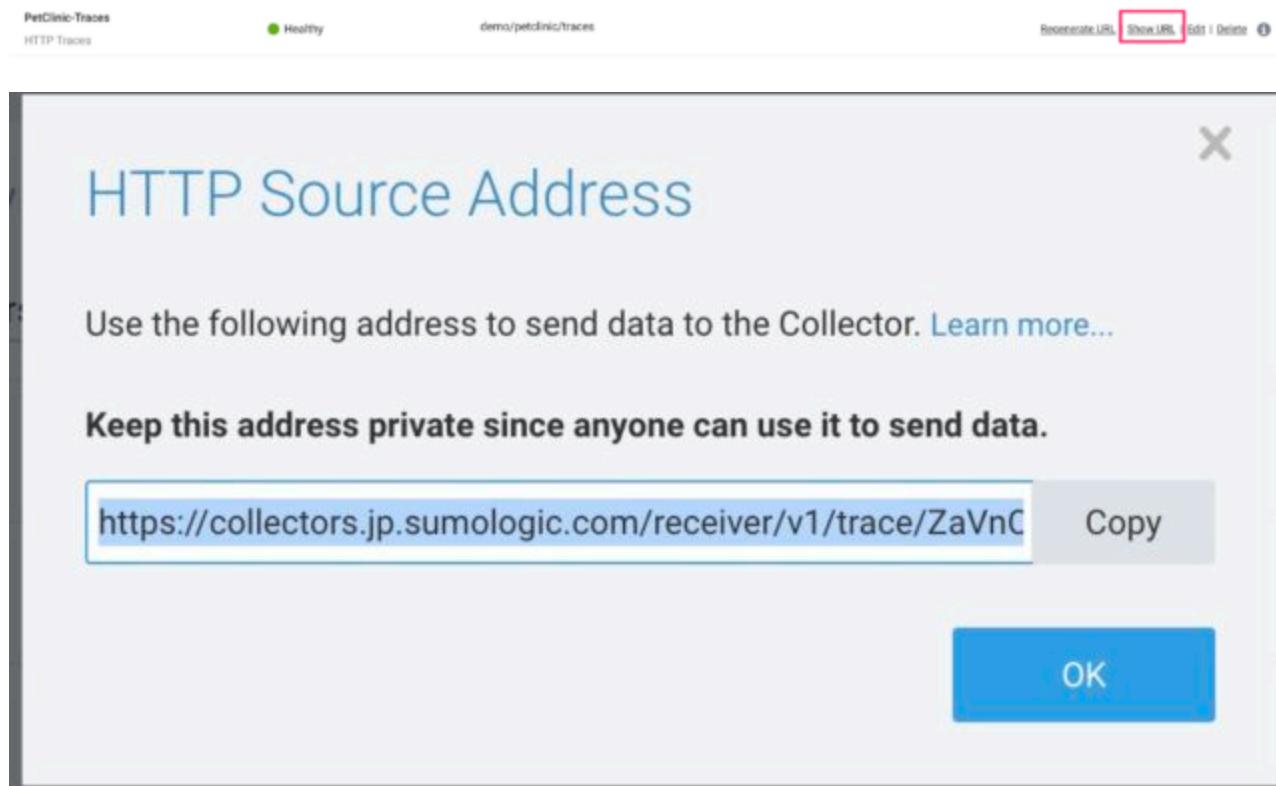
Source Host
Host name for the system from which the data is being collected. This is optional, as not all data sources have host names. This will override the default set in the "Host Name" field at the Collector level. This data is queried using the '_sourceHost' key name.

Source Category
Category metadata to use later for querying, e.g. prod/web/apache/access . This data is queried using the '_sourceCategory' key name.

Cancel
Save

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>
```

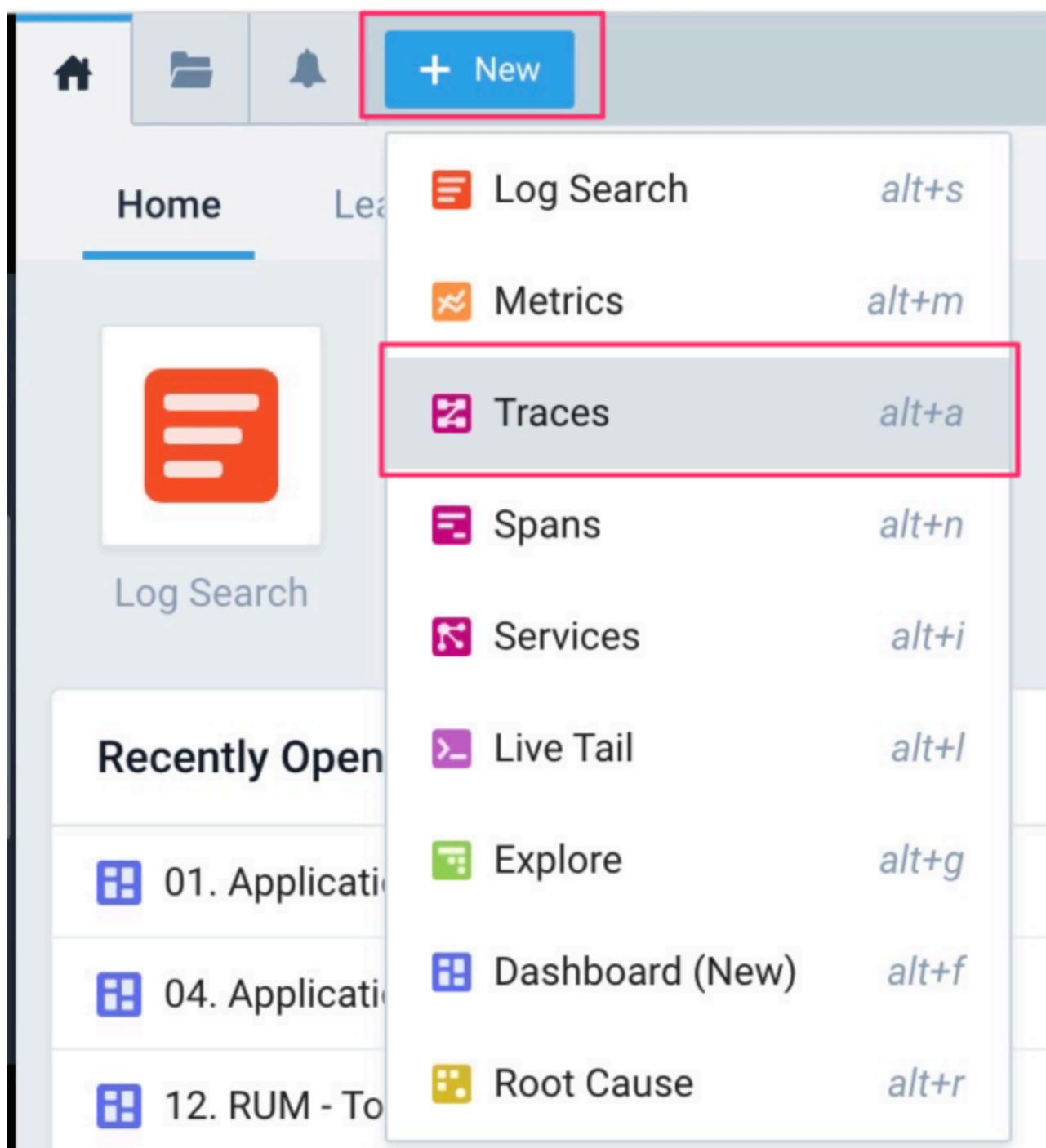
```
docker run --rm --name ot-petclinic -p 8080:8080 \
--env JAVA_TOOL_OPTIONS="-javaagent:/agent/opentelemetry-javaagent.jar" \
--env OTEL_SERVICE_NAME=petclinic-svc \
--env OTEL_RESOURCE_ATTRIBUTES=application=petclinic-app \
--env OTEL_METRICS_EXPORTER=otlp \
--env OTEL_TRACES_EXPORTER=otlp \
--env 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.

The screenshot shows the Sumo Logic Traces search interface. At the top, there's a header with the Sumo Logic logo, a search bar, and a 'Traces 3' dropdown. Below the header is a search bar with filters and a search button. The main area is titled 'Trace Query Visualizations' and shows a table titled 'Traces matching queries'. The table has columns for Trace ID, Root Service, Root Operation Name, Started At, Duration, Number of s..., Duration Breakdown, Number of er..., and Status. There are 31 rows of data, each representing a trace. The last row, which corresponds to the transaction highlighted in the detailed view below, is selected and highlighted with a red border. The status column for this row shows '302'. At the bottom of the table, it says 'Displayed traces from range [1, 31] out of 31 found.'

#A	Trace ID	Root Service	Root Operation Name	Started At	Duration	Number of s...	Duration Breakdown	Number of er...	Status
#A	b7fc308bead2de6da0e9...	petclinic-svc	/users	06/18/2023 9:32:34.663 ..	17.49ms	5	[redacted]	1	200
#A	bcd1ae967d8a06fda1bf...	petclinic-svc	/users	06/18/2023 9:32:34.168 ..	7.82ms	5	[redacted]	1	200
#A	390a97ee092c3c8499f5...	petclinic-svc	/users	06/18/2023 9:32:34.020 ..	9.25ms	5	[redacted]	1	200
#A	6aae53174eaed669495301067...	petclinic-svc	/users	06/18/2023 9:32:33.859 ..	18.29ms	5	[redacted]	1	200
#A	ee39030c669495301067...	petclinic-svc	/users	06/18/2023 9:32:33.627 ..	17.6ms	5	[redacted]	1	200
#A	256bb669f987c3550e587...	petclinic-svc	/users	06/18/2023 9:32:31.752 ..	23.52ms	5	[redacted]	1	200
#A	4166784222c3febc2f0...	petclinic-svc	/vets.json	06/18/2023 9:32:30.009 ..	25.28ms	3	[redacted]	0	200
#A	3eb1dc6f7ad0ff97baa4...	petclinic-svc	/vets.xml	06/18/2023 9:32:27.872 ..	29.22ms	3	[redacted]	0	200
#A	a5b7fb3fb42fca7a0d2...	petclinic-svc	/vets	06/18/2023 9:32:25.691 ..	19.65ms	8	[redacted]	0	200
#A	0:61699886ec08f280680...	petclinic-svc	/owners/{ownerId}	06/18/2023 9:32:23.672 ..	27.17ms	18	[redacted]	0	200
#A	eb16fe2339ec2e4d3efc7...	petclinic-svc	/owners/{ownerId}/pets/...	06/18/2023 9:32:23.617 ..	46.75ms	16	[redacted]	0	302
#A	042dbc059ba28903370...	petclinic-svc	/owners/{ownerId}/beta/...	06/18/2023 9:32:17.852 ..	44.5ms	11	[redacted]	0	200
#A	65a1fb6216989268bed...	petclinic-svc	/owners/{ownerId}	06/18/2023 9:32:16.187 ..	24.52ms	8	[redacted]	0	200
#A	7d13eb970a03273c05f...	petclinic-svc	/owners/new	06/18/2023 9:32:16.134 ..	48.4ms	7	[redacted]	0	302

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

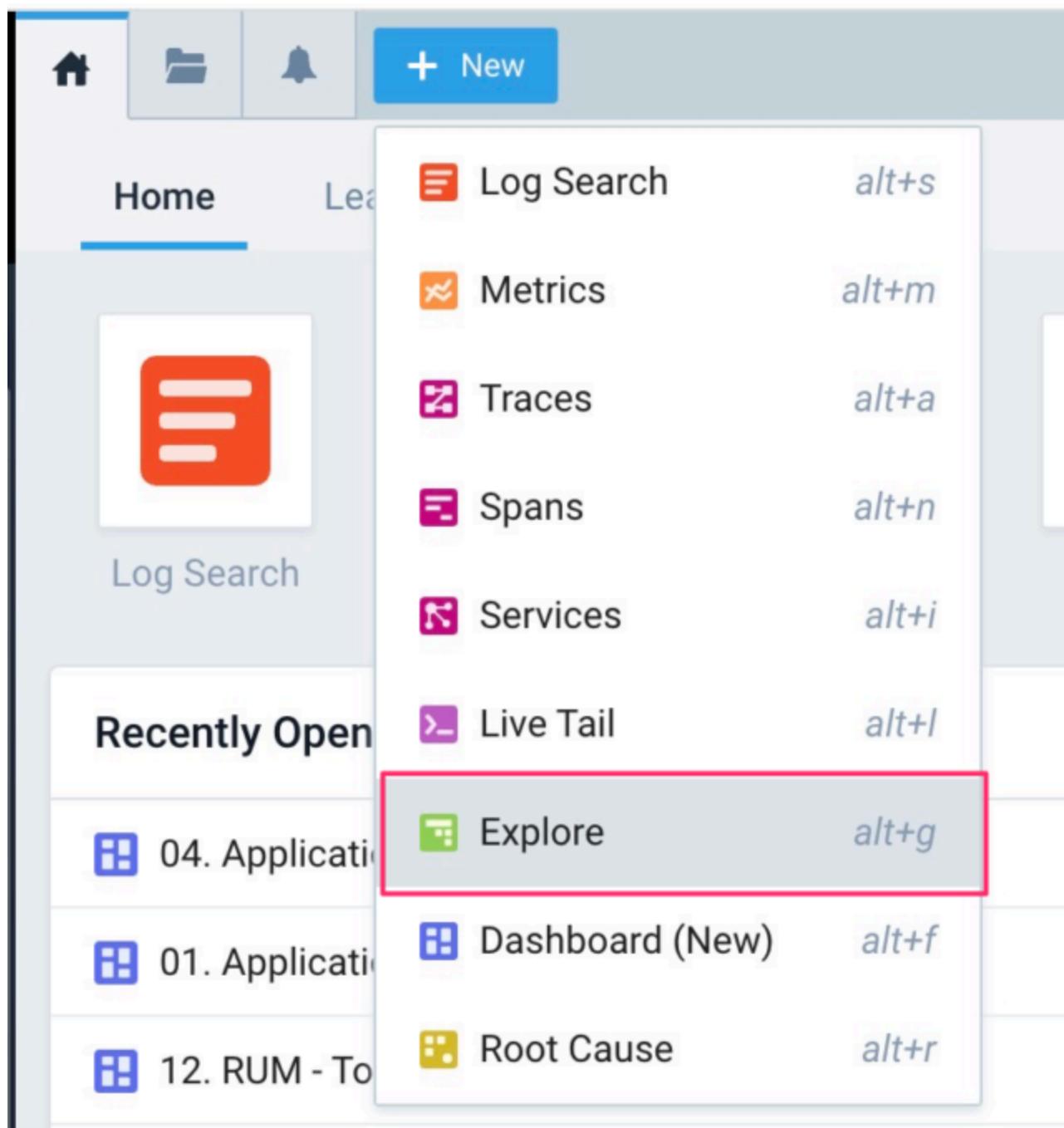
great to be able to retrieve data on application processing by operation and see the context and actual time taken to process. If a bottleneck has occurred, this alone could be useful for correcting the code and identifying the cause.

The screenshot shows a detailed view of a trace. At the top, it says 'petclinic-svc, /owners/{ownerId}/pets/new Trace (ID #eb16fe2339ec2e4d3efc7555819d90b5) 302'. Below that, it shows the start time '06/18/2023 9:32:23 AM' and the total span '46.75ms'. There are '16 Total Spans'. A 'Filters' section is shown. The main part of the screen is a timeline visualization with colored bars representing different operations. The first bar is labeled 'Critical petclinic-backend by service' and 'petclinic-svc - 97%'. The timeline shows several operations: 'Dns' (46.75ms), 'SELECT * FROM owners WHERE id = ?' (46.75ms), 'PreController processCreatePetForUser' (41.91ms), 'SELECT PetType' (28.79ms), 'SELECT P...' (41.91ms), and 'DeletePet(pet)' (41.91ms). The 'DeletePet(pet)' operation is highlighted with a red border.

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."



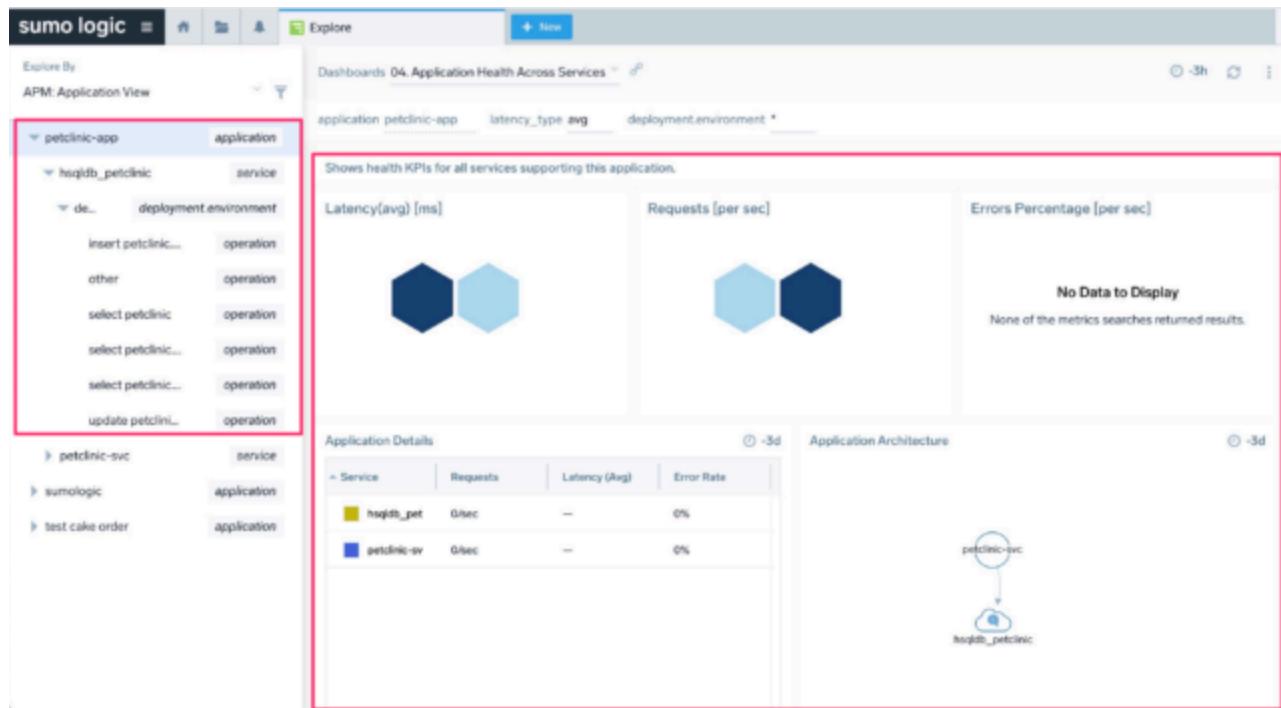
Under "Explore by," you'll see dashboards related to APM.

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.

The screenshot shows the Sumo Logic interface. At the top, there's a dark header bar with the "sumo logic" logo, a menu icon, and three small icons (home, folder, bell). To the right of the header is a "Explore" button with a green square icon. Below the header is a sidebar titled "Explore By" with a filter icon. The sidebar lists several views: "Kubernetes Service View", "Kubernetes Deployment View", "Kubernetes Namespace View", "Kubernetes Node View", "AWS Observability", and "Real User Monitoring". Under "Real User Monitoring", there's a section with a red border containing three items: "APM: Application View", "APM: Service View", and "APM: Environment View". To the right of the sidebar, there's a main content area with a "Dashboards 0" section and some partially visible text: "application pe", "Shows health", and "Latency(av)".

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.