OpenSearch

Documentation

Trace analytics / Analyze Jaeger trace data

Analyze Jaeger trace data

NOTIFICATIONS PLUGIN

JOB SCHEDULER PLUGIN

CLIENTS

 \sim

 \sim

 \sim

INTRODUCED 2.5 The Trace analytics functionality in the OpenSearch Observability plugin now supports Jaeger trace data. If you use OpenSearch as the backend for Jaeger trace data, you can use the Trace analytics built-in analysis capabilities. This provides support for OpenTelemetry (OTEL) formatted trace data. When you perform trace analytics, you can select from two data sources: • **Data Prepper** – Data ingested into OpenSearch through Data Prepper. • Jaeger – Trace data stored within OpenSearch as its backend. If you currently store your Jaeger trace data in OpenSearch, you can now use the capabilities built into Trace Analytics to analyze the error rates and latencies You can also filter the traces and look into the span details of a trace to pinpoint any service issues. When you ingest Jaeger data into OpenSearch, it gets stored in a different index than the OTA-generated index that gets created when you run data through the Data Prepper. You can indicate which data source on which you want to perform trace analytics with the data source selector in the Dashboards. Jaeger trace data that you can analyze includes span data, as well as service and operation endpoint data. Jaeger span data analysis requires some configuration. By default, each time you ingest data for Jaeger, it creates a separate index for that day. To learn more about Jaeger data tracing, see the Jaeger open source documentation. **Data Ingestion Requirements** To use Trace Analytics with Jaeger data, you need to configure error capability for use with trace analytics. Jaeger data that is ingested for OpenSearch needs to have the environment variable ES_TAGS_AS_FIELDS_ALL set to true for errors. If data is not ingested in this format it will not work for errors and error data will not be available for traces in trace analytics with OpenSearch. About Data ingestion with Jaeger indexes Trace analytics for non-Jaeger data use OTEL indexes with the naming conventions otel-v1-apm-span-* Or otel-v1-apm-service-map*. Jaeger indexes follow the naming conventions jaeger-span-* Or jaeger-service-*. How to set up OpenSearch to use Jaeger data We provide a sample Docker compose file that contains the required configurations. Step 1: Run the Docker compose file Use the following Docker compose file to enable Jaeger data for trace analytics with the ES_TAGS_AS_FIELDS_ALL environment variable set to true to enable errors to be added to trace data. Copy the following Docker compose file contents and save it as docker-compose.yml. version: '3' services: opensearch-node1: # This is also the hostname of the container within the Docker network (i.e. https://opensearch-node1/) image: opensearchproject/opensearch:latest # Specifying the latest available image - modify if you want a specific version container_name: opensearch-node1 environment: - cluster.name=opensearch-cluster # Name the cluster - node.name=opensearch-node1 # Name the node that will run in this container - discovery.seed_hosts=opensearch-node1,opensearch-node2 # Nodes to look for when discovering the cluster - cluster.initial_cluster_manager_nodes=opensearch-node1,opensearch-node2 # Nodes eligible to serve as cluster manager - bootstrap.memory_lock=true # Disable JVM heap memory swapping - "OPENSEARCH_JAVA_OPTS=-Xms512m -Xmx512m" # Set min and max JVM heap sizes to at least 50% of system RAM ulimits: memlock: soft: -1 # Set memlock to unlimited (no soft or hard limit) hard: -1 nofile: soft: 65536 # Maximum number of open files for the opensearch user - set to at least 65536 hard: 65536 volumes: - opensearch-data1:/usr/share/opensearch/data # Creates volume called opensearch-data1 and mounts it to the container ports: - "9200:9200" - "9600:9600" networks: - opensearch-net # All of the containers will join the same Docker bridge network opensearch-node2: image: opensearchproject/opensearch:latest # This should be the same image used for opensearch-node1 to avoid issues container_name: opensearch-node2 environment: - cluster.name=opensearch-cluster

Download About Community Documentation

ON THIS PAGE

Data Ingestion

Requirements

How to set up

in OpenSearch

Dashboards

OpenSearch

Dashboards

Traces

Services

Dashboard view

Jaeger data

OpenSearch to use

Step 2: View trace data

Use trace analytics in

WAS THIS PAGE HELPFUL?

See a problem? Submit issues or edit this page on GitHub.

≭No

✓ Yes

Q Search...

- node.name=opensearch-node2

- discovery.seed_hosts=opensearch-node1,opensearch-node2

- cluster.initial_cluster_manager_nodes=opensearch-node1,opensearch-node2
- bootstrap.memory_lock=true

- "OPENSEARCH_JAVA_OPTS=-Xms512m -Xmx512m"

ulimits: memlock:

soft: -1

hard: -1 nofile: soft: 65536 hard: 65536 volumes: - opensearch-data2:/usr/share/opensearch/data networks: - opensearch-net opensearch-dashboards: image: opensearchproject/opensearch-dashboards:latest # Make sure the version of opensearch-dashboards matches the version opensearch-dashboards container name: opensearch-dashboards ports: - 5601:5601 # Map host port 5601 to container port 5601 expose: - "5601" # Expose port 5601 for web access to OpenSearch Dashboards environment: OPENSEARCH_HOSTS: '["https://opensearch-node1:9200","https://opensearch-node2:9200"]' # Define the OpenSearch nodes that OpenS networks: - opensearch-net jaeger-collector: image: jaegertracing/jaeger-collector:latest ports: - "14269:14269" - "14268:14268" - "14267:14267" - "14250:14250" - "9411:9411" networks: - opensearch-net restart: on-failure environment: - SPAN_STORAGE_TYPE=opensearch - ES_TAGS_AS_FIELDS_ALL=true - ES USERNAME=admin - ES_PASSWORD=admin - ES_TLS_SKIP_HOST_VERIFY=true command: ["--es.server-urls=https://opensearch-node1:9200", "--es.tls.enabled=true", 1 depends_on: - opensearch-node1 jaeger-agent: image: jaegertracing/jaeger-agent:latest hostname: jaeger-agent command: ["--reporter.grpc.host-port=jaeger-collector:14250"] ports: - "5775:5775/udp" - "6831:6831/udp" - "6832:6832/udp" - "5778:5778" networks: - opensearch-net restart: on-failure environment: - SPAN_STORAGE_TYPE=opensearch depends_on: - jaeger-collector hotrod: image: jaegertracing/example-hotrod:latest ports: - "8080:8080" command: ["all"] environment: - JAEGER_AGENT_HOST=jaeger-agent - JAEGER_AGENT_PORT=6831 networks: - opensearch-net depends_on: - jaeger-agent volumes: opensearch-data1: opensearch-data2: networks: opensearch-net:

Step 2: Start the cluster

Run the following command to deploy the Docker compose YAML file.

docker compose up -d

To stop the cluster, run the following command:

docker compose down

Step 2: View trace data in OpenSearch Dashboards

After you generate Jaeger trace data you can go to OpenSearch Dashboards to view your trace data.

Go to OpenSearch Dashboards Trace Analytics at Trace Analytics.

Use trace analytics in OpenSearch Dashboards

To analyze your Jaeger trace data in the Dashboards, you need to set up Trace Analytics first. To get started, see Get started with Trace Analytics.

Data sources

You can specify either Data Prepper or Jaeger as your data source when you perform trace analytics. From the OpenSearch Dashboards, go to **Observability > Trace Analytics** and select Jaeger.

OpenSearch Da	shboards	
	rervability Trace analytics	Dashboard
Observability Application analytics Trace analytics	Dashboa _{Jaeger} ~	ard
Traces Services Event analytics	CHOOSE DATA TYPE	oup name, service nam
Metrics analytics Operational panels Notebooks	 Q Filter options ✓ Jaeger ← Data Prepper 	'er time

Dashboard view

After you select Jaeger for the data source, you can view all of your indexed data in **Dashboard** view including **Error rate** and **Throughput**.

Error rate

You can view the Trace error count over time in the Dashboard, and also view the top five combinations of services and operations that have a non-zero error rate.

ashboard					
Trace ID, trace group name, service n	ame	🛗 🛩 Last 5 years		Show dates	C Refre
+ Add filter					
race error rate over time				Error rate Th	nroughput
		Now: 51.15%			
60%					
40%					
0%	2021	2021	2021	2021	
op 5 Service and Operation Errors	(5)				
Service and Operation Name $^{\odot}$		Average latency (ms) $^{\odot}$		Error rate ⁽¹⁾	Traces
frontend0,HTTP GET /config0		0.04		100%	89,77
frontend0,HTTP GET /config2		0.04		100%	88,44
frontend0,HTTP GET /config4		0.04		100%	89,58

Throughput

With **Throughput** selected, you can see the throughput of traces on Jaeger indexes that are coming in over time.

You can select an individual Trace from **Top 5 Service and Operation Latency** list and view the detailed trace data.

Dashboard Jaeger ~			
Q Trace ID, trace group name, service name	iii ← Last 5 years		Show dates C Refresh
+ Add filter			
Traces over time			Error rate Throughput
	Now: 1,306,425		
(U) 1M			
0 20212021	2021	2021	2021
Top 5 Service and Operation Latency (5)		< 95 percentile	>= 95 percentile
Service and Operation Name Latency variance (ms) 0 0.2 0.4 0.6	0.8 1 ^V	Average latency (ms) $^{\odot}$	Error rate [©] Traces [©]
frontend0,HTTP GET /config0		0.04	100% 89,774
frontend0,HTTP GET /config1		0.04	0% 91,358

You can also see the combinations of services and operations that have the highest latency.

If you select one of the entries for Service and Operation Name and go to the **Traces** column to select a trace, it will add the service and operation as filters for you.

Traces

In **Traces**, you can see the latency and errors for the filtered service and operation for each individual Trace ID in the list.

Traces			
Q Trace ID, trace group nam	e, service name	iii → Last 5 years	Show dates C Refre
process.serviceName: fronten	d0 × operationName: HTTP GET /config0 ×	× + Add filter	
Traces (3000) Trace ID 个		Latency (ms)	Errors Last updated
00de6a9aaf045bd400	(i)	0.04	Yes 12/16/2022 10:08:12
			10.00.12
00de6a9aaf045bd4010	1	0.04	Yes 12/16/2022 10:08:12

If you select an individual Trace ID, you can see more detailed information about the trace, such as time spent by the service and each span for the service and operation. You can also view the payload that you get from the index in JSON format.

Observability Trace analytics Traces Odd	e6a9aaf045bd400			
0de6a9aaf045bd400				
Dverview				
Trace ID 00de6a9aaf045bd400 👔	Latency 0.04 Last updated 12/16/2022 10:08:12		Errors Yes	
Time spent by service	Spans (3)			Timeline Span list
• frontend0 100%	0 ms frontend0 HTTP GET /config0 frontend0 HTTP GET /config0 frontend0 HTTP GET /config0	0.01 ms 0.02 ms	0.03 ms 0.04 ms	
Payload { { {				
<pre>'_id': '_gaRlIUByJaFH6S_Qt1g', '_score': 12.914443, '_source': { 'traceID': '00de6a9aaf045bd400', 'spanID': '00de6a9aaf045bd400', 'flage': 1, 'operationName': 'WTTP GET /config0', 'references': [], 'startTimeM': 1671214092597974, 'startTimeMillis': 1671214092597, 'duration': 40, 'tags': [], 'tag': { 'component': 'net/http', 'http0method': 'GET', 'http0method': 'GET', 'http0</pre>				

Services

You can also look at individual error rates and latency for each individual service. Go to **Observability > Trace Analytics > Services**. In Services, you can see the average latency, error rate, throughput and trace for each service in the list.

Jaeger 🗸			
Q Trace ID, trace group name, service	🛗 🗸 Last 5 years	Show dates C	Refrest
+ Add filter			
Services (7)			
Name 🛧	Average latency (ms)	Error rate Throughput	Traces
frontend0	0.04	50% 746,657	685,7 32
frontend1	0.04	50% 52,000	39,96 9
frontend10	0.04	50% 52,000	39,73 8
frontend11	0.04	50% 52,000	39,55 6
frontend12	0.04	50% 52,000	40,00 4
frontend13	0.04	50% 52,000	40,38 1

Get Involved	Resources	Connect
Code of Conduct	FAQ	Connect
Forums	Testimonials	
Github	Brand Guidelines	
Community Projects	Trademark Usage Policy	
	OpenSearch Disambiguation	
	Trademark Usage Policy	

© 2023 OpenSearch contributors. OpenSearch is a registered trademark of Amazon Web Services.

© 2005-2021 Django Software Foundation and individual contributors. Django is a registered trademark of the Django Software Foundation. This website was forked from the BSD-licensed <u>djangoproject.com</u> originally designed by <u>Threespot</u> & <u>andrevv</u>. We \heartsuit Django and the Django community. If you need a <u>high-level Python framework</u>, check it out.