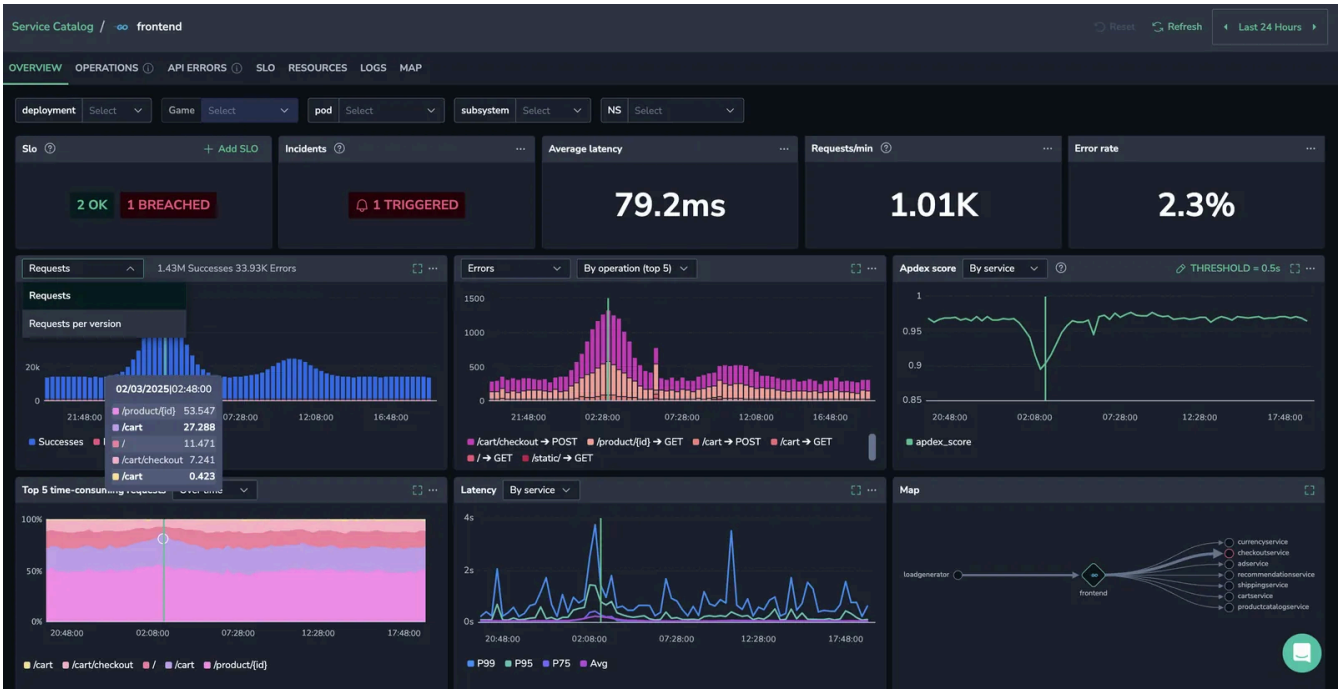
APM Onboarding

This guide provides step-by-step instructions on configuring and using Coralogix Application Performance Monitoring (APM). Follow these steps to effectively onboard your services, install collectors, correlate resources, and leverage advanced features.

Overview

To begin monitoring your applications efficiently, follow these steps:

1. Deploy the OpenTelemetry Collector:
    - Install and configure the OpenTelemetry Collector to automatically collect spans and generate Span Metrics for comprehensive monitoring and performance tracking.
    - If needed, enable Span Metrics. By default, Span Metrics are enabled for efficient and scalable APM monitoring.
2. Generate metrics and spans. Use OpenTelemetry instrumentation or the eBPF auto-instrumentation to send traces and metrics from your services.
3. Optimize with head sampling for improved performance and data efficiency (by default, it is set at 10%).
4. Correlate logs and infrastructure resource data with your services.
5. If needed, configure serverless instrumentation. Learn how to send your serverless functions data to Coralogix.



Deploy the OpenTelemetry Collector

Deployment methods

Select from any of the following configuration options:

- Docker
- EC2
- ECS-Fargate
- ECS-EC2
- Kubernetes Complete Observability:
  a. Download the `values.yaml` file (for users starting from scratch).
  b. Make sure all K8s and Helm prerequisites (minimum Kubernetes and Helm versions, secrets setup) are met.
  c. Make sure that the `spanMetrics` preset is enabled.

```
spanMetrics:
    enabled: true
```

  d. Use Helm installation commands detailed in the installation guide.

Enable Span Metrics

By default, Span Metrics are enabled in our latest Helm chart. However, you can add Span Metrics to your configuration file or enable it in your values.yaml. Refer to Coralogix Span Metrics for details.

When Span Metrics are enabled, the following metrics are generated:

Metric name (OTLP)	Prometheus exported name	Metric type	Auto-generated labels (dimensions)
`requests`	`calls_total`	Counter	`service.name`, `span.name`, `span.kind`, `status.code`
`duration`	`duration_ms_bucket`, `duration_ms_sum`, `duration_ms_count`	Histogram	`service.name`, `span.name`, `span.kind`, `status.code`, `le` (for bucket)
`errors`	`errors_total`	Counter	`service.name`, `span.name`, `span.kind`, `status.code`

Additional dimensions introduced by Coralogix

When using the Coralogix `values.yaml`, the following dimensions are added to enable APM functionalities and enhance data analysis:

- Dimensions added by default as part of Coralogix `values.yaml`.

```
extraDimensions:
        - name: http.method
        - name: cgx.transaction
        - name: cgx.transaction.root
        - name: status_code
        - name: db.namespace
        - name: db.operation.name
        - name: db.collection.name
        - name: db.system
```

- Once you enable the Span Metrics preset, the `errorTracking` configuration will be enabled automatically. Learn more about API error tracking.
- Once you enable the Span Metrics preset, the `serviceVersion` configuration will be enabled automatically. Learn more about Grouping service metrics by version.
  - Make sure `service.version` attribute is added to your service spans.
- Once you enable the Span Metrics preset, the `dbMetrics` configuration will be enabled automatically. It generates RED (Request, Errors, Duration) metrics for database spans. For example, `query db_calls_total` is used to view generated request metrics. This is needed to enable the Database Monitoring feature inside Coralogix APM.

Transform statements

OpenTelemetry sometimes introduces breaking changes that, if left unaddressed, can disrupt the experience for customers upgrading to newer versions. Coralogix APM and the OpenTelemetry Collector (deployed via the Coralogix Helm chart) support a `transform` processor to rename attributes before exporting them. Refer to Aligning Coralogix and OTel Naming Conventions for details.

Optimize high cardinality in metrics and spans

High cardinality (over 300K), which occurs when metrics or spans contain labels with numerous unique values, such as user IDs, UUIDs, or session-specific data. This creates a large number of metric combinations, often exceeding practical limits. For example, using user-specific values in span names or labels can lead to exponentially growing cardinality, complicating metric analysis and visualization. In cases of high cardinality caused by overly unique span names, we recommend adjusting your instrumentation or using the `spanMetrics.spanNameReplacePattern` parameter to replace the problematic values with a generic placeholder. For example, if your span name corresponds to template `user-1234`, you can use the following pattern to replace the user ID with a

generic placeholder. This will result in your spans having a generalized name `user-{id}`. Learn how to replace specific `span.name` with a generic one as detailed here.

Use `metrics_expiration` when you want to control how long unexported metrics are kept in memory.

Generate metrics and spans

OpenTelemetry instrumentation

Instrument your services to generate spans. Choose your programming language:

- Manual instrumentation.
- Auto instrumentation.

Quick enablement for Kubernetes using eBPF

Auto-instrument your applications using eBPF with OBI.

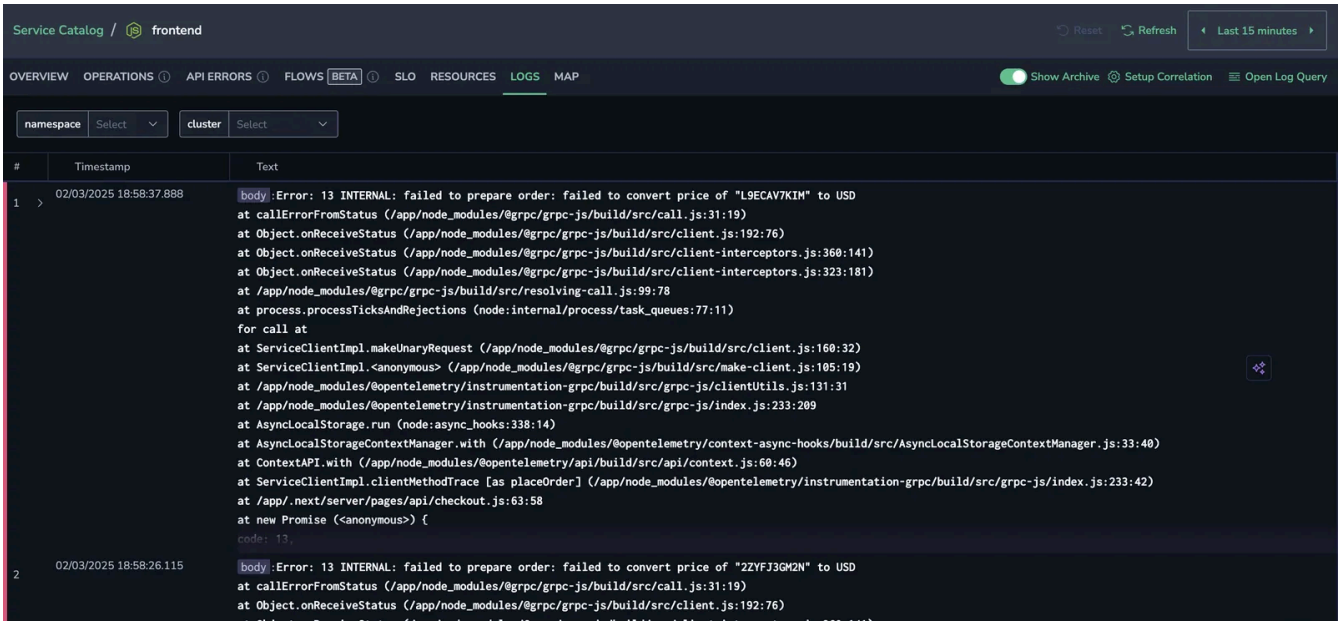
Optimize with head or tail sampling (recommended)

By default, our latest Helm chart includes head sampling of traces. Use one of these methods for upgrading to tail sampling:

- **Kubernetes**: Tail sampling with OpenTelemetry using Coralogix Helm chart.
- **Docker Compose**: Tail sampling with OpenTelemetry using Docker Compose.

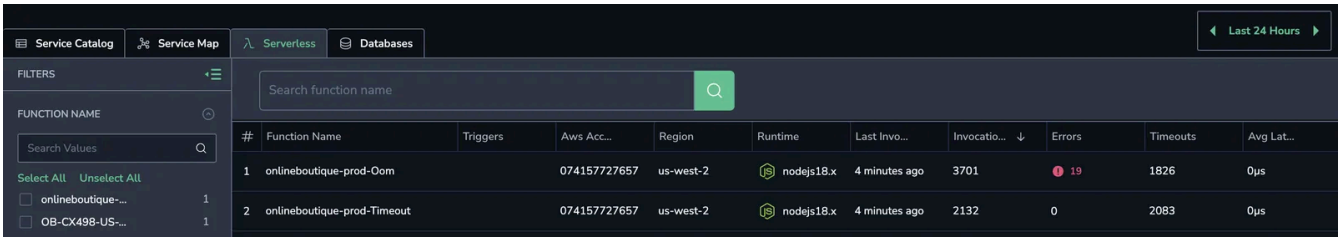
Log correlation

Investigate your service logs by enriching them with a subsystem name that matches your service name. Alternatively, follow these instructions to set up your log correlation.



Instrument serverless functions

Auto-instrument AWS Lambda by following these instructions.



GitHub documentation	Coralogix OTel integration for K8s
Tutorial	OpenTelemetry and Coralogix introduction
Tutorial	OpenTelemetry Lambda function integration
Tutorial	ECS to Coralogix using OpenTelemetry

Last updated: July 30, 2025

Was this helpful?

Leave your feedback here.

◯ **Yes** ◯ **No**

Send

Generated on: November 18, 2025
Source: https://coralogix.com/docs/user-guides/apm/getting-started/apm-onboarding-tutorial/