OpenTelemetry ECS Fargate

> **NOTE**
>
> Previous versions of this integration used an ADOT (AWS Distribution for OpenTelemetry) collector image. If you are upgrading an existing deployment, ensure both the configuration and the task definition are updated.
>
> Previous versions required logs to be processed using the Fluentbit log router. This is no longer necessary, as OTEL can now collect logs along with metrics and traces.

Overview

This tutorial demonstrates how to add the OTEL Collector as a sidecar agent to your ECS Task Definitions. We use the standard OpenTelemetry Collector Contrib distribution, leveraging the `envprovider` to generate the configuration from an AWS SSM Parameter Store. You can review an example CloudFormation template here.

Using `envprovider` for configuration

The `envprovider` is utilized to load the OpenTelemetry configuration via AWS Systems Manager Parameter Store. This allows for more dynamic and convenient configuration adjustments compared to embedding a static configuration within the container image.

The config.yaml file contains a standard configuration for ingesting logs, metrics, and traces to our backend. Ensure that the Parameter Store is created in the same region as your ECS cluster. To simplify the process, a sample CloudFormation template has been included to deploy the Parameter Store.

Adding the OTEL Collector to ECS task definitions

Once the Parameter Store is created, you need to add the OTEL container to your existing Task Definitions.

Example container declaration

Below is an example of how to declare the OTEL collector container within your Task Definition:

```
"containerDefinitions": [
```

```
    {
        <Existing Container Definitions>
    },
    {
        "name": "otel-collector",
        "image": "otel/opentelemetry-collector-contrib",
        "cpu": 0,
        "portMappings": [
            {
                "name": "otel-collector-4317-grpc",
                "containerPort": 4317,
                "hostPort": 4317,
                "protocol": "tcp",
                "appProtocol": "grpc"
            },
            {
                "name": "otel-collector-4318-http",
                "containerPort": 4318,
                "hostPort": 4318,
                "protocol": "tcp"
            }
        ],
        "essential": false,
        "command": [
            "--config",
            "env:SSM_CONFIG"
        ],
        "environment": [
            {
                "name": "PRIVATE_KEY",
                "value": "<Coralogix PrivateKey>"
            },
            {
                "name": "CORALOGIX_DOMAIN",
                "value": "<Coralogix Domain>"
            }
        ],
        "mountPoints": [],
        "volumesFrom": [],
        "secrets": [
            {
                "name": "SSM_CONFIG",
```

```
                "valueFrom": "CX_OTEL_ECS_Fargate_config.yaml"
            }
        ],
        "user": "0",
        "logConfiguration": {
            "logDriver": "awsfirelens",
            "options": {
                "Name": "OpenTelemetry"
            }
        },
        "systemControls": [],
        "firelensConfiguration": {
            "type": "fluentbit"
        }
    }
]
```

Configuring log forwarding

In the above example, replace `<Coralogix PrivateKey>` and `<Coralogix Domain>` with the actual values for your setup. The `logConfiguration` in this example forwards OTEL logs to the Coralogix platform. Be sure to apply the same `logConfiguration` to all existing containers:

```
"logConfiguration": {
    "logDriver": "awsfirelens",
    "options": {
        "Name": "OpenTelemetry"
    }
}
```

Alternative log drivers (CloudWatch)

If you don't want specific container logs to be sent to Coralogix, you can configure the log driver for that container to CloudWatch or any other preferred destination. Here's an example for CloudWatch:

```
"logConfiguration": {
    "logDriver": "awslogs",
    "options": {
        "awslogs-create-group": "true",
```

```
            "awslogs-group": "<Log Group Name>",
            "awslogs-region": "<Your Region>",
            "awslogs-stream-prefix": "<Stream Prefix>"
        }
    }
```

Configuration Details

Resource Catalog

The Coralogix Resource Catalog can be used to visualize and track your ECS containers. It collects container details and lets you observe performance metrics and review logs of the associated containers.

Resource Catalog data collection is enabled by default. It is composed of the following components:

- transform/entity-event processor
- coralogix/resource_catalog exporter
- logs/resource_catalog pipeline under service

To disable the Resource Catalog, comment out or delete the `logs/resource_catalog` pipeline under the `service` section in the configuration file.

Using Secrets Manager for your private key

If you prefer to store your Coralogix private key in AWS Secrets Manager, remove the `"PRIVATE_KEY"` config from the `"environment"` section and instead add it to `"secrets"`, referencing the Secret's ARN.

```
"secrets": [
    {
        "name": "SSM_CONFIG",
        "valueFrom": "CX_OTEL_ECS_Fargate_config.yaml"
    },
    {
        "name": "PRIVATE_KEY",
        "valueFrom":
"arn:aws:secretsmanager:region:aws_account_id:secret:secret_name-AbCdEf"
    }
],
```

Create the Secret as "Plaintext" with only the API key with no quotation marks. You will also need to add the `secretsmanager:GetSecretValue` permission to your ECS Task

Execution Role.

Permissions

Granting permissions for parameter store access

To allow your container to access the Systems Manager Parameter Store, you need to provide `ssm:GetParameters` action permission to the ECS Task Execution Role. Here's an example of the required permissions:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:GetParameters"
      ],
      "Resource": [
        "arn:aws:ssm:region:aws_account_id:parameter/parameter_name"
      ]
    }
  ]
}
```

Granting permissions for Secrets Manager Secret access

To allow your container to access the Secrets Manager Secret, you need to provide `secretsmanager:GetSecretValue` action permission to the ECS Task Execution Role. Here's an example of the required permissions:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Resource": [
        "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name-
AbCdEf"
```

```
        ]
      }
    ]
}
```

Submitting metrics and traces

After adding the OTEL container to your Task Definition, your applications can submit traces and metrics to the following endpoints:

- **Traces**: `http://localhost:4318/v1/traces`
- **Metrics**: `http://localhost:4318/v1/metrics`

The OTEL collector will also automatically gather container metrics from all containers within the Task Definition.

This guide outlines the steps to integrate the OTEL collector into your ECS Fargate setup, helping you efficiently collect telemetry data from your applications.

Support

Need help?

Our world-class customer success team is available 24/7 to walk you through your setup and answer any questions that may come up.

Feel free to reach out to us **via our in-app chat** or by sending us an email to support@coralogix.com.

Last updated: February 22, 2025

Was this helpful?

Leave your feedback here.

◯ **Yes** ◯ **No**

Send