To instrument OpenTelemetry for tracing in your existing .NET application running in AWS ECS without modifying your application code, you can use automatic instrumentation with the OpenTelemetry .NET Auto-Instrumentation agent.

**Steps to Enable OpenTelemetry Tracing Without Code Changes**

**1. Use the OpenTelemetry .NET Auto-Instrumentation Agent**

The OpenTelemetry .NET Auto-Instrumentation agent can automatically capture telemetry data from your .NET application without modifying the source code.

**2. Modify ECS Task Definition to Inject the OpenTelemetry Agent**

Since your app is running in AWS ECS, update the ECS Task Definition to:

* Mount the OpenTelemetry auto-instrumentation library.
* Set necessary environment variables.

**Modify the Task Definition JSON**

Add the following essential container for OpenTelemetry:

{

"name": "otel-auto-instrumentation",

"image": "ghcr.io/open-telemetry/dotnet-auto-instrumentation:latest",

"essential": true,

"volumesFrom": [

{

"sourceContainer": "your-app-container"

}

],

"environment": [

{

"name": "OTEL\_EXPORTER\_OTLP\_ENDPOINT",

"value": "http://your-otel-collector:4317"

},

{

"name": "OTEL\_SERVICE\_NAME",

"value": "my-dotnet-app"

},

{

"name": "OTEL\_DOTNET\_AUTO\_PLUGINS",

"value": "OpenTelemetry.AutoInstrumentation.Instrumentation.AspNetCore,OpenTelemetry.AutoInstrumentation.Instrumentation.HttpClient"

},

{

"name": "CORECLR\_ENABLE\_PROFILING",

"value": "1"

},

{

"name": "CORECLR\_PROFILER",

"value": "{918728DD-259F-4A6A-AC2B-B85E1B658318}"

},

{

"name": "CORECLR\_PROFILER\_PATH",

"value": "/otel-auto-instrumentation/OpenTelemetry.AutoInstrumentation.Native.so"

}

]

}

* OTEL\_EXPORTER\_OTLP\_ENDPOINT → Set this to your OpenTelemetry Collector.
* OTEL\_SERVICE\_NAME → Name of your service for identification.
* OTEL\_DOTNET\_AUTO\_PLUGINS → Automatically instruments ASP.NET Core and HttpClient.
* Profiler Environment Variables → Enables OpenTelemetry auto-instrumentation.

**3. Deploy an OpenTelemetry Collector in ECS**

To collect traces, deploy an OpenTelemetry Collector alongside your application.

**Example ECS Task for OpenTelemetry Collector**

{

"name": "otel-collector",

"image": "otel/opentelemetry-collector-contrib:latest",

"essential": true,

"command": ["--config=/etc/otel-collector-config.yaml"],

"mountPoints": [

{

"sourceVolume": "otel-config",

"containerPath": "/etc/otel-collector-config.yaml",

"readOnly": true

}

],

"portMappings": [

{

"containerPort": 4317,

"hostPort": 4317

},

{

"containerPort": 4318,

"hostPort": 4318

}

]

}

Ensure the OTLP Exporter in OpenTelemetry Collector is configured to send traces to AWS X-Ray, Jaeger, or another backend.

**4. Run Your .NET Application with the Agent**

Once the ECS task definition is updated, restart your task, and OpenTelemetry will automatically capture ASP.NET Core, HttpClient, Entity Framework, and other common .NET telemetry without modifying your code.