Introduction to Monitoring

In this lesson you will learn how monitoring works in .NET applications and how to use OpenTelemetry to collect, aggregate and export monitoring data to popular tools like Prometheus and Grafana.

How Monitoring Works in .NET

A great benefit from the .NET platform is its built-in support for tracking important metrics of your application, that you can later use to monitor and potentially alert when things are not going as expected. Let's see how .NET applications can take advantage of this.

Any time your application code needs to start tracking metrics, it creates a Meter, which is the entry point for your microservice to create a group of instruments. These instruments record the numeric measurements that are needed to calculate metrics.

A few of the available instruments are the Counter, which tracks a value that increases over time; the Observable Gauge, which reports non-additive values when the instrument is being observed; and the Histogram, which can be used to report arbitrary values that are likely to be statistically meaningful.

Your application code can use these instruments to produce the relevant metrics. These can be custom metrics directly related to your microservice, like a counter that keeps track of successful purchases, or they can be metrics that are automatically tracked by the .NET platform, like a histogram that tracks the duration of inbound HTTP requests.

Third party libraries like MassTransit will also bring in many additional metrics, like the counter of the number of messages received.

After your application has been instrumented, it's time to enable the collection of all those relevant metrics. Here's where you can use the OpenTelemetry libraries to collect and aggregate metrics that you can then export to popular monitoring tools like Prometheus.

OpenTelemetry will enable an endpoint in your microservice, which by default is under the /metrics, which Prometheus can then query at a configurable interval, to retrieve all the available microservice metrics.

You can then monitor your metrics in Prometheus, and even create alerts that trigger any time those metrics are not in the expected ranges.

To get an even better view of your monitoring data, you can connect your Prometheus server to Grafana, which is a powerful and very popular tool to create many kinds of dashboards that can report your metrics in real time with beautiful visualizations that are highly customizable.

As you can see, the combination of the .NET Platform with OpenTelemetry, Prometheus and Grafana make is very straightforward to enable monitoring for your microservices.

In the next lesson you will instrument one of your microservices using the built-in .NET libraries and OpenTelemetry.