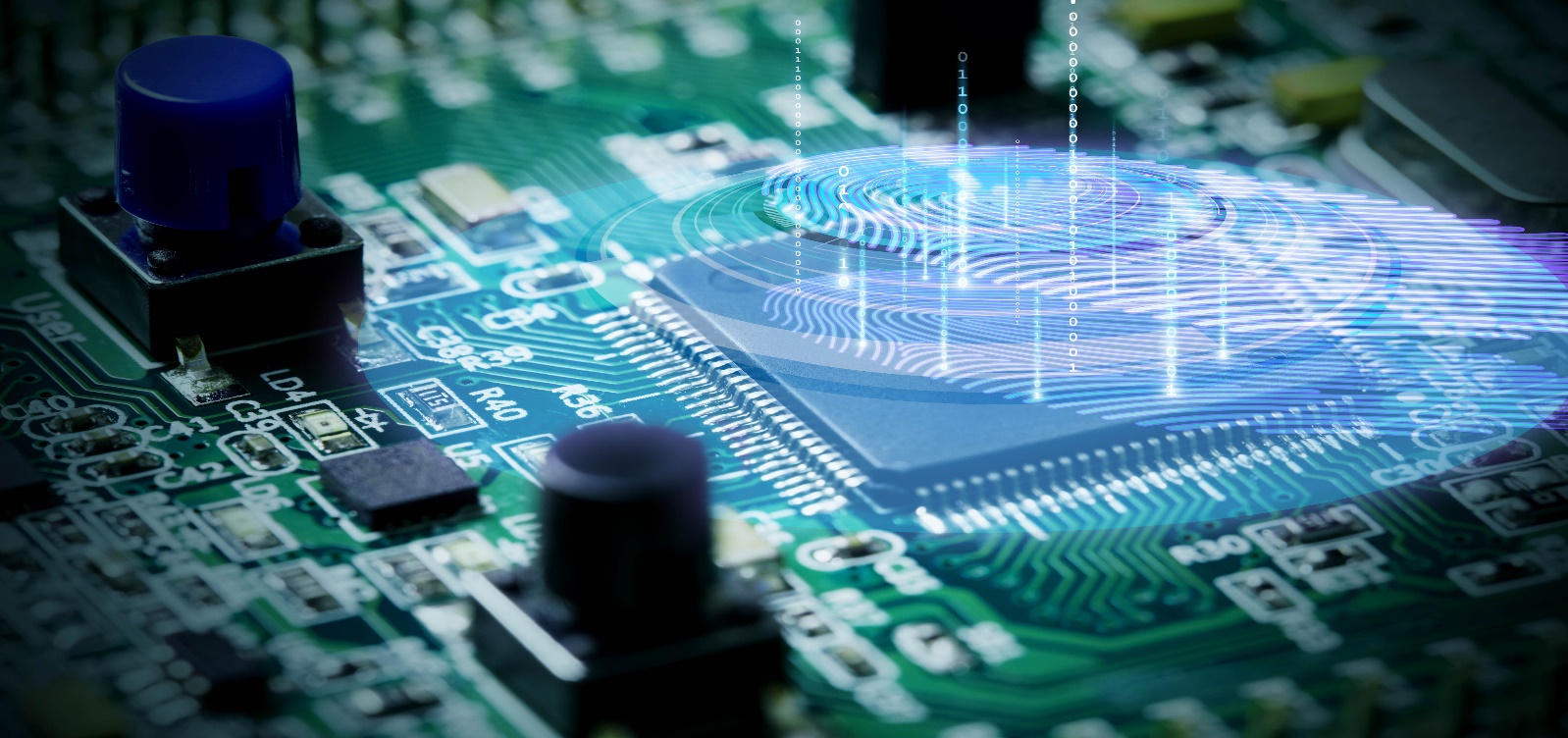
Logo

Description automatically generated



ASP.NET Core 6 Worker Service with OpenTelemetry

Hi there!

Today we will create a worker service application using ASP.NET Core 6 and integrate it with open telemetry. We will also export telemetry data to Jaeger and Zipkin collectors.

**OpenTelemetry (OTel or OTEL)**

OpenTelemetry is a collection of tools, APIs, and SDKs. It is used to instrument, generate, collect, and export telemetry data (metrics, logs, and traces) to help you analyse your software’s performance and behaviour. It is 100% free and open source.

OpenTelemetry offers several components, most notably:

* APIs and SDKs per programming language for generating and emitting telemetry.
* Collector component to receive, process and export telemetry data
* OTLP protocol for transmitting telemetry data

**Advantages**

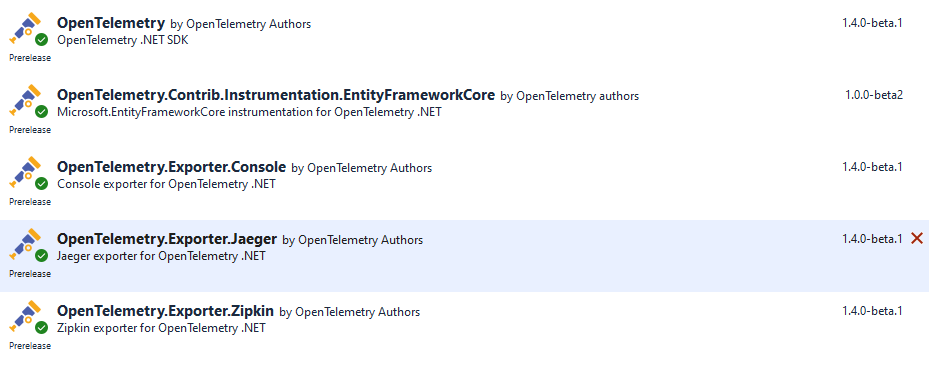
* Consistency: OpenTelemetry provides a consistent path for capturing telemetry data and transmitting it to a backend without changing instrumentation, offering a de facto standard for adding observability to cloud-native apps.
* Simpler choice: OpenTelemetry merges the code of OpenTracing and OpenCensus, each of which offered a different approach to achieving observability. There is no risk from switching to OpenTelemetry if you were previously using one or the other. It is also backward compatible.
* Streamlined observability: OpenTelemetry lets developers view application usage and performance data from any device or web browser. This convenient interface makes it easy to track and analyse observability data in real time.
* The greatest benefit is gaining the observability necessary to achieve business goals. It consolidates the telemetry data needed to determine if systems are functioning properly, understand where issues may be compromising performance and fix root causes, potentially before service is interrupted – resulting in greater stability and reliability for supporting business processes.

OpenTelemetry feeds collected data into an AI engine to automatically produce actionable insights. The AI engine can continuously analyse data that OTEL captures along with other useful and desirable data and look for anomalies throughout the full stack without any human intervention. There are four components of observability:

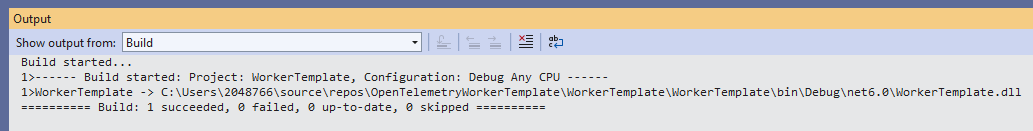
* Instrumentation: Measuring tools collect telemetry data from a CPU, container, service, application and any other component of your system that produces data. This provides and maintains visibility across your entire infrastructure as you add new components and data types.
* Data correlation: The telemetry data collected from your system is processed and correlated. This creates context and enables automated or custom data curation for the observability tool’s visualization.
* Incident response: Automated technologies forward data about outages to the relevant people and teams.
* AIOps: Machine learning automatically aggregates, correlates and prioritizes incident data. This filters out alert noise, letting teams detect issues and accelerate incident response.

First create an ASP.NET Core 6 Worker Service application. Then, add the following packages to it.

Packages to be installed:

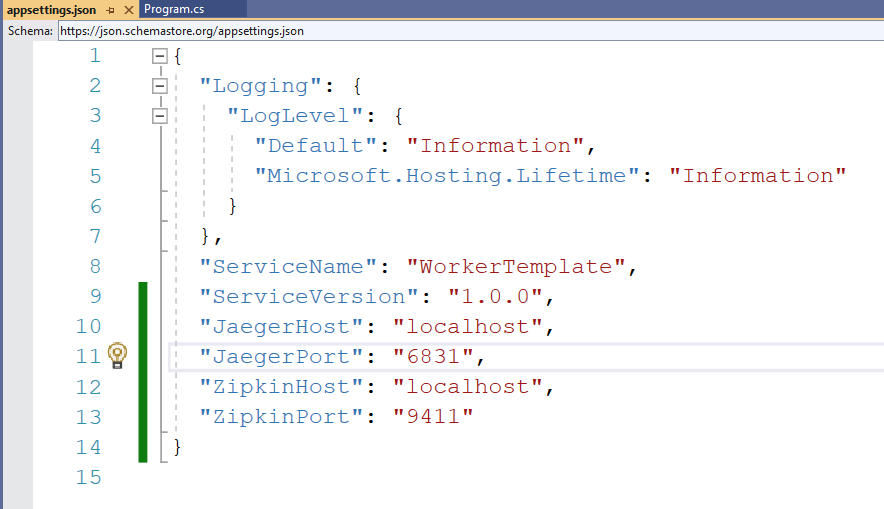


Then, right-click the project and click Build in order to build the project.



This will create the DLL files in bin and obj folders.

Now, add the following codes to appsettings.json file.

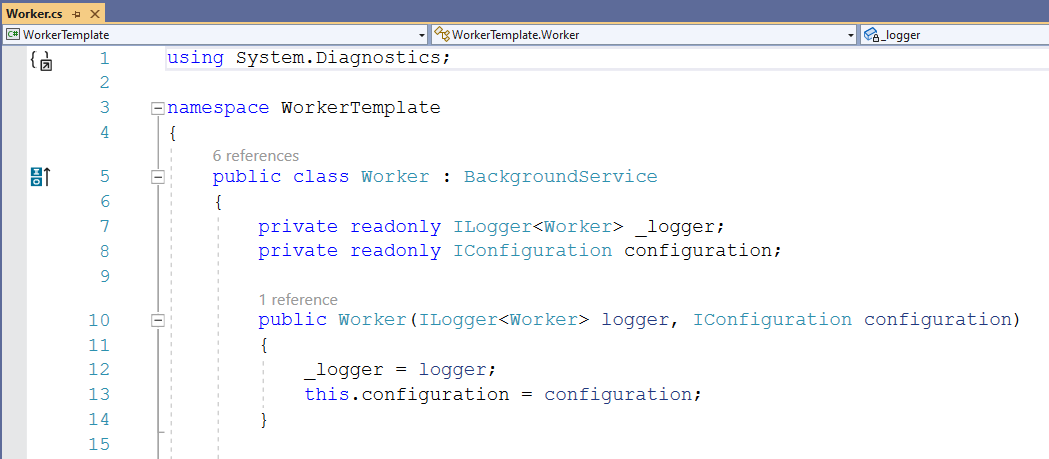


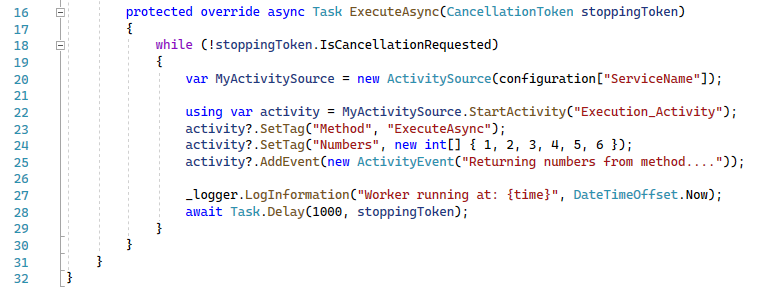
Now, add the following codes to Program.cs





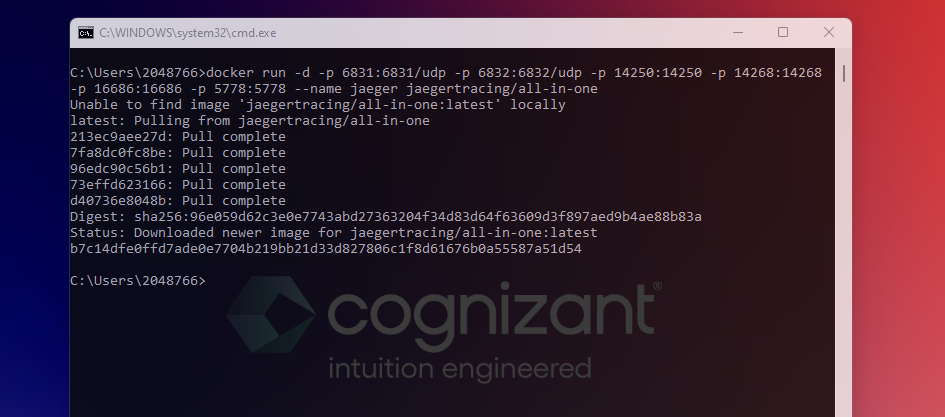
Add the following codes in Worker.cs:





Now, we need to run the OTEL opensource collectors. Then, run the following commands in terminal to run Jaeger and Zipkin.

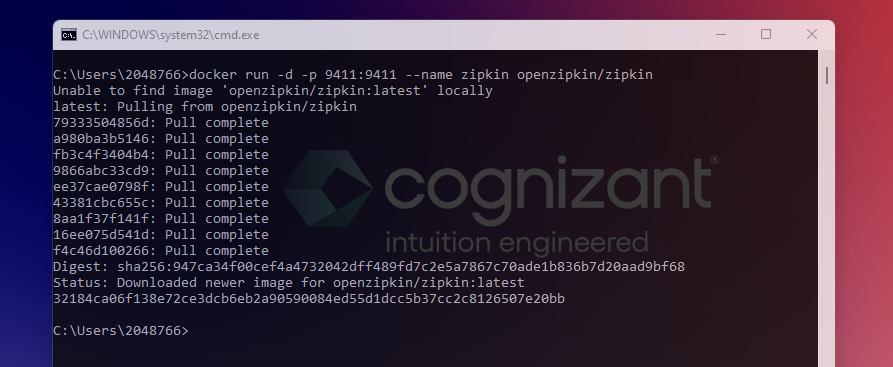
docker run -d -p 6831:6831/udp -p 6832:6832/udp -p 14250:14250 -p 14268:14268 -p 16686:16686 -p 5778:5778 --name jaeger jaegertracing/all-in-one



Then, check this URL: <http://localhost:16686/>

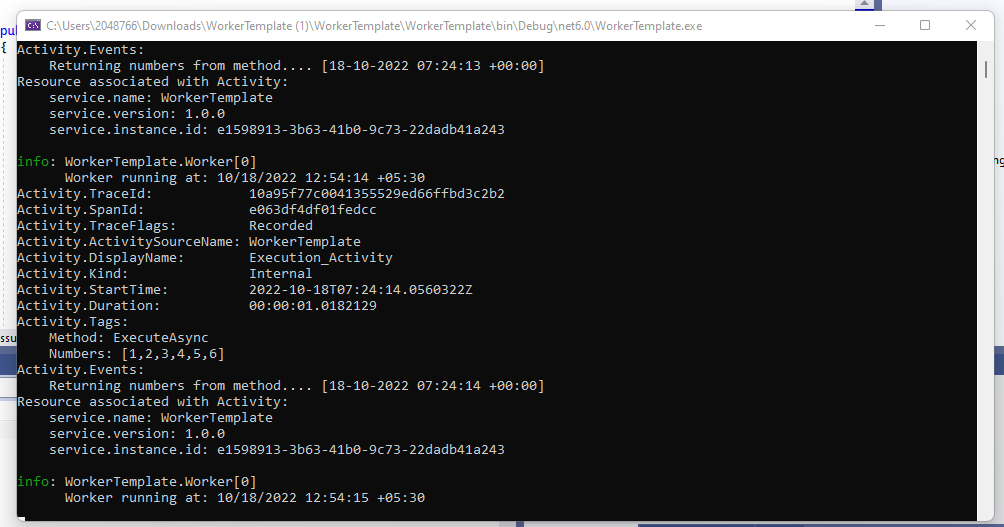
Now, run the following commands to run Zipkin in your computer.

docker run -d -p 9411:9411 --name zipkin openzipkin/zipkin



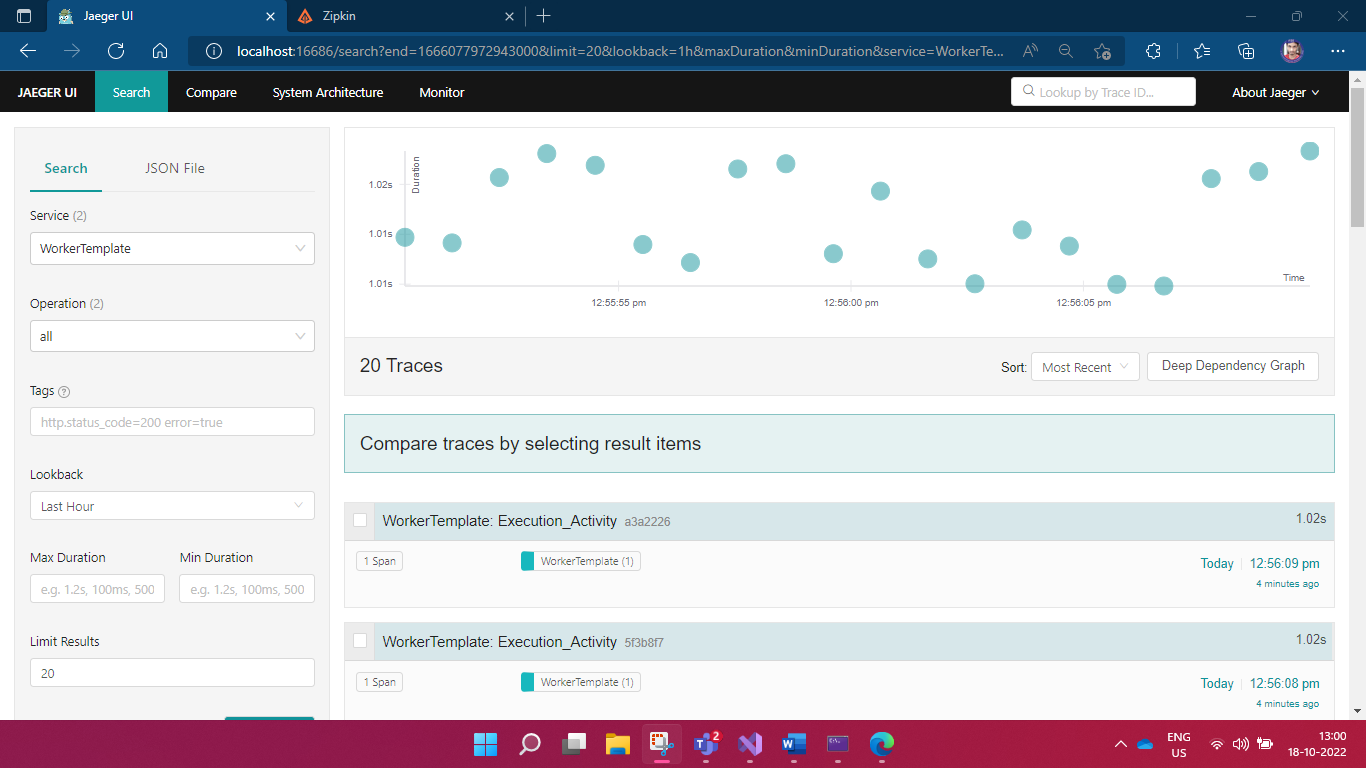
Then, check this URL: <http://localhost:9411/>

Now, it’s time to run the Worker Service and check the traces in Jaeger and in Zipkin.

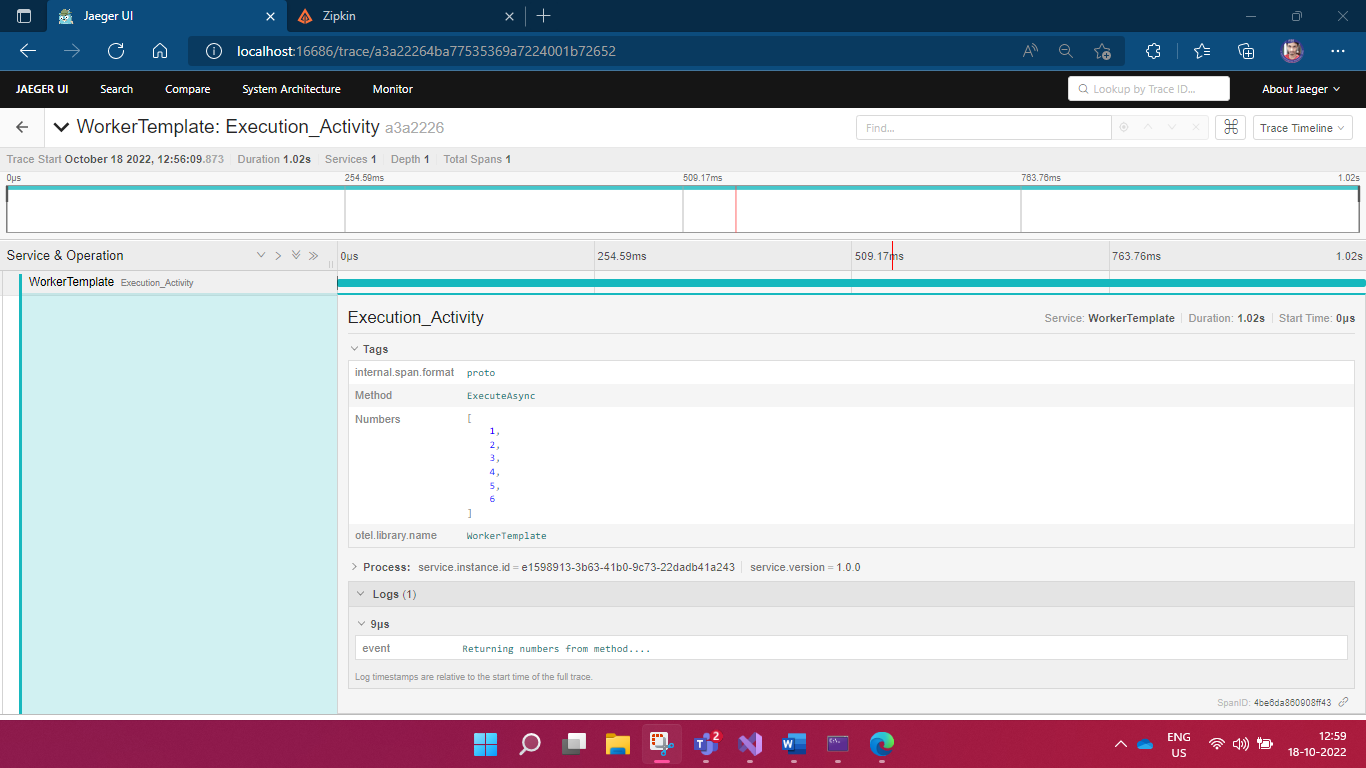


Now, refresh Jaeger and Zipkin URLs.

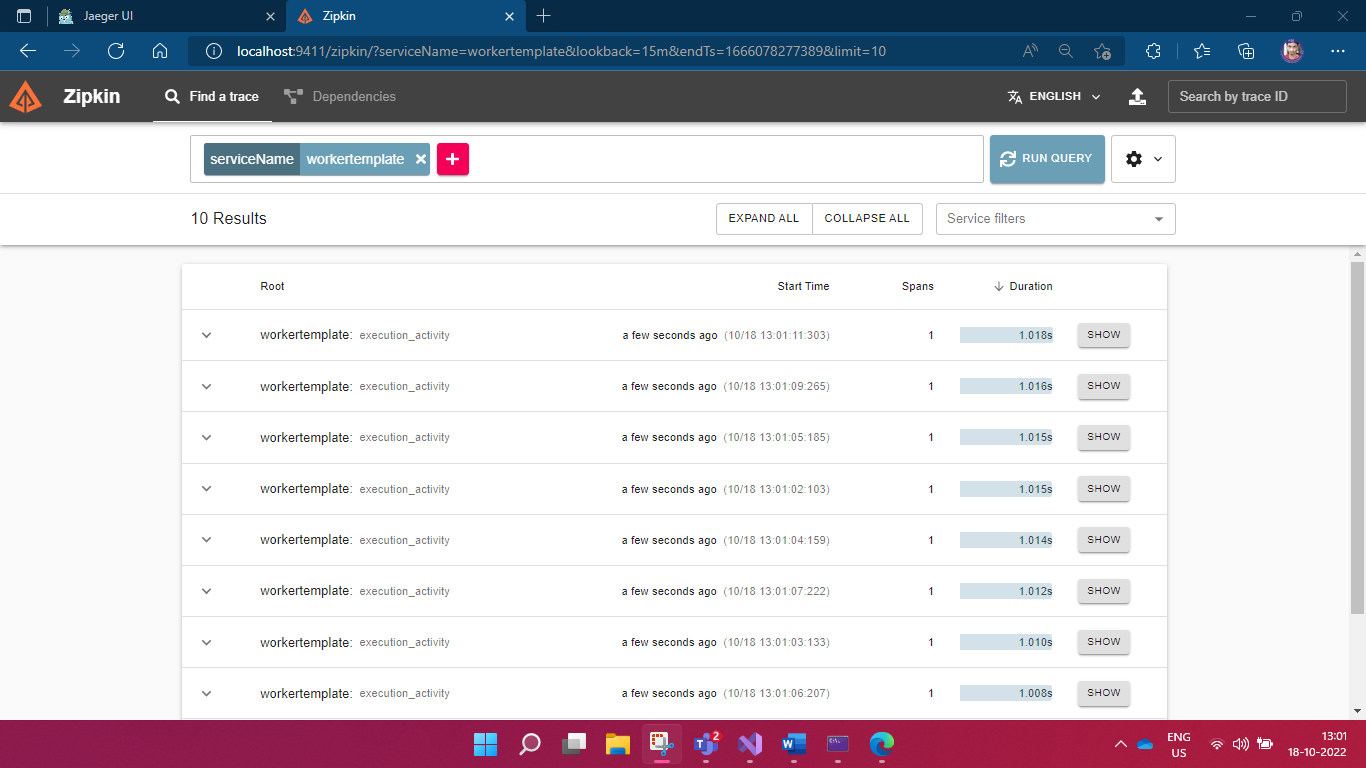
Select the service and click Find Traces.

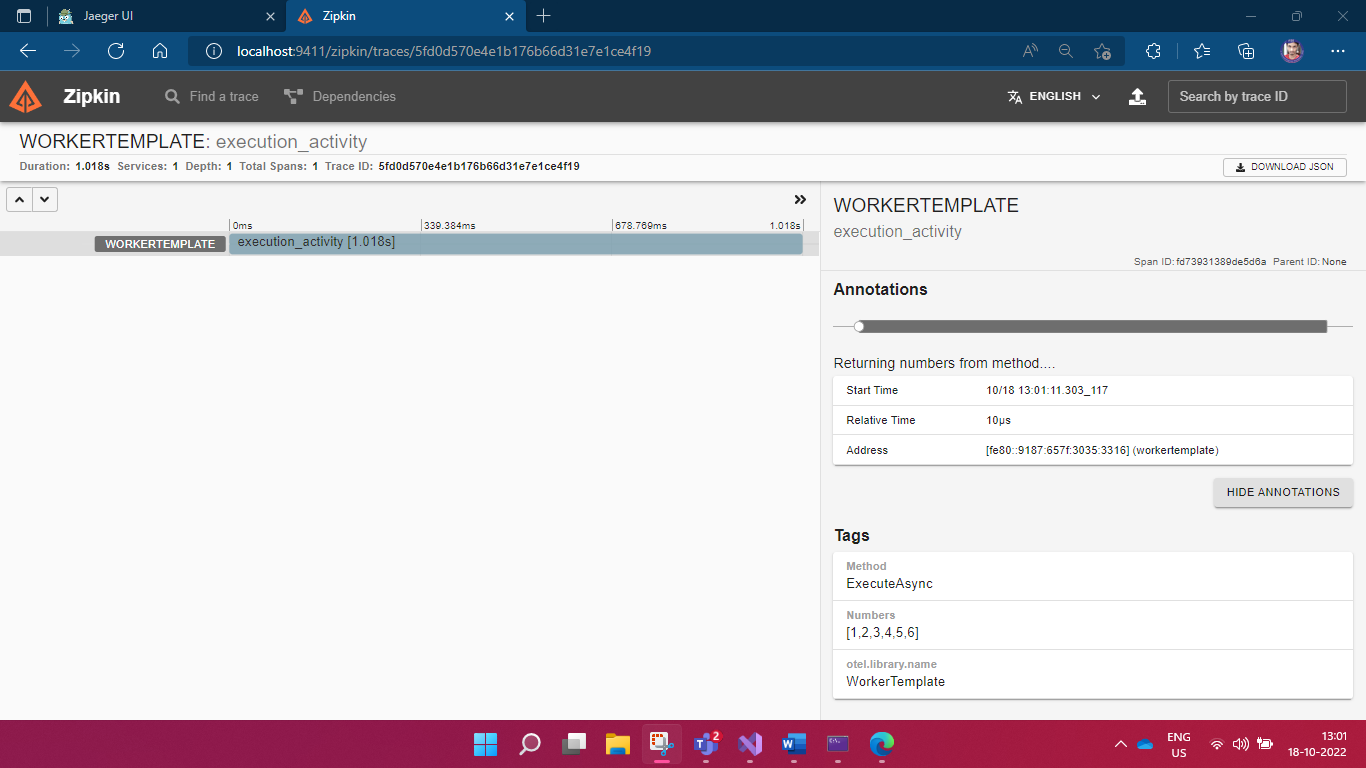


Select a trace and see the results.



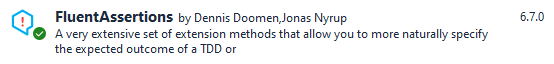
Now, check the Zipkin. Select the service name and click Run Query.

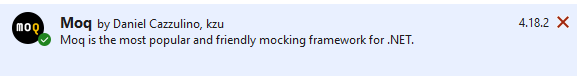


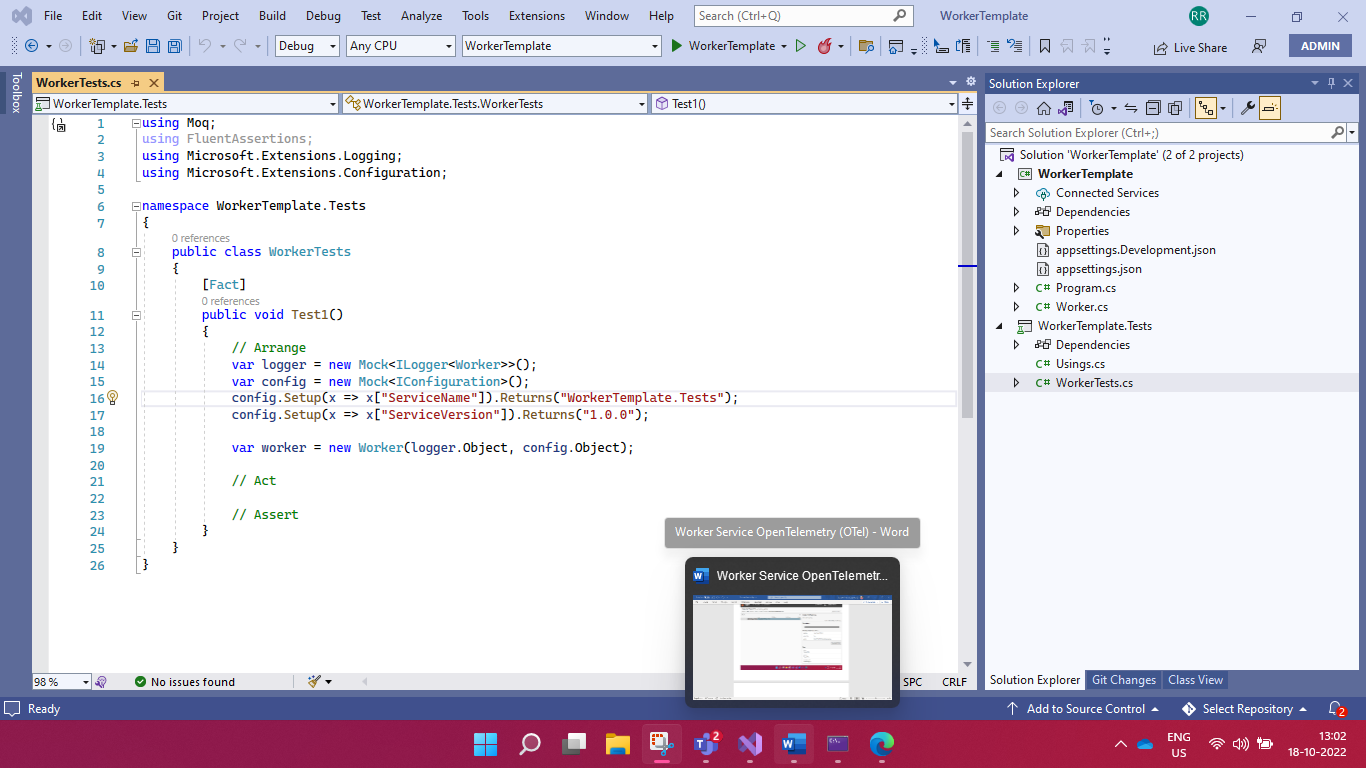


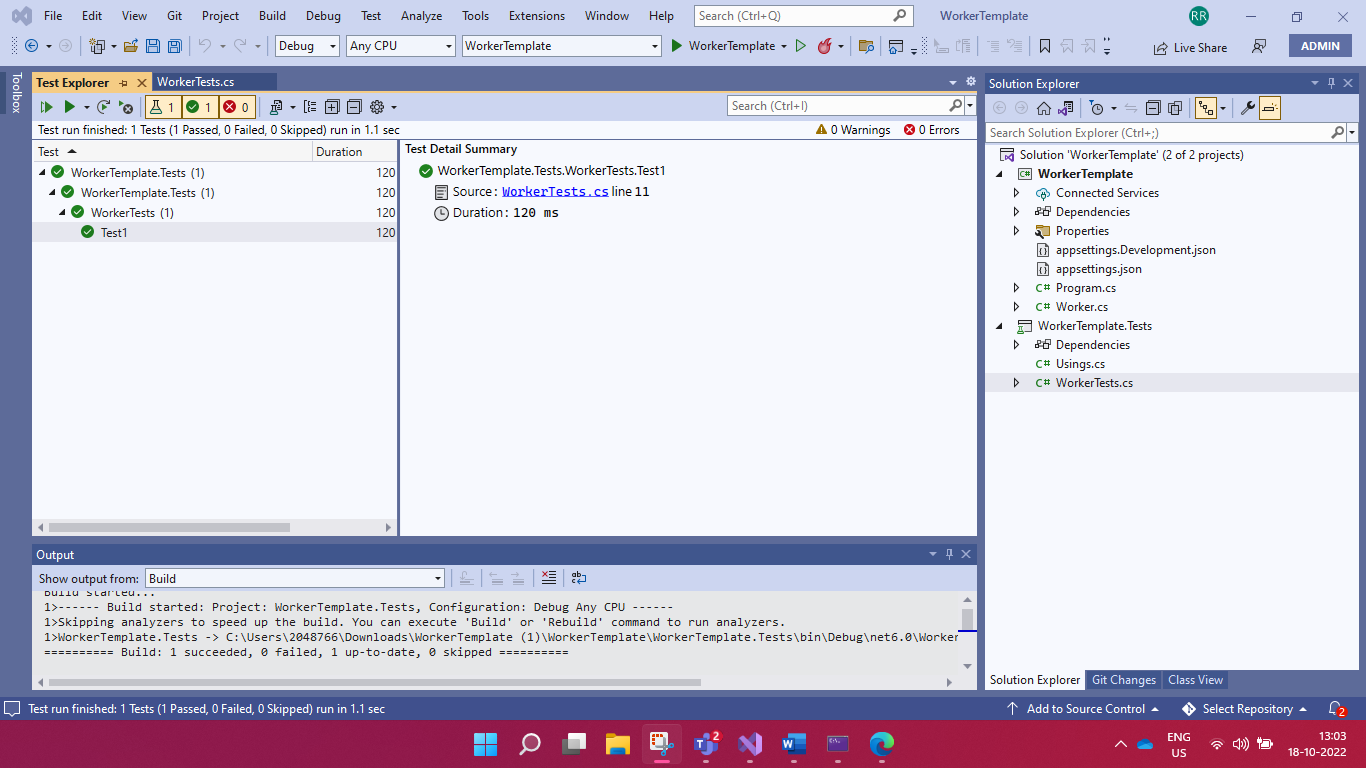
Unit tests (xUnit C#):

Packages to be Installed:



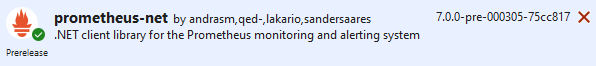




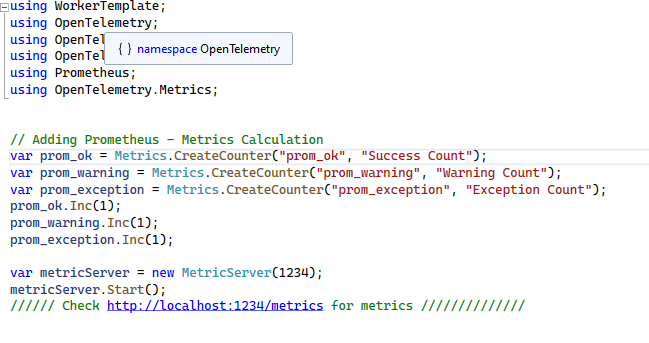


Adding Prometheus

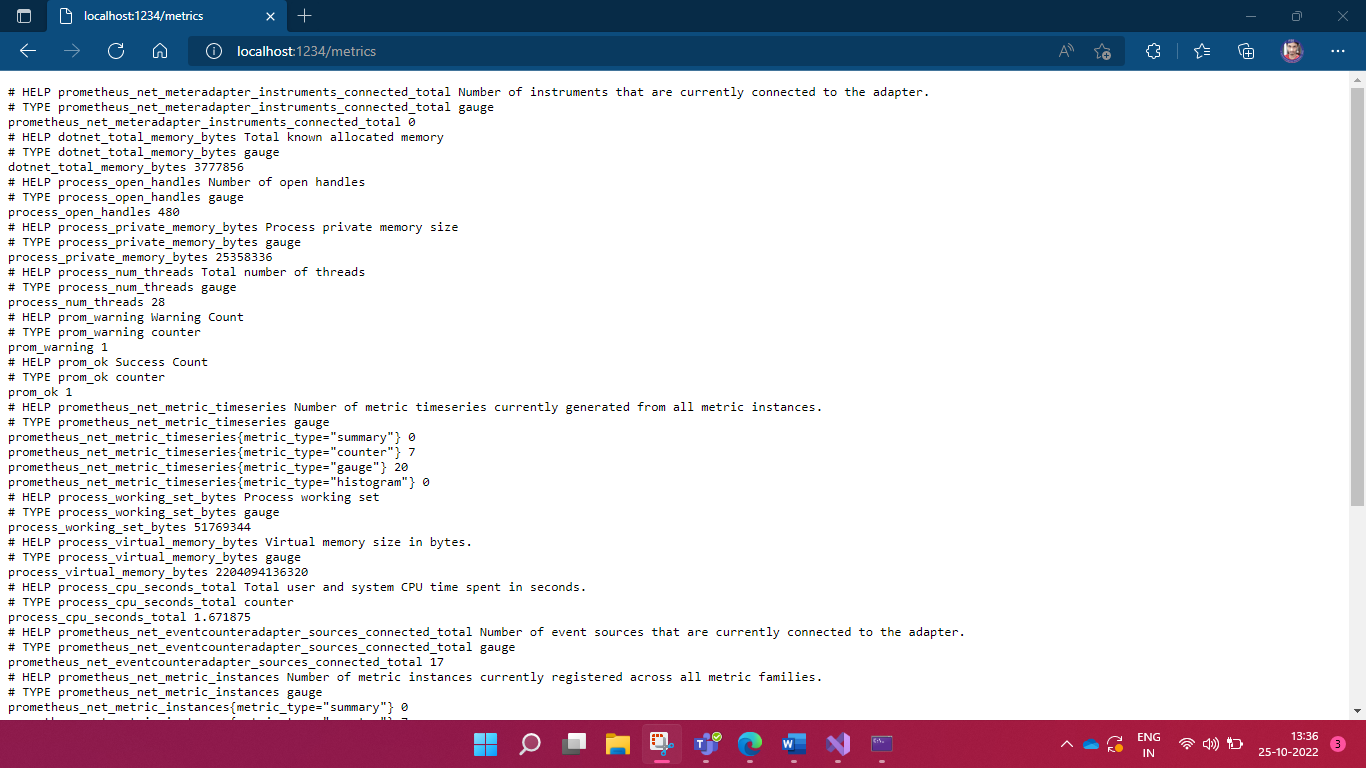
Packages To Be Installed:



In Program.cs



Browse: <http://localhost:1234/metrics>



Thank You