



**Christian Nagel**

<https://csharp.christiannagel.com>



# Logs, Metrics, Distributed Tracing, and OpenTelemetry

**THRIVECONF.COM**  
**#ThriveITConf**

3rd – 4th June 2025





# Zahvaljujemo se našim sponzorjem

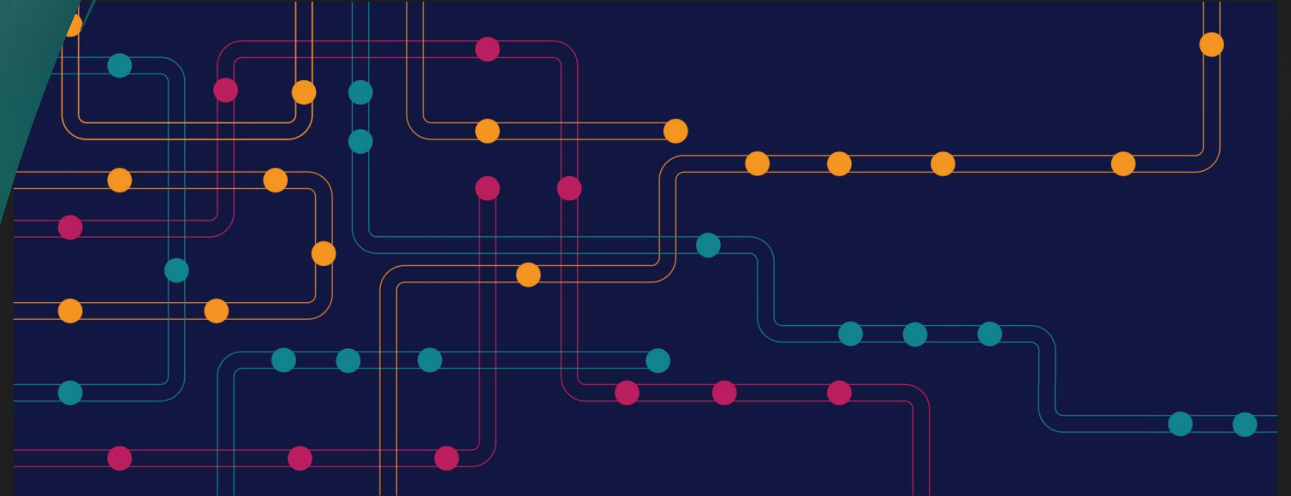


# Christian Nagel

- Training
- Coaching
- Consulting
- Development
  
- Microsoft MVP
- [www.cninnovation.com](http://www.cninnovation.com)
- [csharp.christiannagel.com](http://csharp.christiannagel.com)
- @christiannagel



<packt>



1ST EDITION

## Pragmatic Microservices with C# and Azure

Build, deploy, and scale microservices efficiently  
to meet modern software demands



# Topics

- Implementing
  - Logging
  - Metrics
  - Distributed Tracing
- Displaying
  - .NET Aspire Dashboard
  - Prometheus and Grafana
  - Azure Application Insights



# Logging

# Logging use cases



MONITOR APPLICATION  
BEHAVIOR



DIAGNOSE ISSUES

## 3 important parts

### Providers

- Where is the message written to

### Sources

- Who is logging

### Levels

- How important is the message

# Log Providers

- Console
  - Write messages to the console
- Debug
  - `System.Diagnostics.Debug`
- EventSource
  - Name: Microsoft-Extensions-Logging
- EventLog
  - Only on Windows



# Hierarchical Named Log Sources

- *Microsoft.EntityFrameworkCore*
- *Microsoft.AspNetCore.Server.Kestrel*
- *Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker*
- *Yarp*
- *System.Net.Http*
- ...

# Log Levels

- *Trace*
- *Debug*
- *Information*
- *Warning*
- *Error*
- *Critical*

# Filtering

Filter logging based on providers, sources, and log levels

## Creating Logs *ILogger* and *ILoggerFactory*

Inject *ILogger<T>* to  
specify the log  
category

Inject *ILoggerFactory*  
to specify the log  
category as string



# Structured / Semantic Logging

Don't use

Don't use interpolated strings with logging

Log

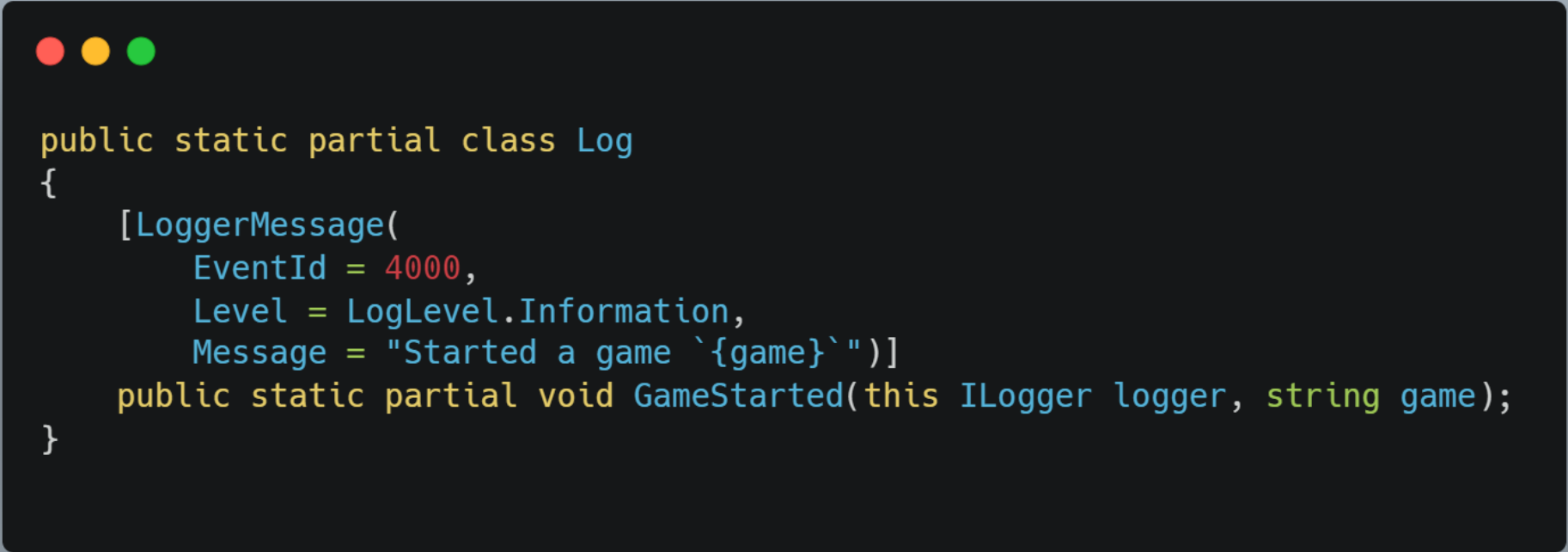
Log methods use templates

Be

Placeholder names can be used for filtering

# Source generated logging

Prefer source generated logging



```
public static partial class Log
{
    [LoggerMessage(
        EventId = 4000,
        Level = LogLevel.Information,
        Message = "Started a game `{game}`")]
    public static partial void GameStarted(this ILogger logger, string game);
}
```

# Check event log level for complex parameters

```
[LoggerMessage(
    EventId = 4005,
    Level = LogLevel.Information,
    Message = "Returned {NumberGames} games using {Query}")]
private static partial void QueryGames(this ILogger logger,
    int numberGames, string query);

public static void QueryGames(this ILogger logger,
    IEnumerable<Game> games, string query)
{
    if (logger.IsEnabled(LogLevel.Information))
    {
        QueryGames(logger, games.Count(), query);
    }
}
```





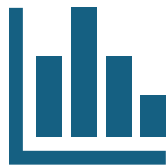
Metrics



# Metrics use cases



Monitor system  
behavior



Monitor solution  
behavior



Trigger alerts



Performance  
monitoring

# Built-in .NET counters

- NET Runtime
- DI Container
- System.Net
- ASP.NET Core
- EF Core
- ...

## .NET Tool

- dotnet counters

# Metrics APIs

## System.Diagnostics.Tracing.EventSource

- Available since .NET Core 3.1
- Derive from **EventSource**
- Logging and tracing events
- Limited count types
- Not OTEL compliant
- Preview libraries for OpenTelemetry

## System.Diagnostics.Metrics.Meter

- Available since .NET 6
- Use **Meter** class
- Only metrics
- Advanced features
- OTEL compliant
- Supports OpenTelemetry

# Writing custom counts

- Meter
  - Track instruments
- Counter
  - Requests per second or cumulative total
- UpDownCounter
  - Report positive and negative numbers
- Gauge
  - Non-additive values (e.g. room temperature)
- Histogram
  - Arbitrary values likely to be statistic meaningful
- Observable[Counter | UpDownCounter | Gauge]
  - Delegate invoked to retrieve value



# .NET 8 Metrics Updates

- *AddMetrics* – register with DI container
- *IMeterFactory*
  - create multiple meters
- *MetricCollector*
  - Record metric measurements

The background of the slide is a dark green field filled with a complex network of bright green lines. These lines connect numerous light blue circular nodes of varying sizes. The nodes are scattered across the frame, with some appearing as larger, more prominent hubs and others as smaller, peripheral points. The overall effect is a sense of a dynamic, interconnected system, likely representing a distributed network or data flow.

# Distributed Tracing

# Distributed Tracing

- How do services communicate?
- Where did the error originate?
- Which service is a bottleneck?
- What services are dependent on others?







## .NET Activity

- Id
  - Hierarchical structure
  - separated by '.' or '\_'
- Tags
  - Information logged with activity
- Baggage
  - Logged with activity and passed to child
- Events
  - Time-stamped messages (e.g. "done now")

modern information technologies at the top event in Slovenia



# Distributed Tracing

- Trace information across service boundaries
- ActivitySource.StartActivity
  - Start an activity
  - Hierarchical ids (W3C TraceContextId)
  - ID passed with HTTP header
- Add Information
  - Tags (written to activity trace)
  - Baggage (written to activity trace and forwarded to child activities)



# .NET Tools

dotnet trace

- Performance analysis

dotnet counters

- Investigate performance counters

dotnet monitor

- Processes, dumps traces, metrics, logs, operations...

# OpenTelemetry

- <https://opentelemetry.io>
- Instrumentation
  - C++, .NET, Erlang/Elixir, Go, Java, JavaScript, PHP, Python, Ruby, Rust, Swift
- Collectors
  - Vendor-agnostic way to receive, process, export telemetry data



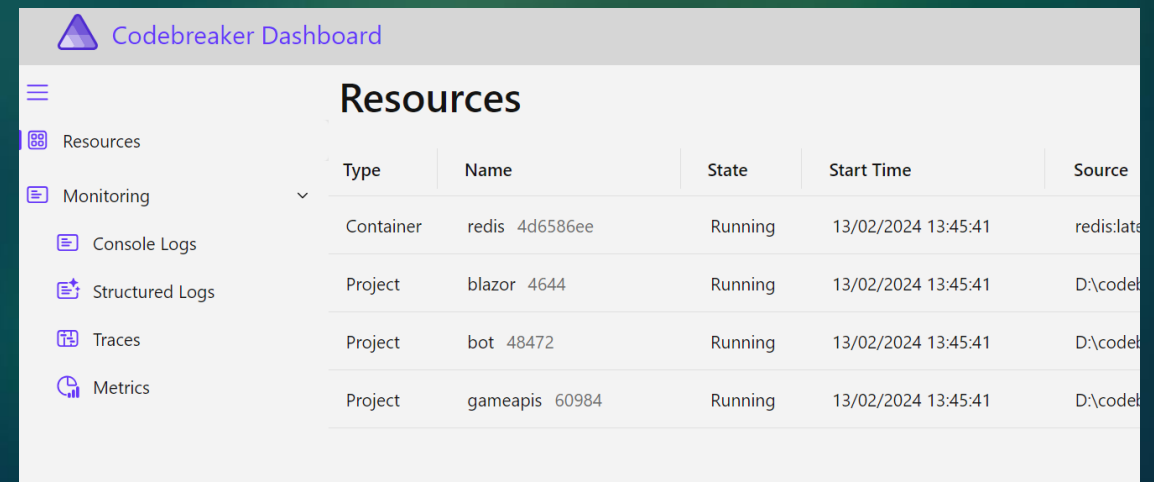






# .NET Aspire Dashboard

- Resources
- Monitoring
  - Console Logs
  - Structured Logs
  - Traces
  - Metrics



The screenshot shows the 'Codebreaker Dashboard' with a sidebar menu on the left containing 'Resources', 'Monitoring', 'Console Logs', 'Structured Logs', 'Traces', and 'Metrics'. The 'Resources' section is active, displaying a table with the following data:

Type	Name	State	Start Time	Source
Container	redis 4d6586ee	Running	13/02/2024 13:45:41	redis:lat
Project	blazor 4644	Running	13/02/2024 13:45:41	D:\codeb
Project	bot 48472	Running	13/02/2024 13:45:41	D:\codeb
Project	gameapis 60984	Running	13/02/2024 13:45:41	D:\codeb



# Collecting data

## Prometheus (Metrics)

- Monitoring and alerting toolkit
- Stores metrics as time series
- PromQL query language

## Loki (Logs)

- Aggregation system from Grafana Labs
- Works with Prometheus and Grafana
- Logs efficiently

## Jaeger (Traces)

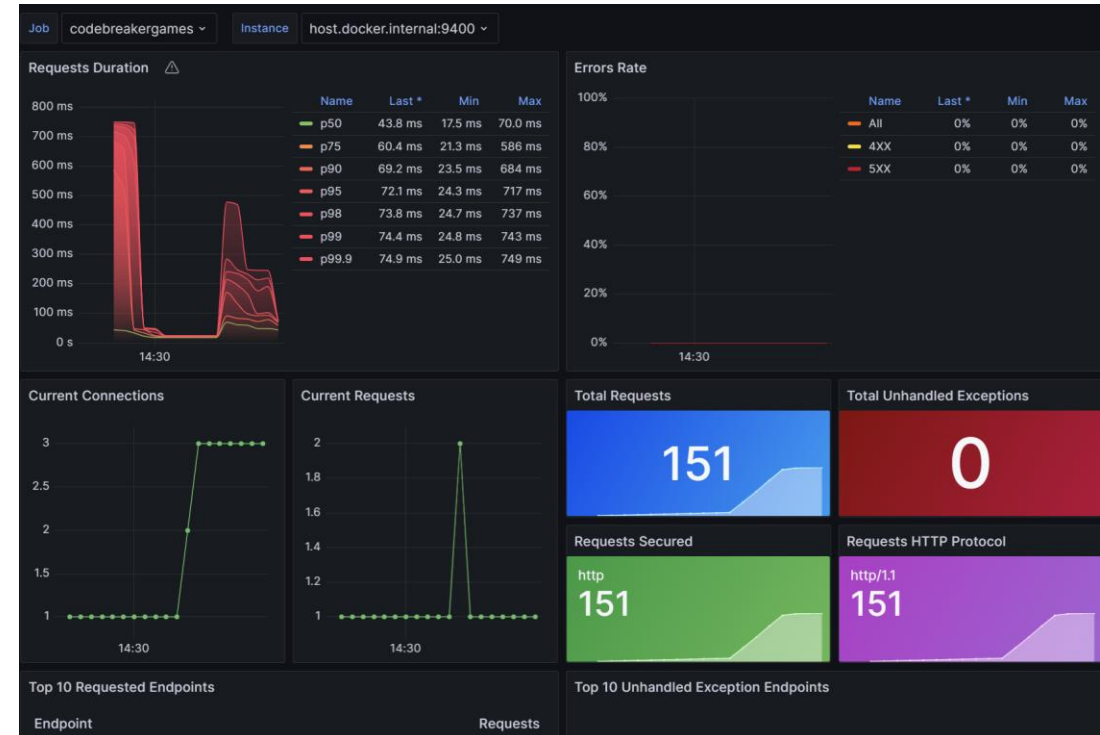
- Web UI to search, view, analyze traces
- Integrated with Grafana and Prometheus

## OpenTelemetry Collector

- Not required
- Central point to collect, process, and route
- Receive, process, export telemetry data (metrics, logs, traces)
- Exports data to backends like Prometheus, Loki, Jaeger

# Visualizing

- Grafana
  - Analysis and visualization platform
  - Connects data sources like Prometheus
  - Customizable dashboards



# Azure services



## Azure Log Analytics

Collects, aggregates, analyzes  
Kusto Query Language (KQL)

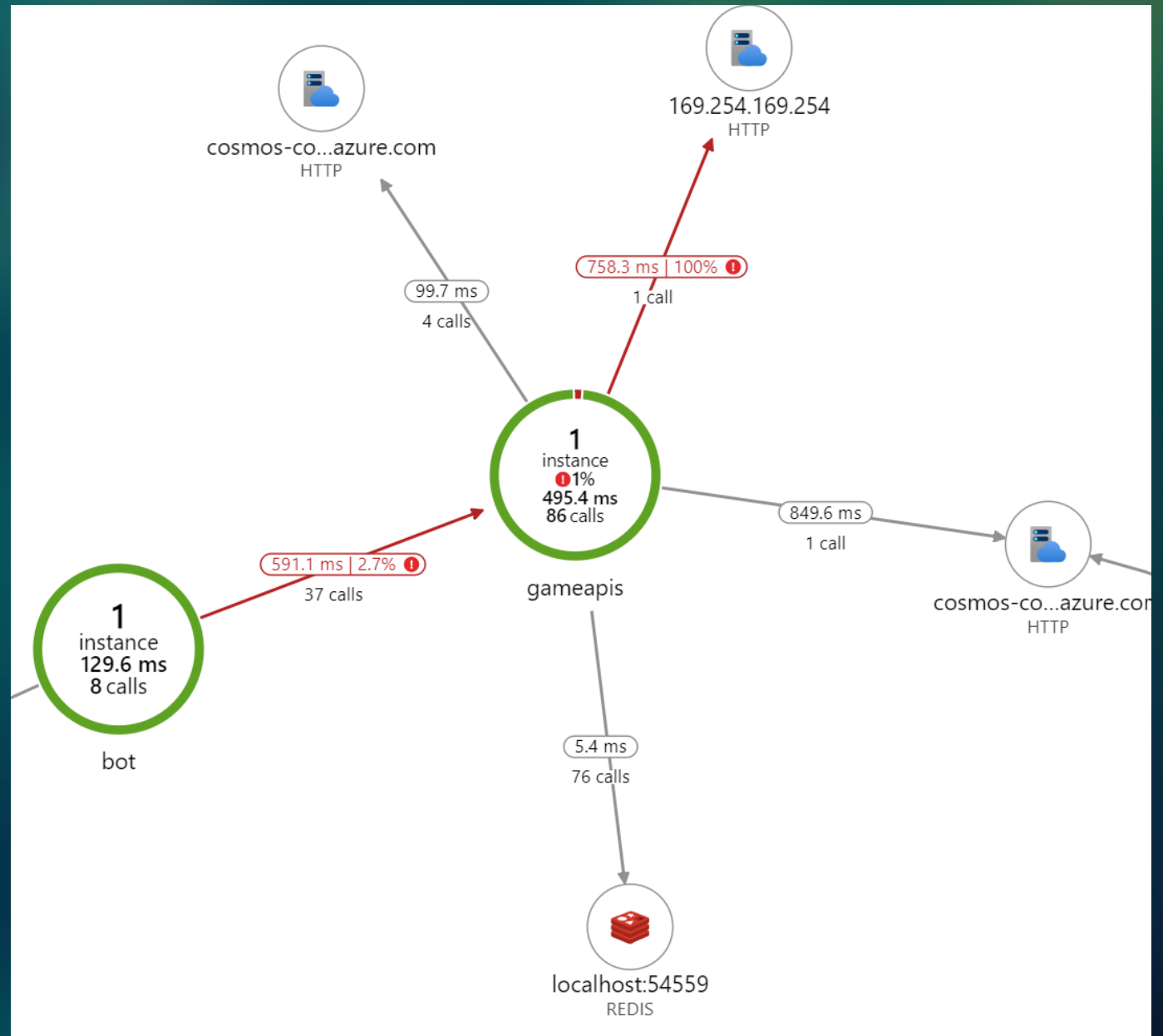


## Azure Application Insights

Application Performance Monitoring (APM)  
Telemetry on availability, performance, usage,  
exceptions

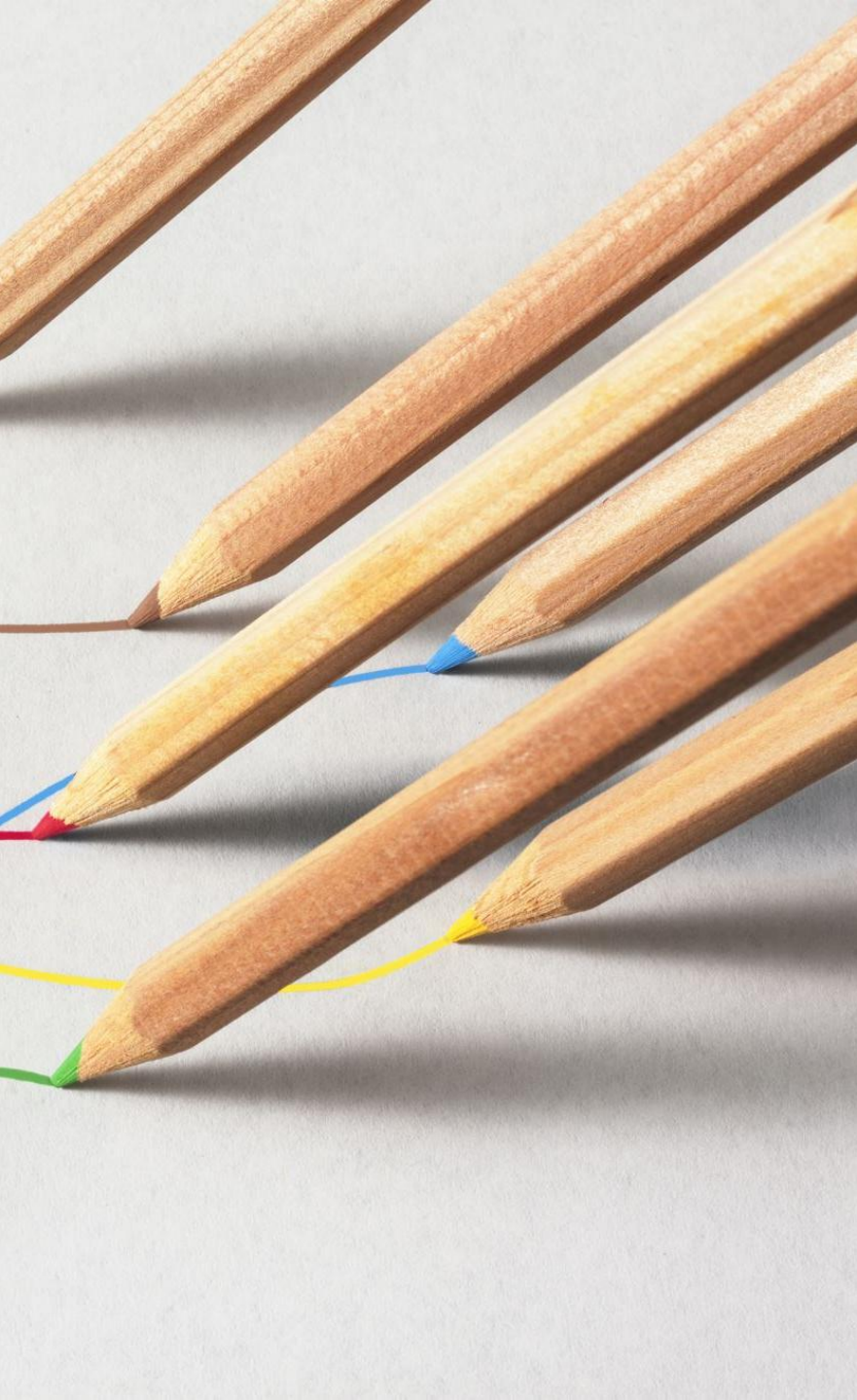
And Azure Log Analytics  
Workspace

# Application Insights



Modern information technologies at the top event in Slovenia





# Summary

- Logging
- Metrics
- Distributed Tracing
- .NET Aspire

Modern information technologies at the top event in Slovenia



<https://github.com/cnilearn/thrive2025>

<https://csharp.christiannagel.com>

<https://www.cninnovation.com>



# Thank you

We greatly value your input. Please take a moment to complete our survey, accessible either through the Run.events app or by the link below:

<https://bit.ly/Thrive25Survey>





See you all next year