



# ORDINA

Ahead of change

## ADVANCED TELEMETRY WITH AZURE MONITOR APPLICATION INSIGHTS

Bart Wullems



[bart.wullems@ordina.be](mailto:bart.wullems@ordina.be)



<https://bartwullems.blogspot.com>







NEED MORE COFFEE

User Story 2943: Implement App | +

dev.azure.com/ordina-ncore/NCore/\_workitems/edit/2943

The 'designing with...' Explore DDD - You... GraphQL Weekly ... What are microserv... Dashboard | edX AWS Training and C... petabridge/akka-b... reWork Kubernetes for Begi... Domain analysis for... »

Azure DevOps ordina-ncore / NCore / Boards / Work items

Search

NCore

Work Items | Back to Work Items

USER STORY 2943  
2943 Implement Application Insights

Bart Wullems 0 comments Add tag

Save Follow Refresh More

Updated by Bart Wullems: Just now

Details Scans Time Log

Description

Click to add Description

Planning

Story Points

Reason New Area NCore

Iteration NCore

Acceptance Criteria

Click to add Acceptance Criteria

Deployment

To track releases associated with this work item, go to [Releases](#) and turn on deployment status reporting for Boards in your pipeline's Options menu. [Learn more about deployment status reporting](#)

Risk

Priority 2

Discussion

Add a comment. Use # to link a work item, ! to link a pull request, or @ to mention a person.

Classification

Value area Business

Development

+ Add link

Link an Azure Repos commit, pull request or branch to see the status of your development. You can also create a branch to get started.

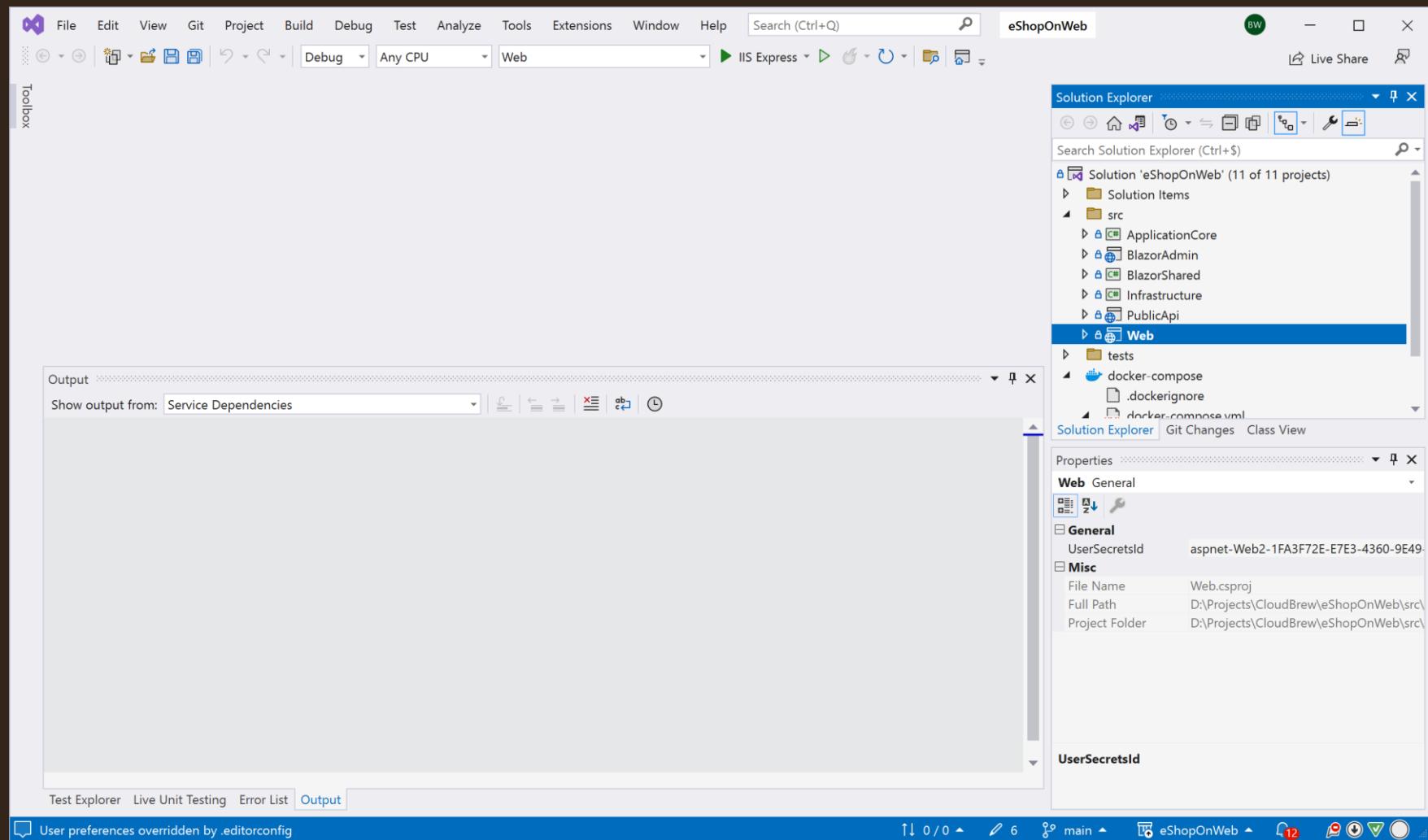
Related Work

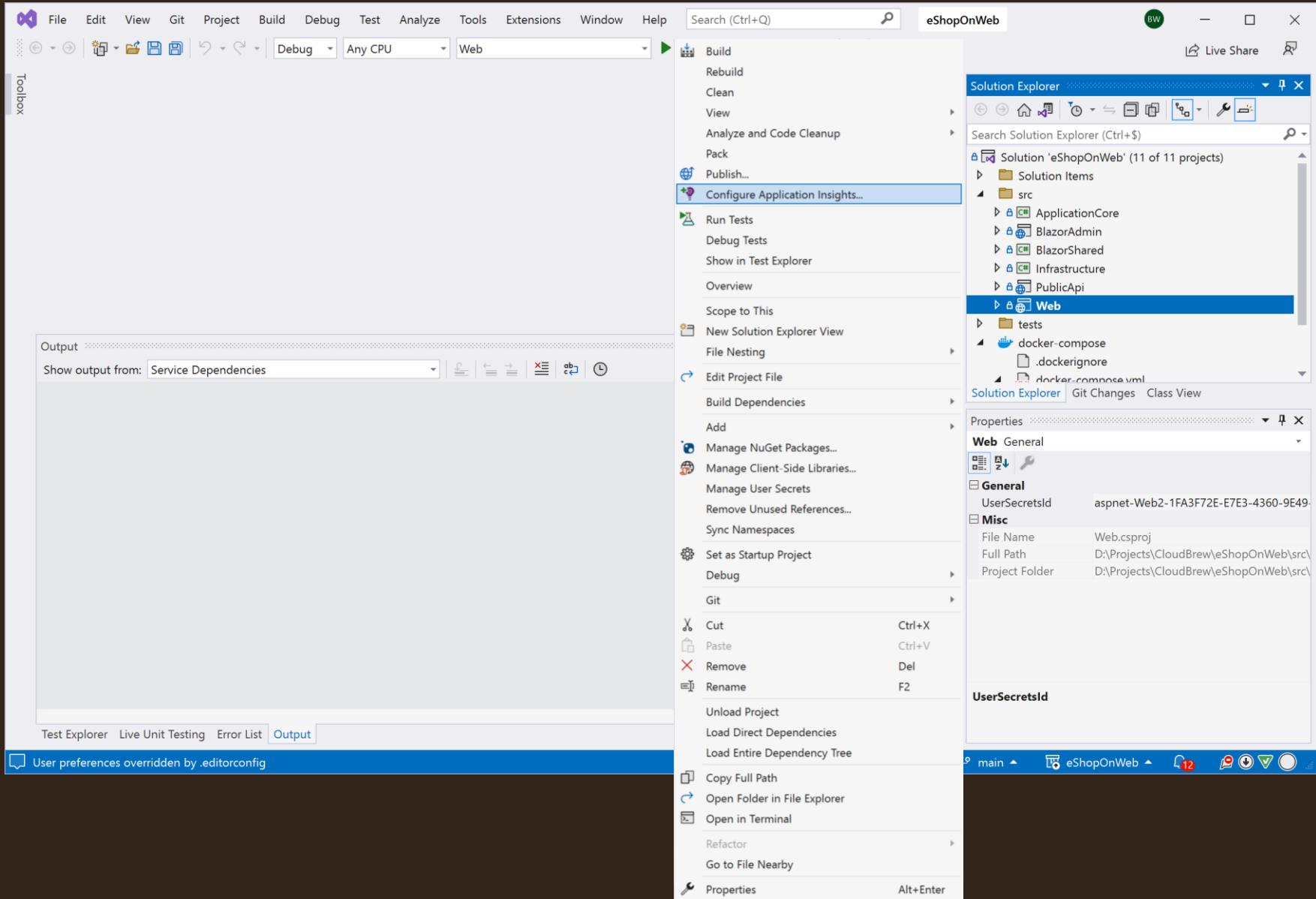
+ Add link

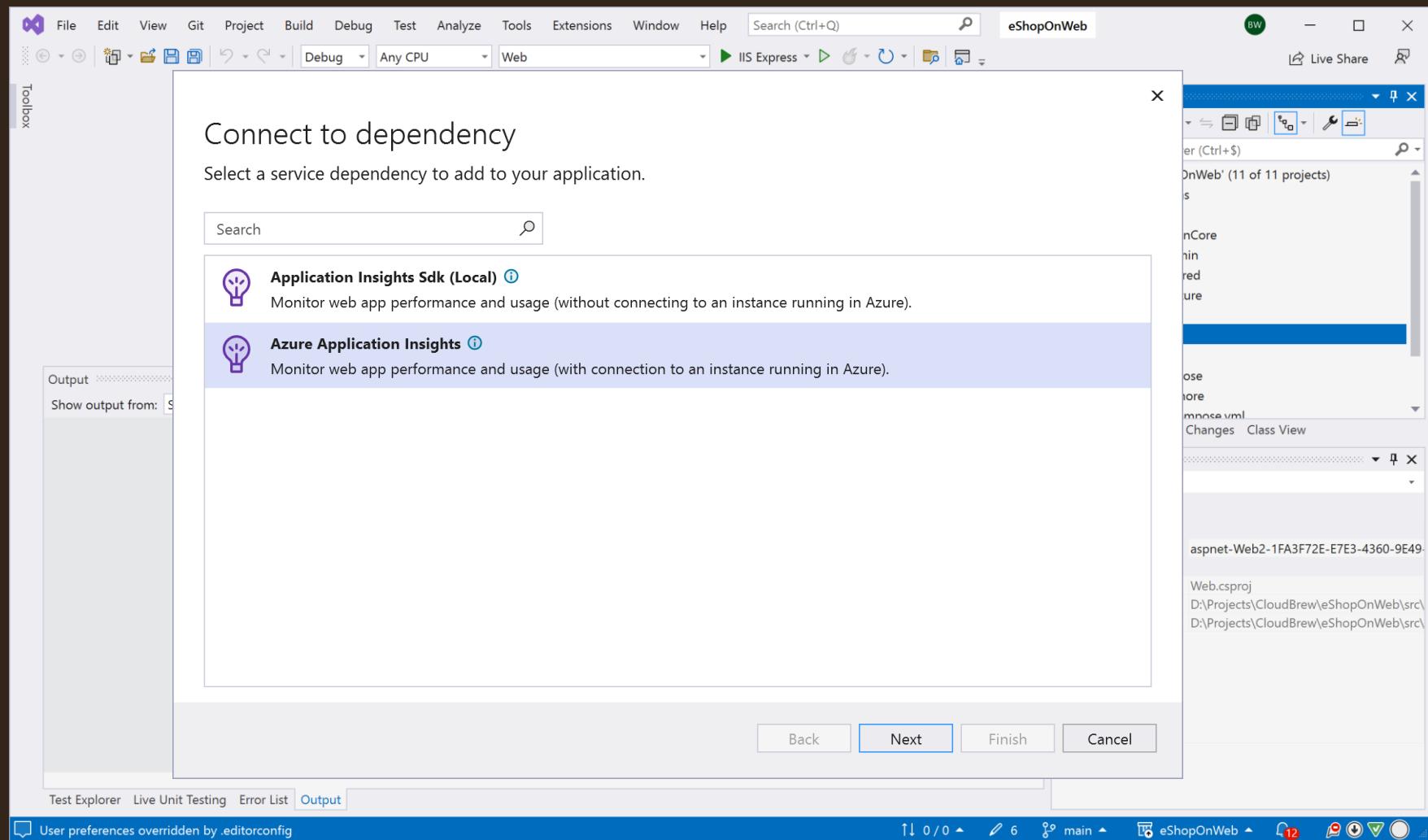
Add an existing work item as a parent

Project settings









File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) eShopOnWeb

Live Share

Microsoft account wullemsb@gmail.com

Re-enter your credentials

## Connect to Azure Application Insights

Select a service dependency to add to your application.

Subscription name

Windows Azure MSDN - Visual Studio Premium

Application Insights instances

Name	Location	Resource group
marvintfsbot	East US	Default-Web-NorthEurope
loadtestappinsight	East US	Default-ApplicationInsights-CentralU!
appinsightsdemo	East US	Default-ApplicationInsights-CentralU!
cloudbreweshopweb	West Europe	Default-Web-WestEurope

+ Create new

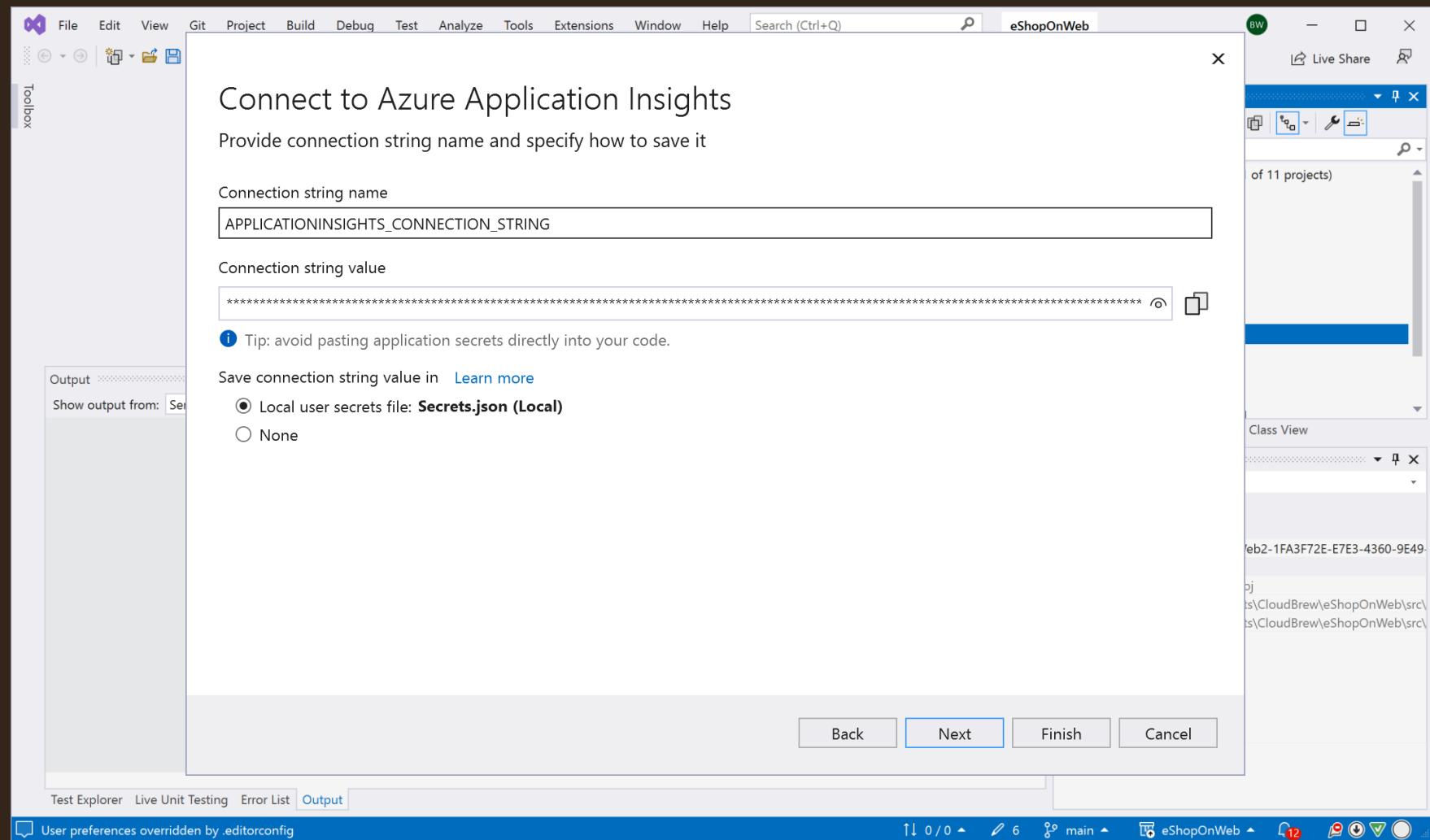
Output Show output from

Back Next Finish Cancel

User preferences overridden by .editorconfig

11 0 / 0 main eShopOnWeb 12

Test Explorer Live Unit Testing Error List Output





# ORDINA

Ahead of change



## ADVANCED TELEMETRY WITH AZURE MONITOR APPLICATION INSIGHTS

Bart Wullems



[bart.wullems@ordina.be](mailto:bart.wullems@ordina.be)



<https://bartwullems.blogspot.com>

# AGENDA



Why application monitoring?

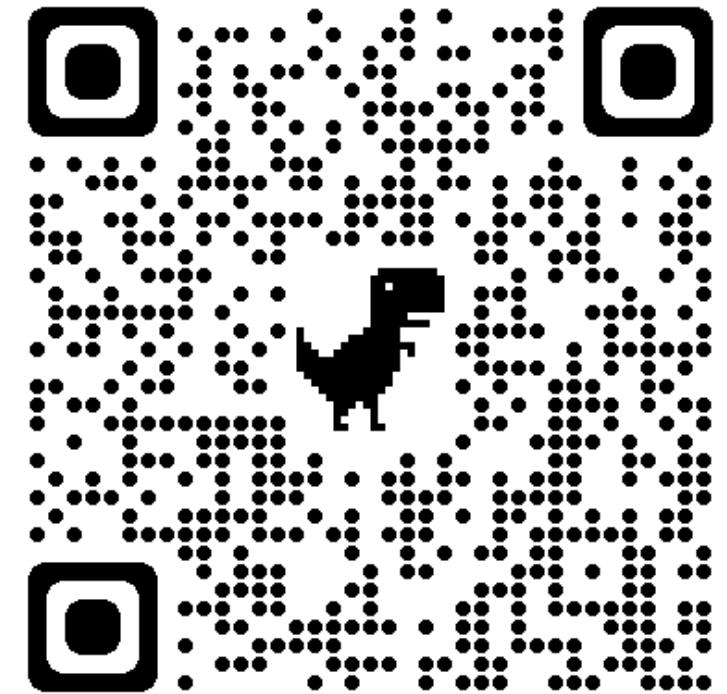
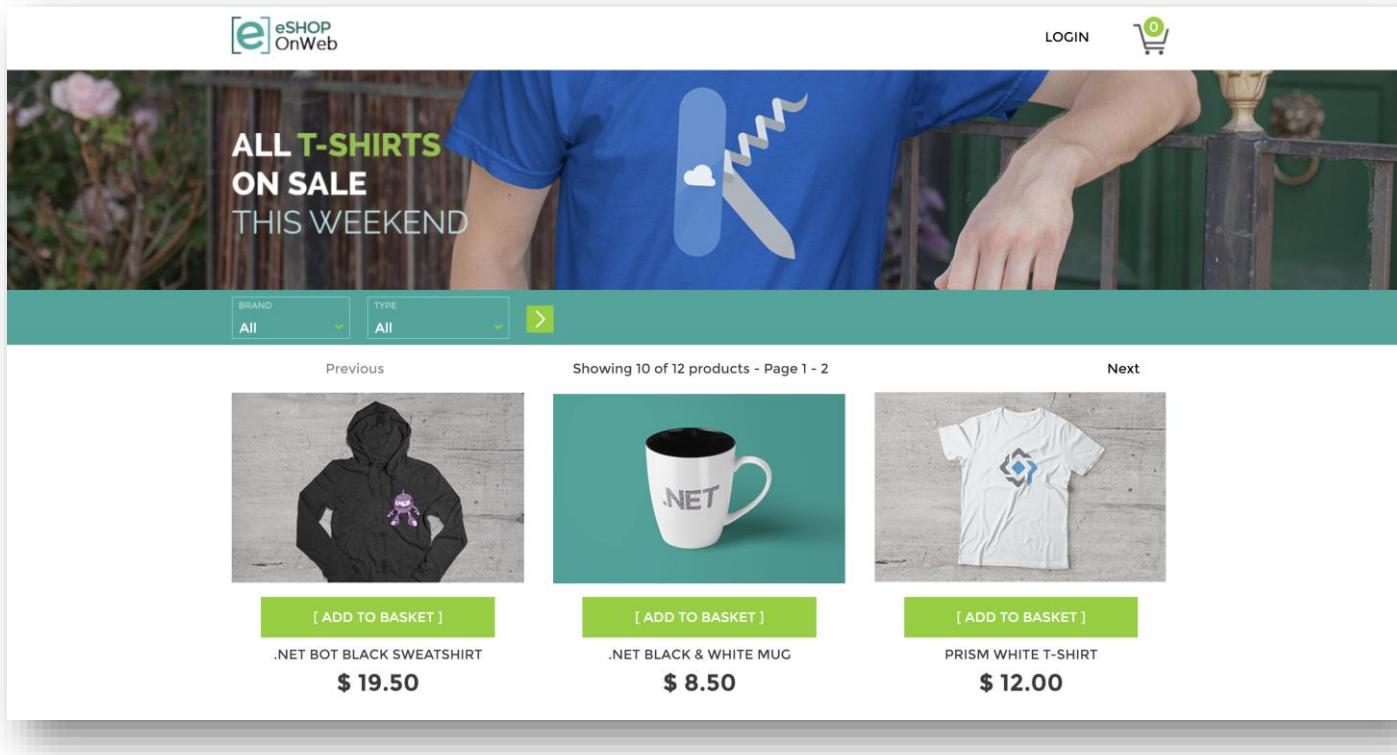


Application  
Insights

What is it?  
Feature  
overview



Hands-on demos



<https://cloudbreweshopweb.azurewebsites.net>



# WHY APPLICATION MONITORING?

# WHY APPLICATION MONITORING?

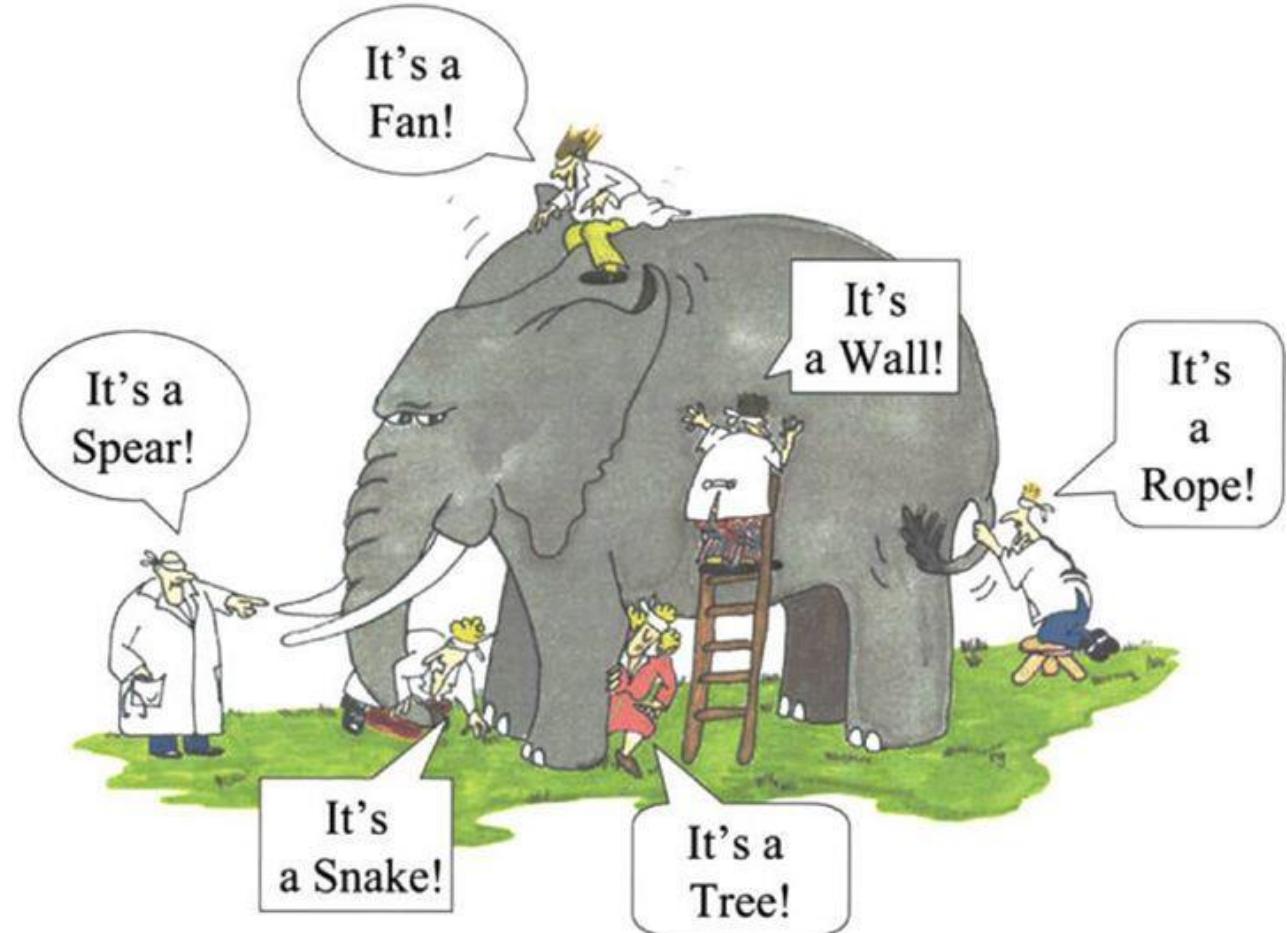
## Find out:

- What's going on inside your application service
- how your application is performing
- how your application is used
- what errors users face
- events that need your reaction



# IS IT BETTER THAN LOGGING?

- Building good logging is a challenging task
- Most of logging frameworks don't provide analyzable logs
- Log correlation may be difficult
- Logs are not a perfect fit for all telemetry(statistics,...)

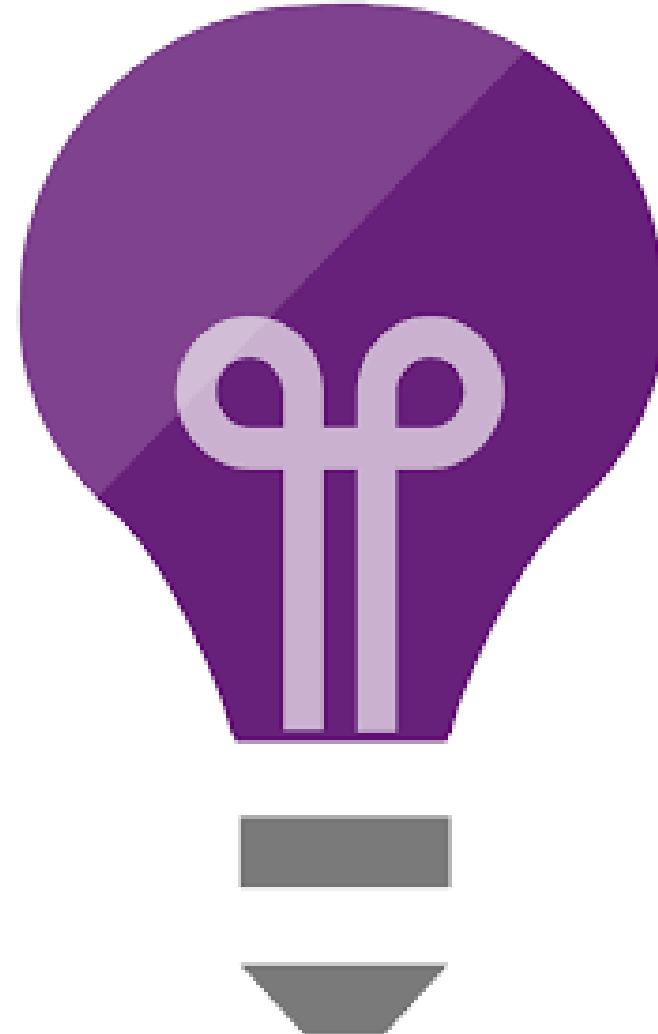


# WHY TELEMETRY DATA?

Useful for:

- Developers: find bugs and performance bottle necks
- Testers: discover areas that need more testing
- UX designers: find out how users are using applications
- Administrators: get ready for load growth
- Support: find problematic areas and be ready to support users
- Business: see how sales and business operations are going

# APPLICATION INSIGHTS



# WHAT IS APPLICATION INSIGHTS?

1

Telemetry is collected at each tier: mobile applications, server applications and browser



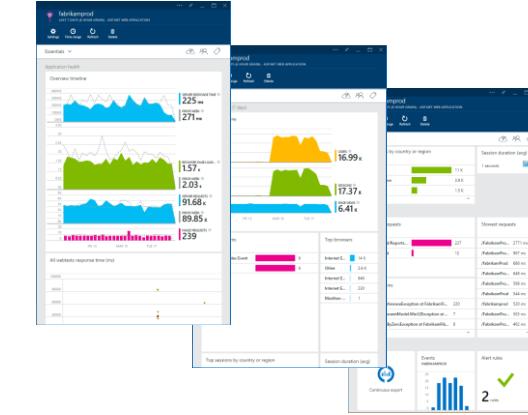
2

Telemetry arrives in the Application Insights service in the cloud where it is processed & stored

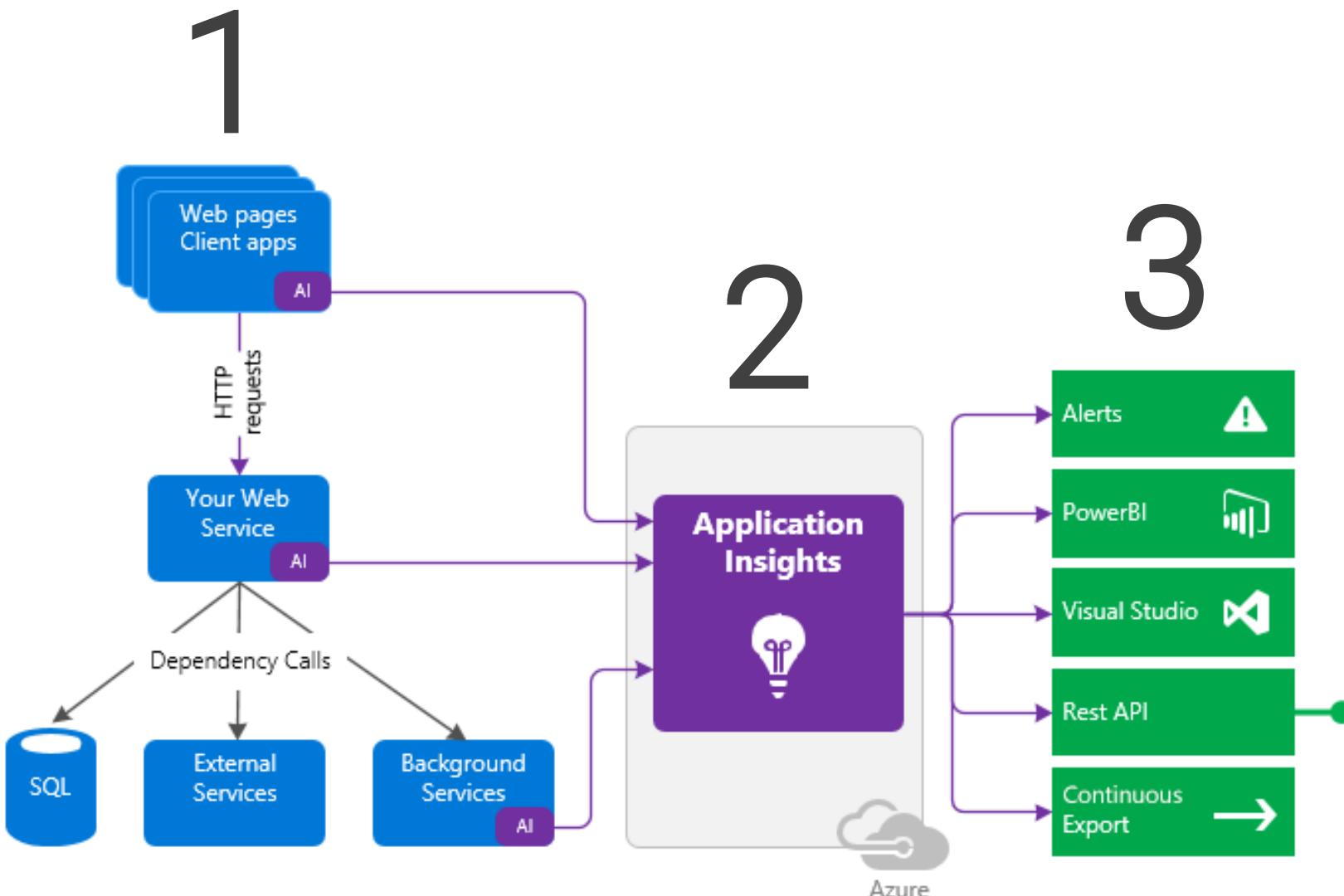


3

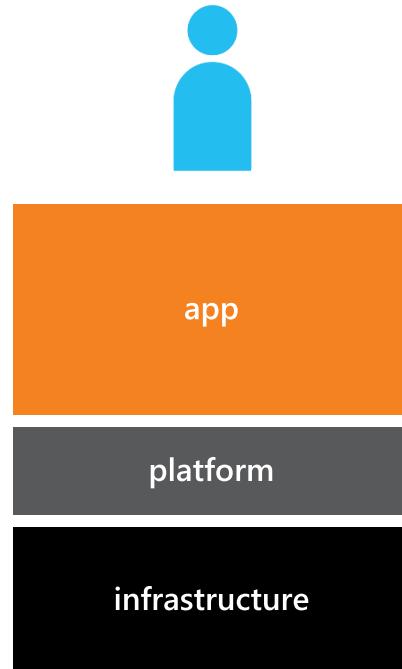
Get a 360° view of the application including availability, performance and usage patterns



# WHAT IS APPLICATION INSIGHTS?



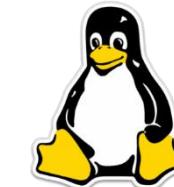
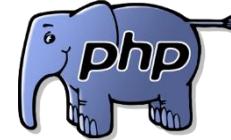
# SOURCES OF TELEMETRY



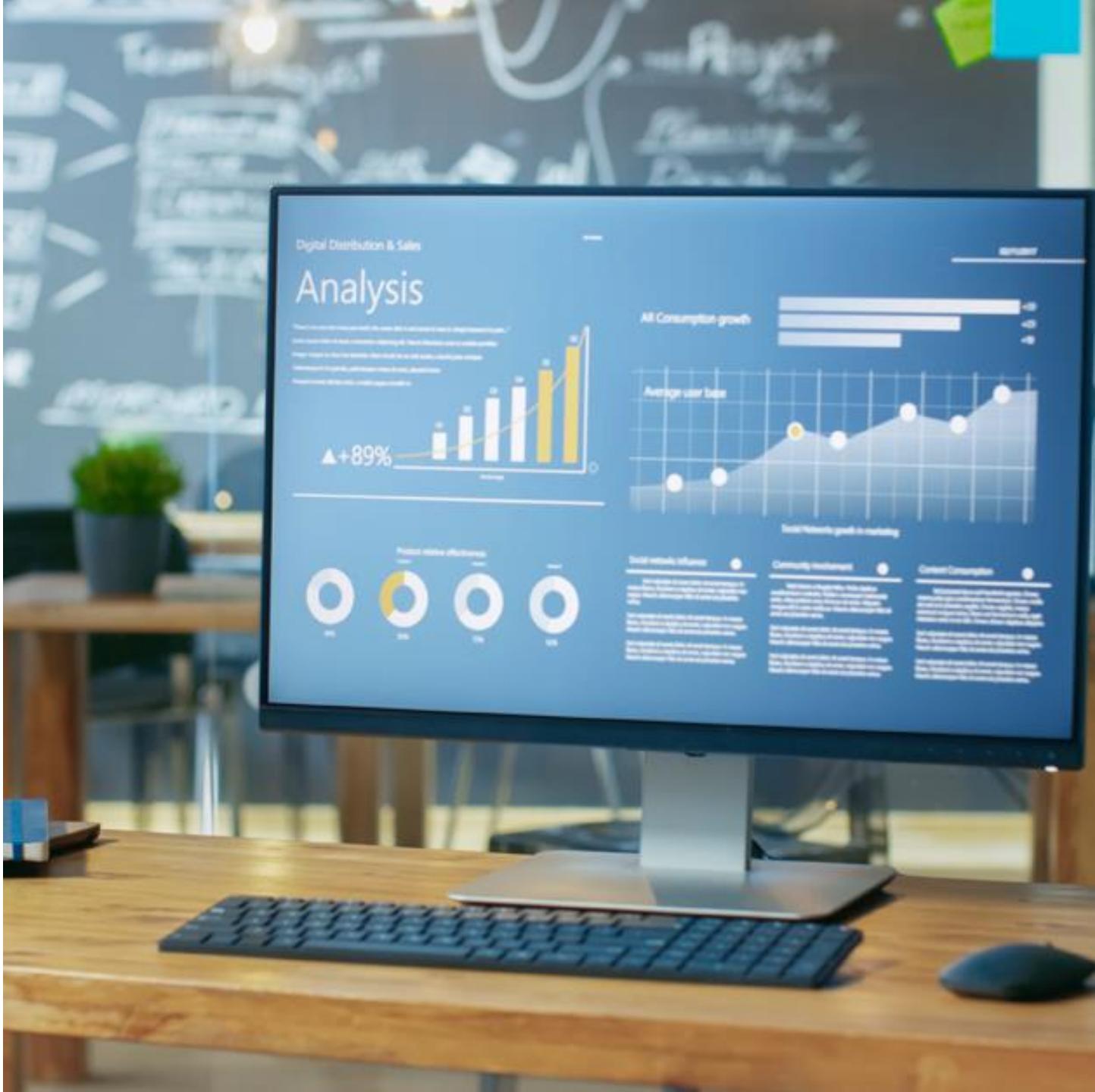
- 1 **Outside-in monitoring**  
URL pings and web tests from 16 global points of presence
- 2 **Observed user behavior**  
How is the application being used?
- 3 **Developer traces and events**  
Whatever the developer would like to send to Application Insights
- 4 **Observed application behavior**  
No coding required – service dependencies, queries, response time, exceptions, logs, etc.
- 5 **Infrastructure performance**  
System performance counters

# SUPPORTED PLATFORMS

- Any app support(mobile, desktop, SPA, middleware, ...)
- Multiple platforms including: ASP.NET, Java/J2EE, iOS, Android, Windows, as well as OSS technologies such as Node.JS, PHP, Ruby, Python, etc.
- Open source SDK:  
<https://github.com/microsoft/ApplicationInsights-dotnet>
- Integrates with most logging frameworks:  
[Log4Net](#), [nLog](#), [System.Diagnostics](#)  
[Log4J](#), [Logback](#), [Serilog](#)



# INSTALLATION & CONFIGURATION



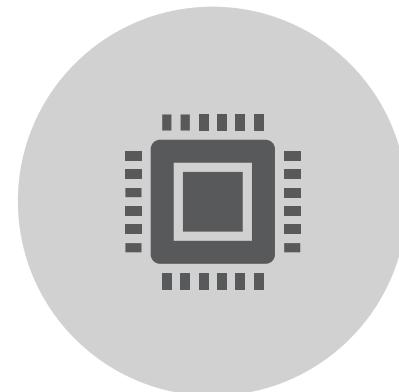
# 3 OPTIONS



BUILD TIME: ADD THE APPLICATION INSIGHTS SDK TO YOUR APP.



RUN TIME: INSTRUMENT YOUR APP ON THE SERVER, USING AZURE MONITOR APPLICATION INSIGHTS AGENT, WITHOUT REBUILDING AND REDEPLOYING THE CODE.



BOTH: BUILD THE SDK INTO YOUR APP, AND ALSO APPLY THE RUN-TIME EXTENSIONS. GET THE BEST OF BOTH OPTIONS.

# RUN TIME – AUTO-INSTRUMENTATION

## Advantages

- No developer investment
- Don't need to update the SDK
- Also works when you don't have access to the source code

## Disadvantages

- Less control

The screenshot shows the Microsoft Azure portal interface for creating a new web application. At the top, there's a navigation bar with the Microsoft Azure logo and a search bar. Below the navigation bar, the breadcrumb trail shows 'Home > Create a resource > Create Web App'. The main title 'Create Web App' is followed by a three-dot ellipsis. Underneath, there are tabs for 'Basics', 'Deployment', 'Networking', 'Monitoring' (which is underlined in blue), 'Tags', and 'Review + create'. The 'Monitoring' section contains a descriptive paragraph about Azure Monitor Application Insights, mentioning it's an Application Performance Management service for developers and DevOps professionals. It explains that enabling it allows automatic monitoring of the application, detecting performance anomalies and providing diagnostic tools. A 'Learn more' link is provided. Below this, there are two radio buttons for 'Enable Application Insights': 'No' (unselected) and 'Yes' (selected). A dropdown menu shows '(New) cloudbreweshopweb (West Europe)' with a 'Create new' option. In the 'Region' field, 'West Europe' is selected.



BUILD TIME  
INTEGRATION

# ASP.NET CORE

## Configure Telemetry

- `services.AddApplicationInsightsTelemetry();`
- Uses “ApplicationInsights” section in appsettings.json by default

## Logging

- By default an ILogger instance is registered
- Default log level= Warning!
- <https://bartwullems.blogspot.com/2022/03/application-insights-configure-log-level.html>
- When using Serilog an Application Insights sink should be configured
- <https://github.com/serilog-contrib/serilog-sinks-applicationinsights>

# .NET CORE WORKER SERVICE

## Configure Telemetry

- services.AddApplicationInsightsTelemetryWorkerService();
- Uses “ApplicationInsights” section in appsettings.json by default

## Logging

- By default an ILogger instance is registered
- Default log level= Warning!
- <https://bartwullems.blogspot.com/2022/03/application-insights-configure-log-level.html>
- When using Serilog an Application Insights sink should be configured
- <https://github.com/serilog-contrib/serilog-sinks-applicationinsights>



**CLIENT SIDE  
TELEMETRY**

# CLIENT SIDE TELEMETRY

## ASP.NET Core MVC, Razor pages

- Add snippet in the head section

```
@inject Microsoft.ApplicationInsights.AspNetCore.JavaScriptSnippet JavaScriptSnippet
```

```
@Html.Raw(JavaScriptSnippet.FullScript)  
</head>
```

## Single Page Applications

- NPM: @microsoft/applicationinsights-web
- Angular: Create a Service and inject this in the main App-Component
- Important to track page changes and correlate requests

```
enableCorsCorrelation: true,  
enableAutoRouteTracking: true
```

# ANGULAR

```
import { Injectable } from '@angular/core';
import { ApplicationInsights } from '@microsoft/applicationinsights-web';
import { environment } from 'src/environments/environment';

@Injectable({
  providedIn: 'root'
})
export class AppInsightsService {

  instance: ApplicationInsights;

  constructor() {
    this.instance = new ApplicationInsights({ config: {
      instrumentationKey: environment.appInsights.instrumentationKey,
      enableCorsCorrelation: true,
      enableAutoRouteTracking: true
    } });
    this.instance.loadAppInsights();
    this.instance.trackPageView();
  }
}
```

# ANGULAR

```
import { Component } from '@angular/core';
import { AuthService } from './auth/auth.service';
import { User } from './auth/user';
import { AppInsightsService } from './app-insights.service';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {

  title = 'Fabrikam Residences';

  get user(): User { return this.auth.getAuthenticatedUser(); }

  constructor(private auth: AuthService, private appInsights: AppInsightsService) { }
}
```

# BLAZOR

## Configure Telemetry

- No official support (yet)
- Use BlazorApplicationInsights Nuget
- <https://github.com/IvanJosipovic/BlazorApplicationInsights>
- Add call to Program.cs

```
builder.Services.AddBlazorApplicationInsights();
```

- Add component to App.razor

```
<ApplicationInsightsComponent />
```

- Add Application Insights JS to head in index.html
- Add JS Interop to the bottom of body in index.html

```
<script src="_content/BlazorApplicationInsights/JsInterop.js"></script>
```

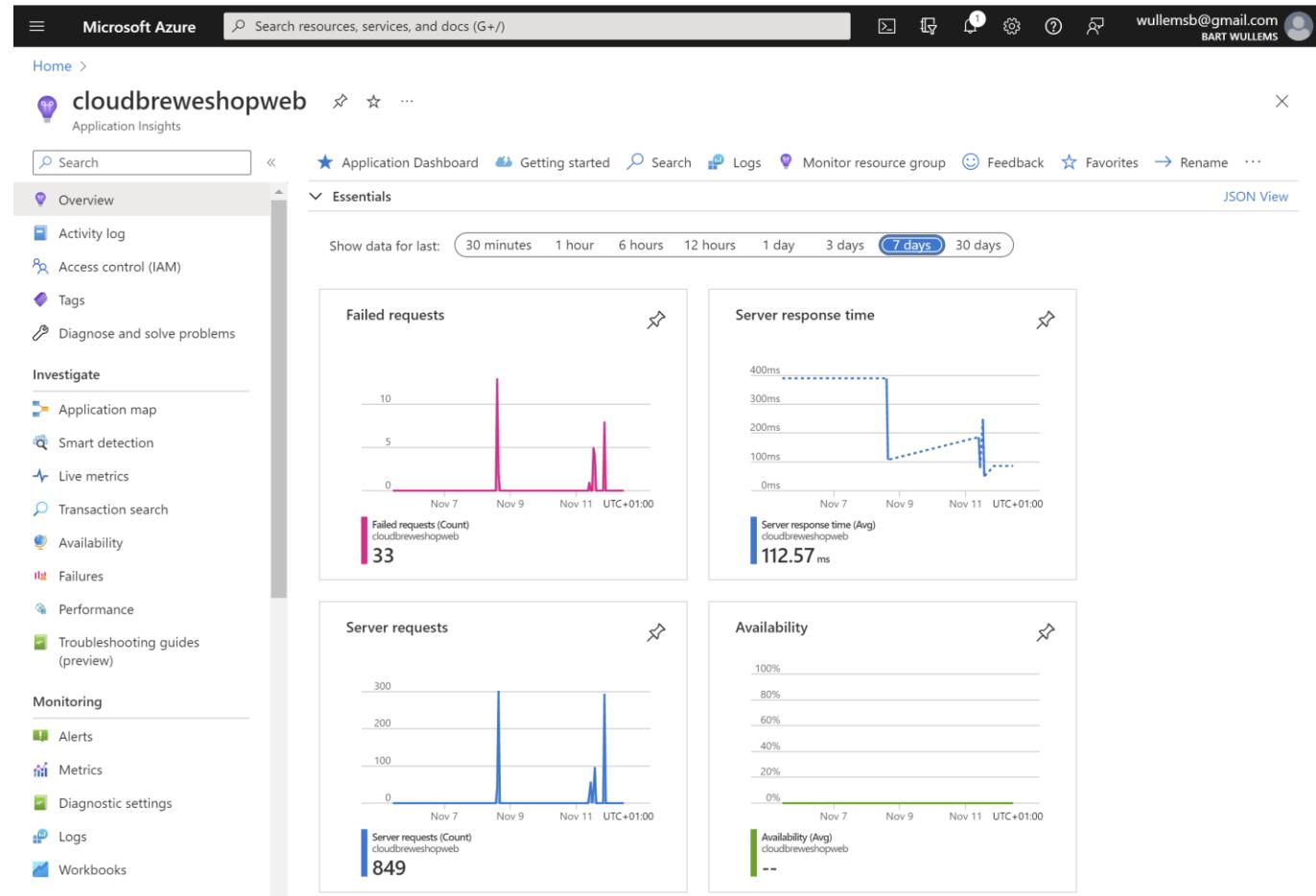


# CONSUMING TELEMETRY DATA



# OVERVIEW

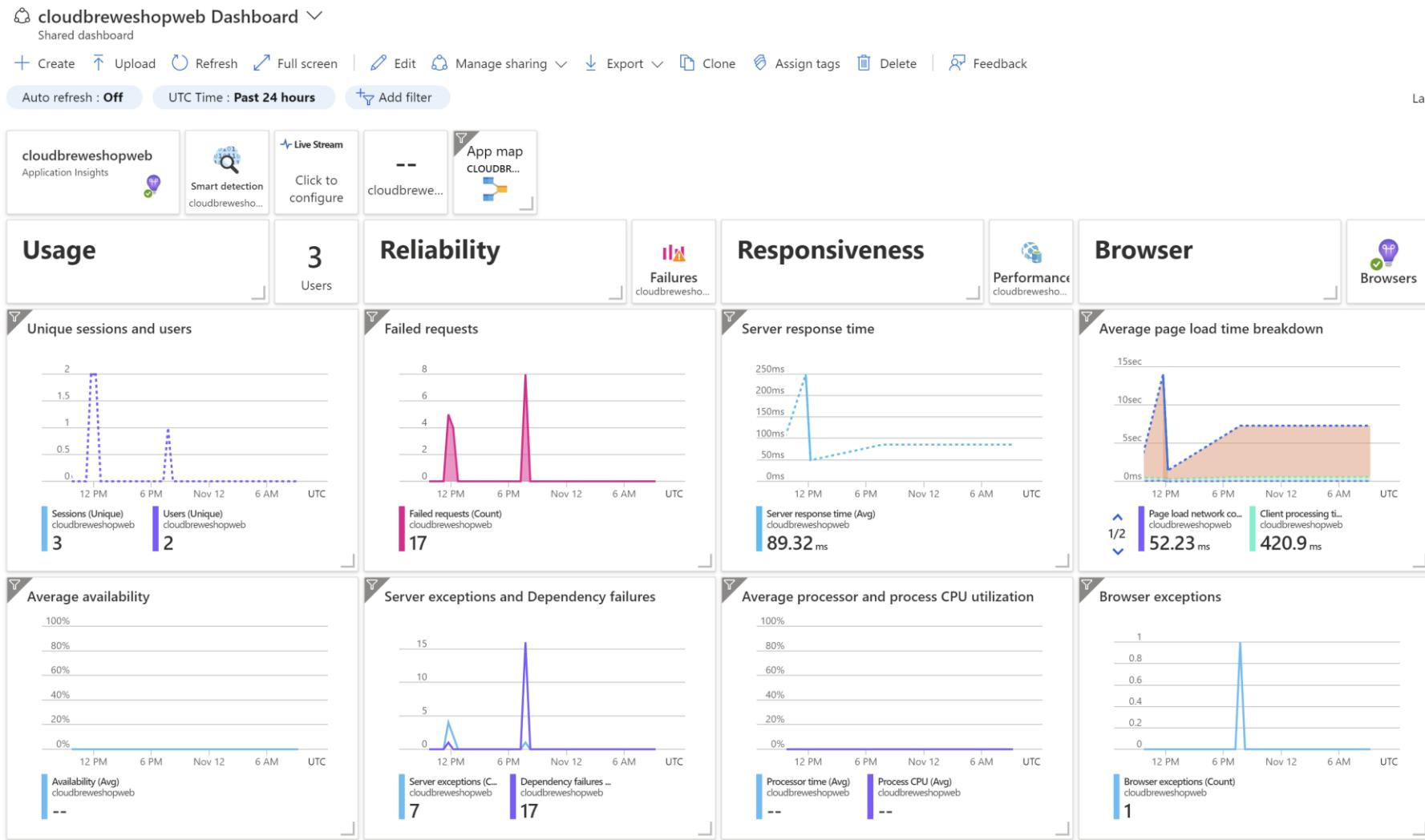
- Provides a summary of the application's health - availability, performance and usage
- Paired timeline allows users to quickly correlate metrics and identify trends
- Drilldown into results





**CONSUMING  
TELEMETRY DATA**

# APPLICATION DASHBOARD



# WORKBOOKS

**Workbooks provide a flexible canvas for data analysis and the creation of rich visual reports within the Azure portal. They allow you to tap into multiple data sources from across Azure, and combine them into unified interactive experiences.**

## Usage through the day

This report helps you understand the usage patterns of your application through the hours of a day. It should help you identify patterns like:

- My app is primarily on weekdays in the evenings between 6pm and 11pm
- On weekends, usage is spread evenly through the day.

Use the parameters below to select the app activities you are interested in, the analysis time window, and the metric you wish to analyze (count of users, sessions, events).

Activities: All Events and Page Views ▾ TimeRange: Last 14 days ▾ Metric: Unique Users ▾ ExcludeWeekends: No ▾

Day	↑↓	12am ↑↓	2am ↑↓	4am ↑↓	6am ↑↓	8am ↑↓	10am ↑↓	12pm ↑↓	2pm ↑↓	4p...↑↓	6pm ↑↓	8pm ↑↓	10pm ↑↓
6/3 - Wed	0	8	52	114	78	90	105	102	51	64	32	32	4
6/4 - Thu	1	13	66	118	125	125	128	118	76	78	34	34	2
6/5 - Fri	0	17	73	134	131	147	128	107	63	56	30	30	5
6/6 - Sat	1	8	48	79	45	63	46	38	26	14	7	7	0
6/7 - Sun	0	1	4	21	24	26	25	31	50	32	17	17	1

# APPLICATION MAP

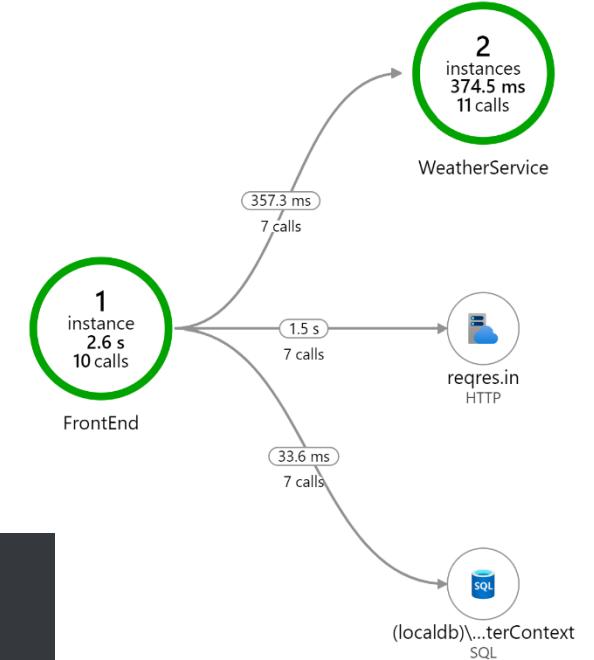


# APPLICATION MAP

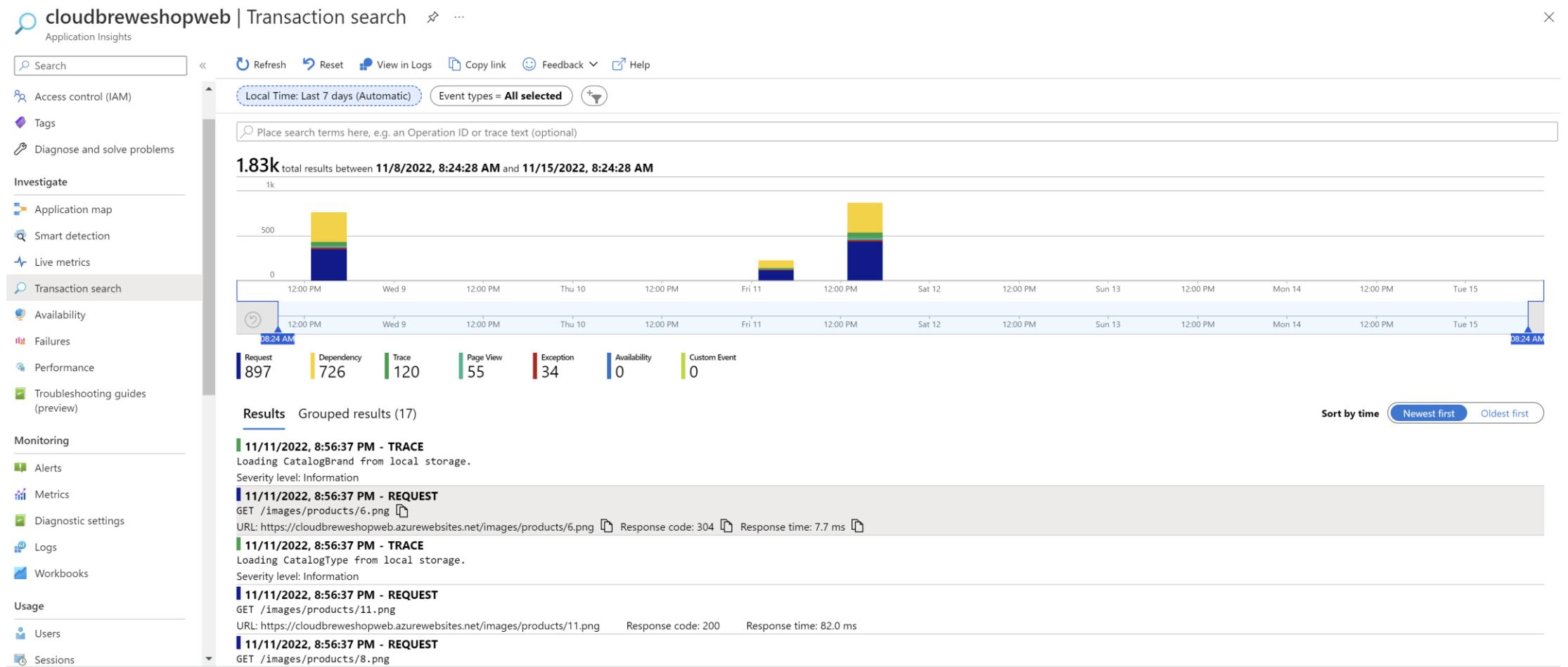
- Use the same Application Insights Resource for your complete solution
- Update the RoleName to better identify the different components

```
public class CloudRoleNameTelemetryInitializer : ITelemetryInitializer
{
    public void Initialize(ITelemetry telemetry)
    {
        // set custom role name here
        telemetry.Context.Cloud.RoleName = "Custom RoleName";
    }
}
```

```
services.AddSingleton<ITelemetryInitializer, CloudRoleNameTelemetryInitializer>();
```



# TRANSACTION SEARCH



# LOGS

- Learn KQL:
  - Kusto Query Language is a powerful tool to explore your data and discover patterns, identify anomalies and outliers, create statistical modeling, and more.
  - It uses schema entities that are organized in a hierarchy similar to SQL's: databases, tables, and columns.

The screenshot shows the Kusto Query Editor interface. At the top, there is a toolbar with buttons for 'Run' (highlighted in blue), 'Save', 'Share', 'New alert rule', 'Export', 'Pin to', and 'Format query'. Below the toolbar, the time range is set to 'Last 7 days'. The query editor contains the following KQL code:

```
1 traces
2 | where message has "Send mail successfully"
3
4
```

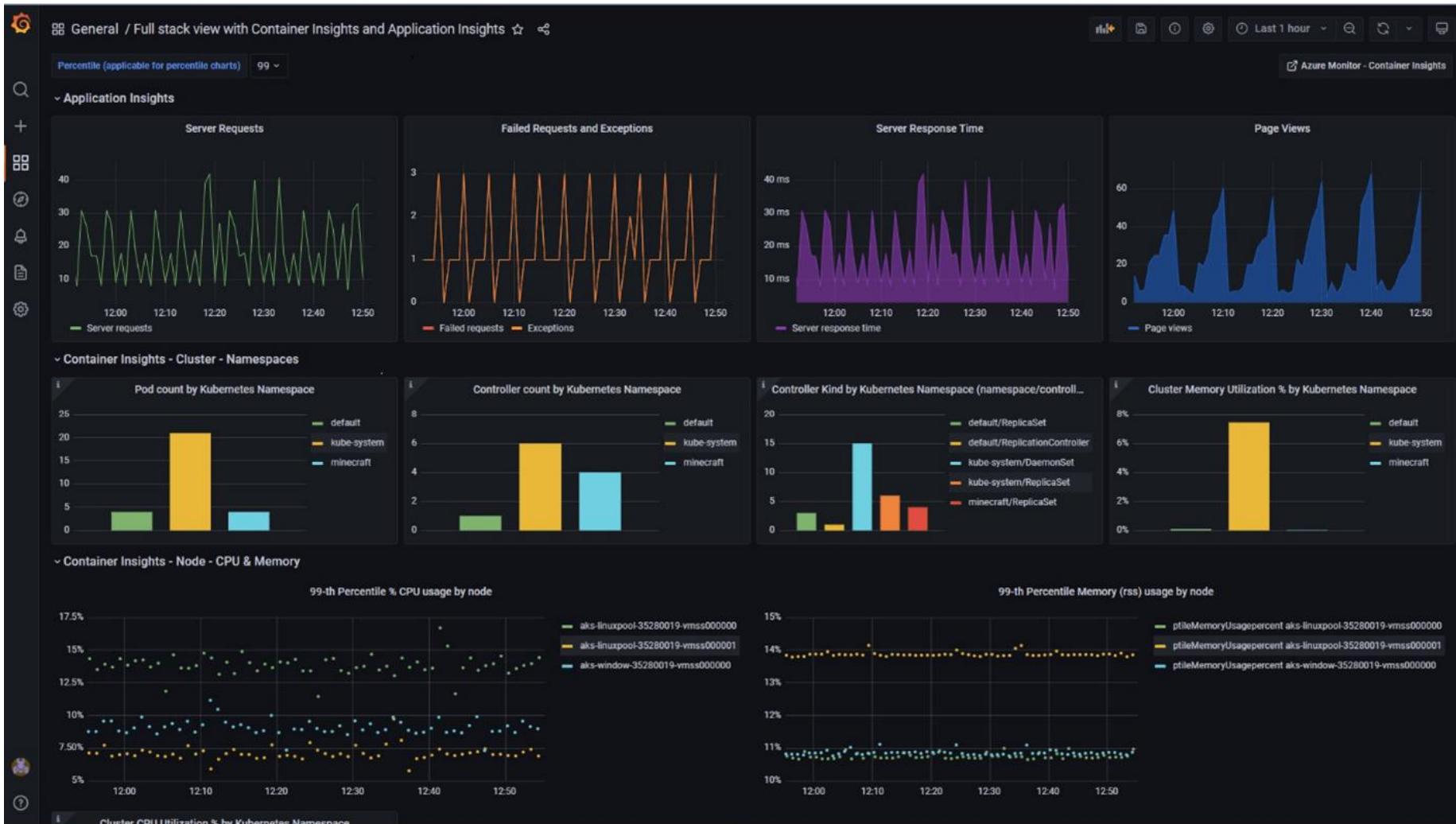
Below the query editor, the results section displays the following information:

- Results tab is selected.
- Display time (UTC+00:00) is set to 00:07.2.
- 21.669 records were found.

The results table has the following columns:

timestamp [UTC]	message	severityLevel	itemType	customDimensions	custom
25/3/2022 10:00:16.737	Send mail successfully to 'Berichtenbox.Test@vlm.be' with subj...	1	trace	{"AspNetCoreEnvironment": "Productie", "CategoryName": "VLM..."}	
25/3/2022 10:00:18.059	Send mail successfully to 'Berichtenbox.Test@vlm.be' with subj...	1	trace	{"AspNetCoreEnvironment": "Productie", "CategoryName": "VLM..."}	
25/3/2022 10:00:23.604	Send mail successfully to 'Berichtenbox.Test@vlm.be' with subj...	1	trace	{"AspNetCoreEnvironment": "Productie", "CategoryName": "VLM..."}	
25/3/2022 10:00:27.817	Send mail successfully to 'Berichtenbox.Test@vlm.be' with subj...	1	trace	{"AspNetCoreEnvironment": "Productie", "CategoryName": "VLM..."}	
25/3/2022 10:00:30.558	Send mail successfully to 'Berichtenbox.Test@vlm.be' with subj...	1	trace	{"AspNetCoreEnvironment": "Productie", "CategoryName": "VLM..."}	

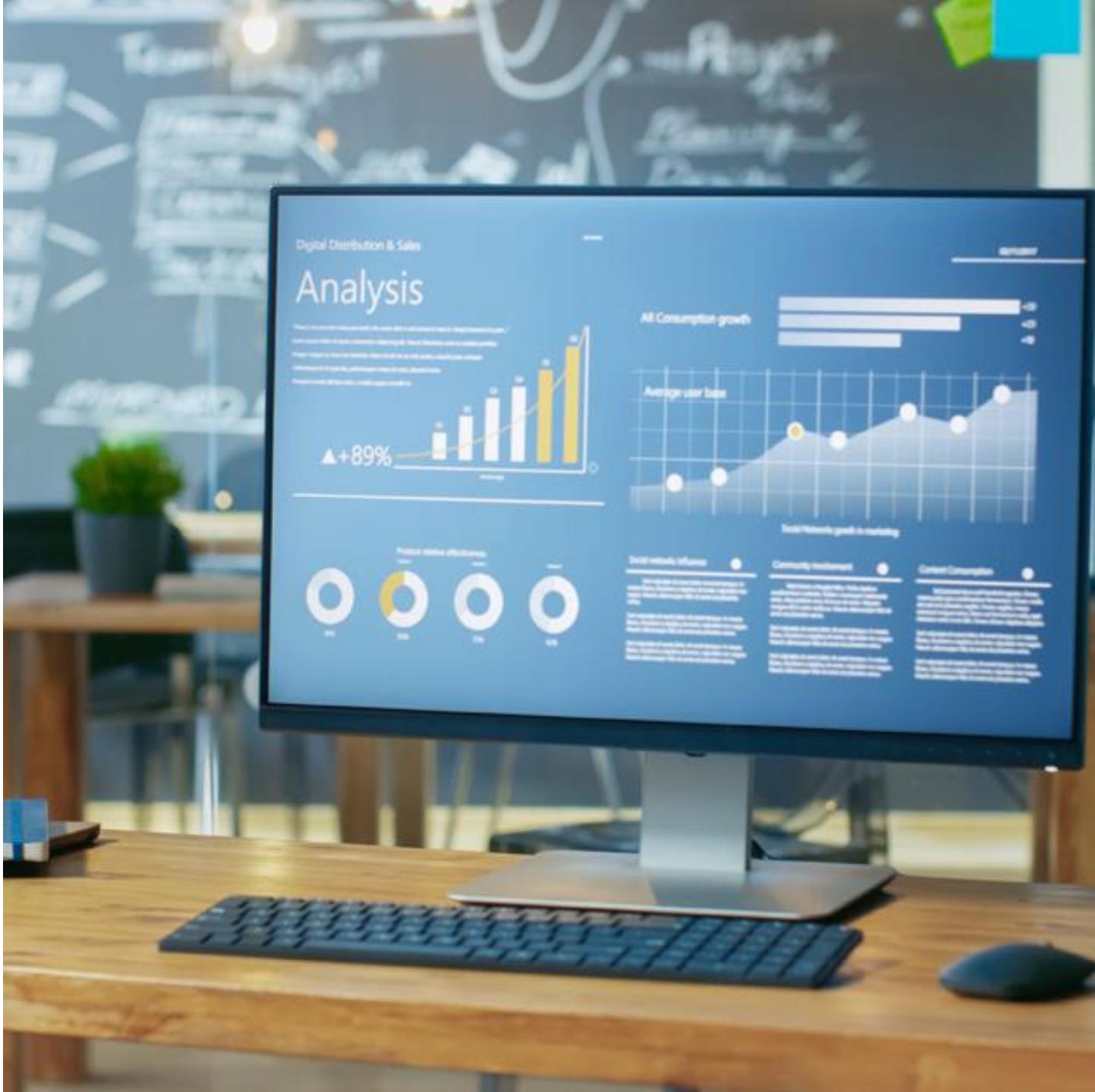
# AZURE MANAGED GRAFANA (PREVIEW)





AZURE MANAGED  
GRAFANA

# PERFORMANCE COUNTERS



# PERFORMANCE COUNTERS

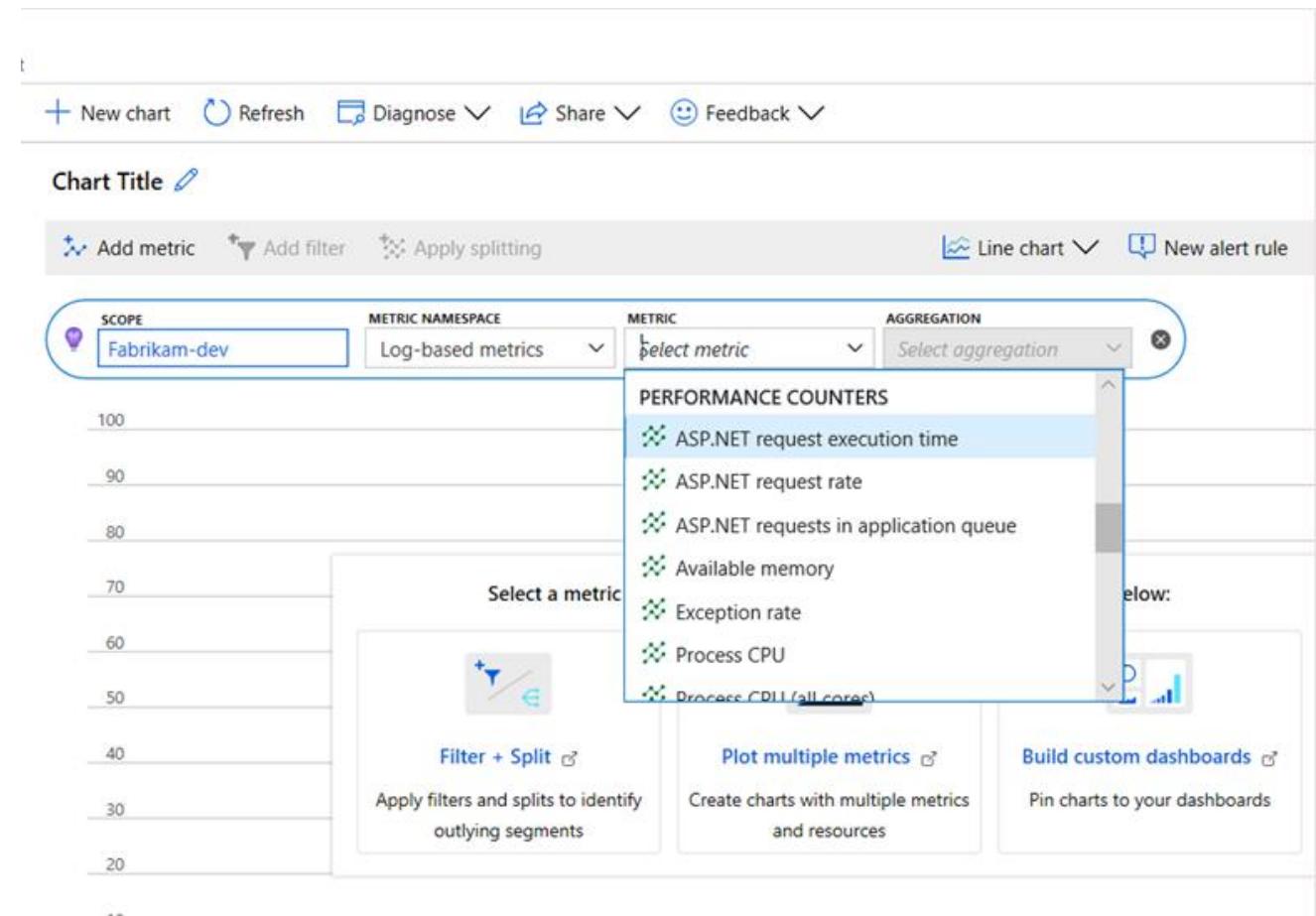
Set of default counters are collected by default

- Memory
- CPU
- # of requests
- ...

Replaced by *EventCounters* to enable x-platform compatibility

- Disabled by default

Tip: If running on-premise; add AppPool user to the correct groups!





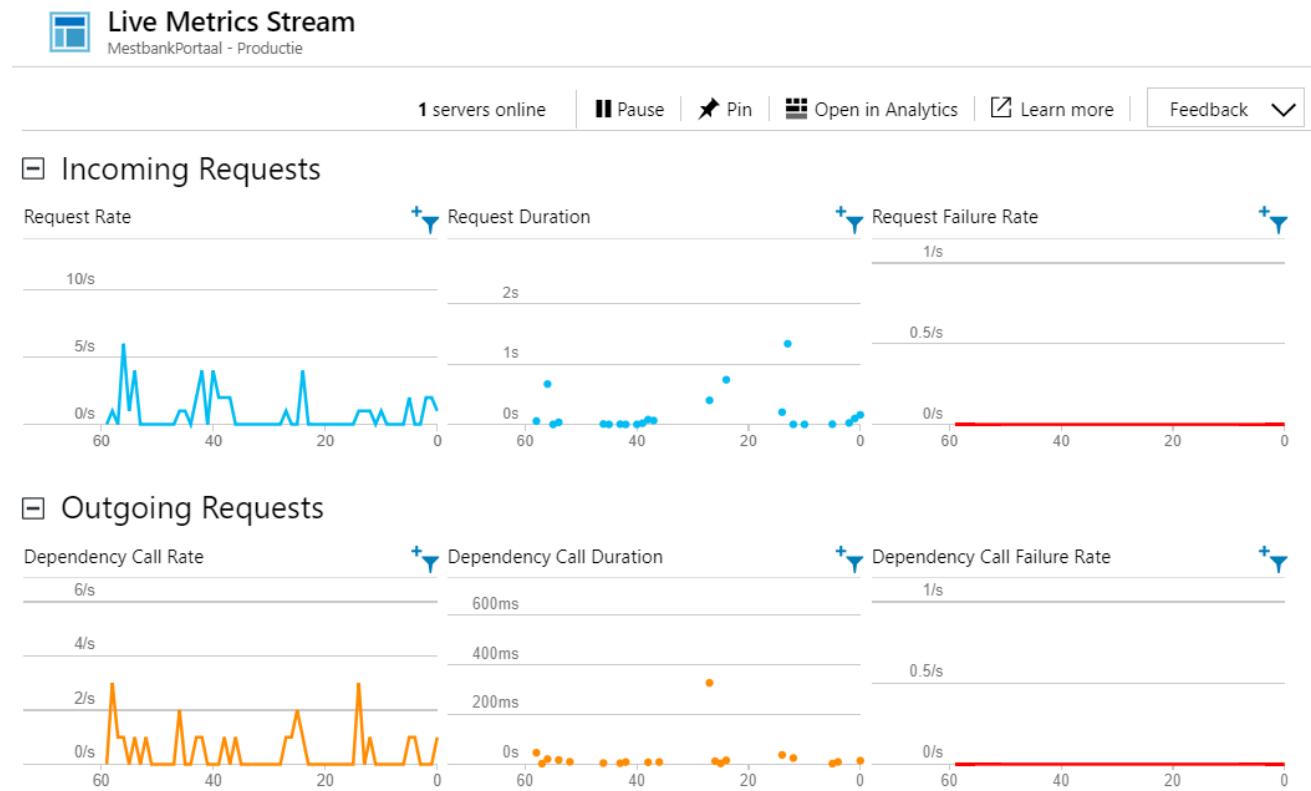
**PERFORMANCE  
COUNTERS**

# LIVE METRICS



# LIVE METRICS

- Enabled by default
- Watch the effect of test loads, and diagnose issues live
- Get exception traces as they happen
- Experiment with filters to find the most relevant KPIs
- Monitor any Windows performance counter live

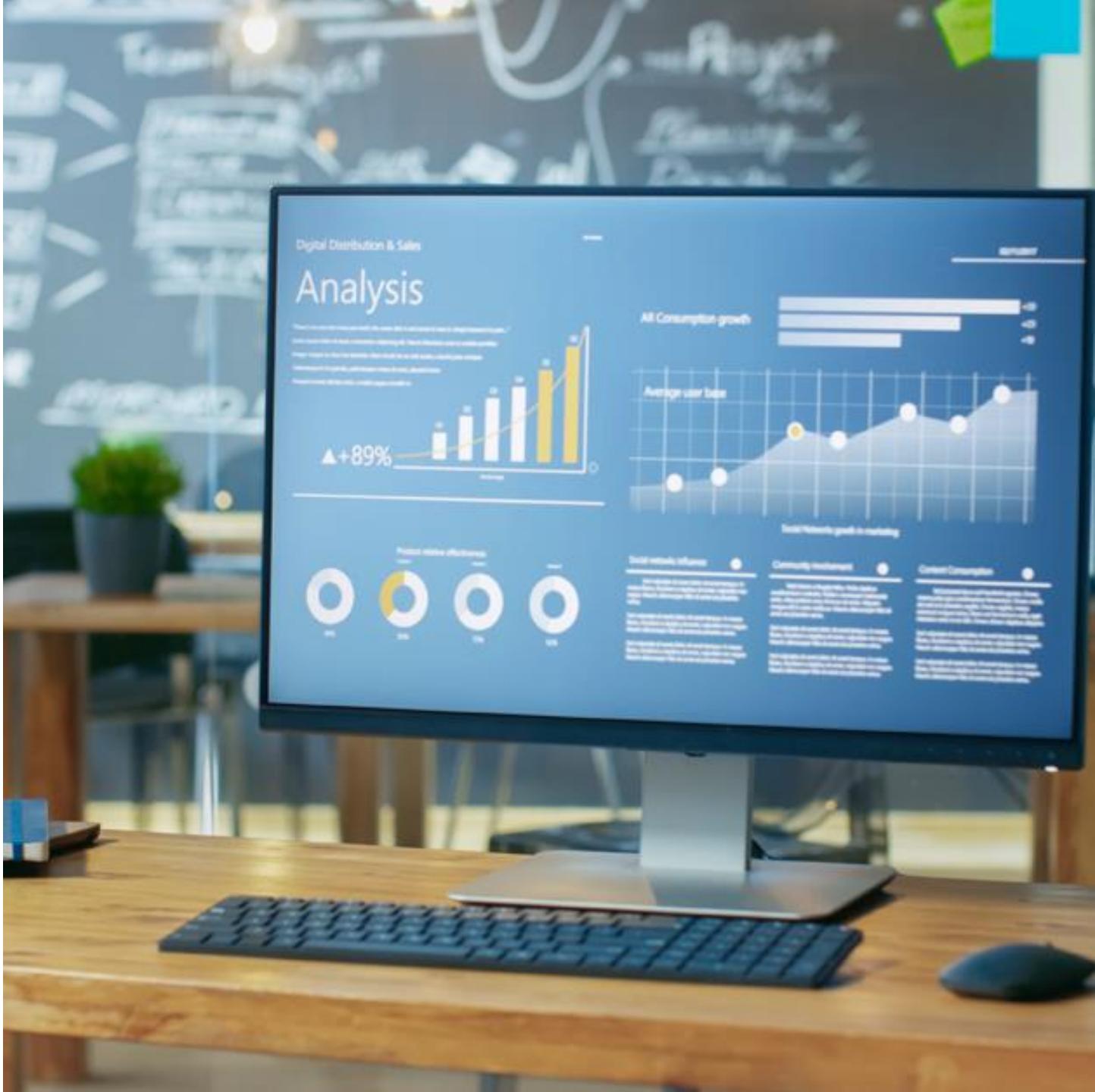




**DEMO**

**LIVE METRICS**

# STRUCTURED LOGGING



# STRUCTURED LOGGING

```
{  
  "dt": "2020-04-07T12:23:34Z",  
  "level": "info",  
  "message": "Completed 200 OK in 79ms (Views: 78.8ms | ActiveRecord: 0.0ms)",  
  "context": {  
    "host": "34.225.155.83",  
    "user_id": 5,  
    "path": "/welcome",  
    "method": "GET"  
  },  
  "http_response_sent": {  
    "status": 200,  
    "duration_ms": 79.0,  
    "view_ms": 78.8,  
    "active_record_ms": 0.0  
  }  
}
```



**STRUCTURED  
LOGGING**

# STRUCTURE

- Supported by most loggers
  - Serilog, Nlog, Log4Net
- Also works with standard .NET objects

```
_logger.LogInformation(message: "Fetching catalog items - Brand: {ProductBrand} - Type: {ProductType}", params args: brand, type);
```

TRACE  
Warning

## Custom Properties

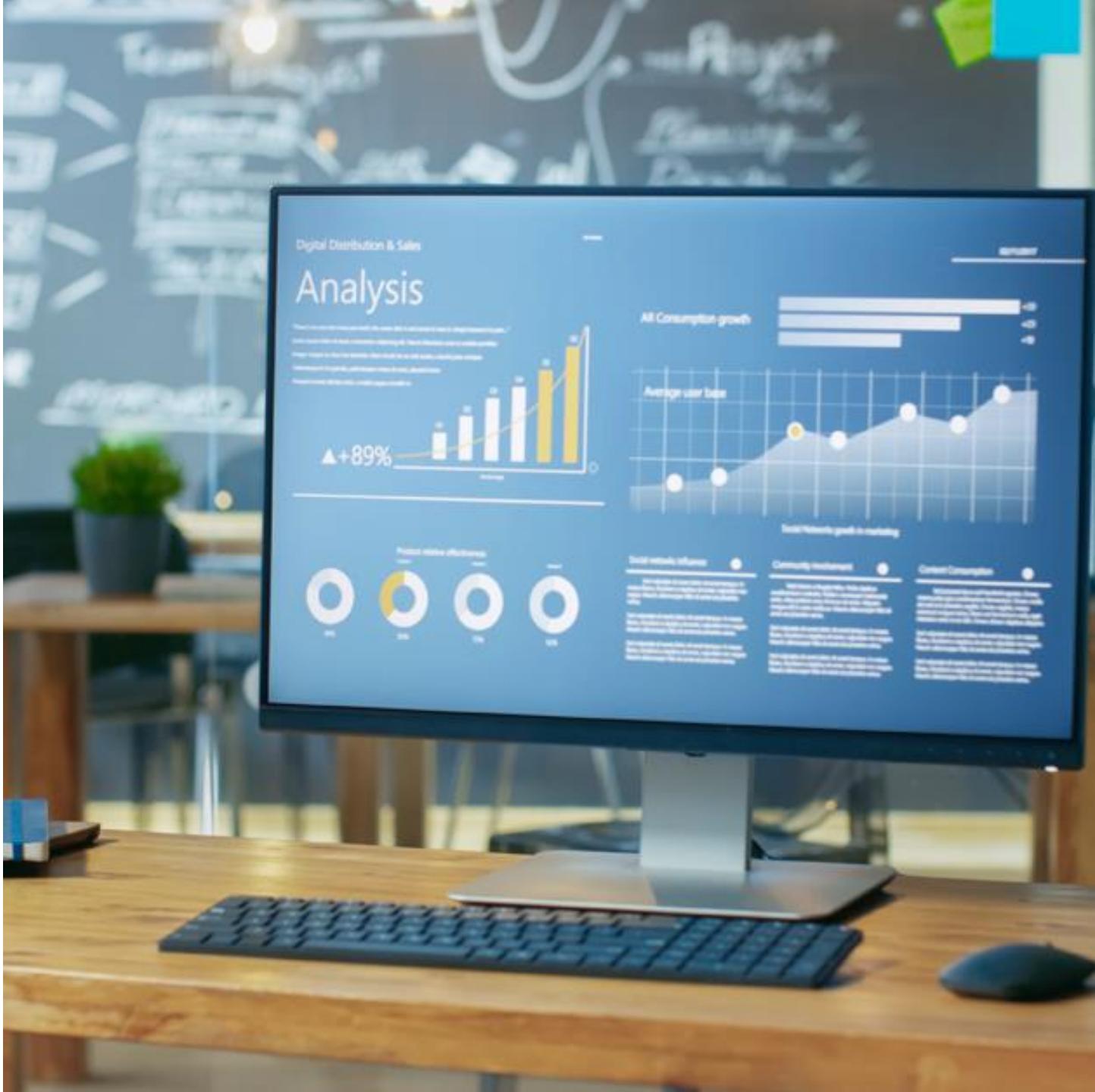
SpanId	6b7bfc088498d924	...
ParentId	0000000000000000	...
{OriginalFormat}	Fetching catalog items - Brand: {ProductBrand} - Type: {ProductType}	...
RequestPath	/	...
ActionName	/Index	...
ProductType	All	...
TraceId	606055cfcb1cfa8e61247e363353e8b6	...
ProductBrand	All	...
CategoryName	Microsoft.eShopWeb.Web.Pages.IndexModel	...
ActionId	d0b325fd-900c-406d-bf11-36b3eaa07b99	...
RequestId	80000bdf-0000-d100-b63f-84710c7967bb	...
AspNetCoreEnvironment	Production	...

pache.org  
ces™

g

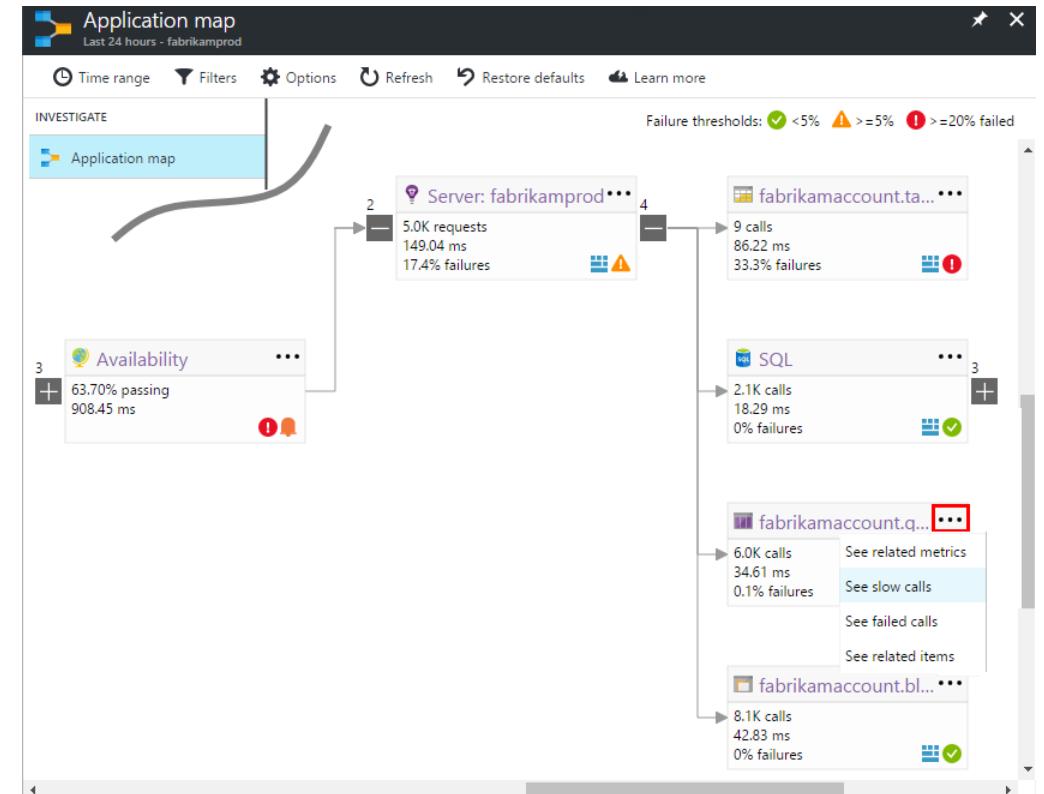


# DEPENDENCY TRACKING



# DEPENDENCY TRACKING

- Dependencies are calls to external systems and services
- Some dependencies are tracked automatically
- We can also monitor dependencies performance
- Tip: Set the cloud role name correct for every component



# DEPENDENCY TRACKING

- Following dependencies are tracked automatically

Dependencies	Details
Http/Https	Local or Remote http/https calls
WCF calls	Only tracked automatically if Http-based bindings are used.
SQL	Calls made with <code>SqlClient</code> . See <a href="#">this</a> for capturing SQL query.
Azure storage (Blob, Table, Queue) ↗	Calls made with Azure Storage Client.
EventHub Client SDK ↗	Version 1.1.0 and above.
ServiceBus Client SDK ↗	Version 3.0.0 and above.
Azure Cosmos DB	Only tracked automatically if HTTP/HTTPS is used. TCP mode won't be captured by Application Insights.

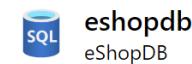


**DEPENDENCY  
TRACKING**

# SQL SERVER

- SQL Command logging is disabled by default
- Enable through:

```
services.ConfigureTelemetryModule<DependencyTrackingTelemetryModule>((module, o)
    => {
    module.EnableSqlCommandTextInstrumentation = true; })
```



## Dependency Properties

[Show all](#)

Event time	11/15/2022, 1:02:16.101 PM (Local time)	...
Type	SQL	...
Call status	true	...
Duration	36.0 ms	...
Name	SQL: tcp:eshopdb.database.windows.net,1433   eShopDB	...

## Custom Properties

AspNetCoreEnvironment	Production	...
-----------------------	------------	-----

## Command

[Copy](#)

```
SELECT TOP(@__p_0) [c].[Id], [c].[CatalogBrandId], [c].[CatalogTypeId],
[c].[Description], [c].[Name], [c].[PictureUri], [c].[Price]
FROM [Catalog] AS [c]
```

## Related Items

	Show what happened before and after this dependency in User Flows	<a href="#">i</a>
	All available telemetry 5 minutes before and after this event	<a href="#">i</a>

# MASSTRANSIT

- You can add other dependencies (e.g. MassTransit)

```
services.ConfigureTelemetryModule<DependencyTrackingTelemetryModule>((module, o)
=> {
    module.EnableSqlCommandTextInstrumentation = true;
    module.IncludeDiagnosticSourceActivities.Add("MassTransit");
});
```

/DEVELOPMENT:VLM.eShopExample.Application.Messages:...  
MassTransit.Transport.Send

Dependency Properties		<a href="#">Show all</a>
Event time	18/5/2022 11:42:00.417 (Local time)	...
Type	Send	...
Call status	true	...
Duration	103 ms	...
Name	MassTransit.Transport.Send	...

# ENHANCE TELEMETRY



# ENHANCE TELEMETRY

## Telemetry initializers

- Sets context properties that are sent along with every item of telemetry.
- Lots of built-in initializers.
- Extra initializers exist; e.g. Kubernetes
- Easy to write your own e.g. CloudRoleNameInitializer.

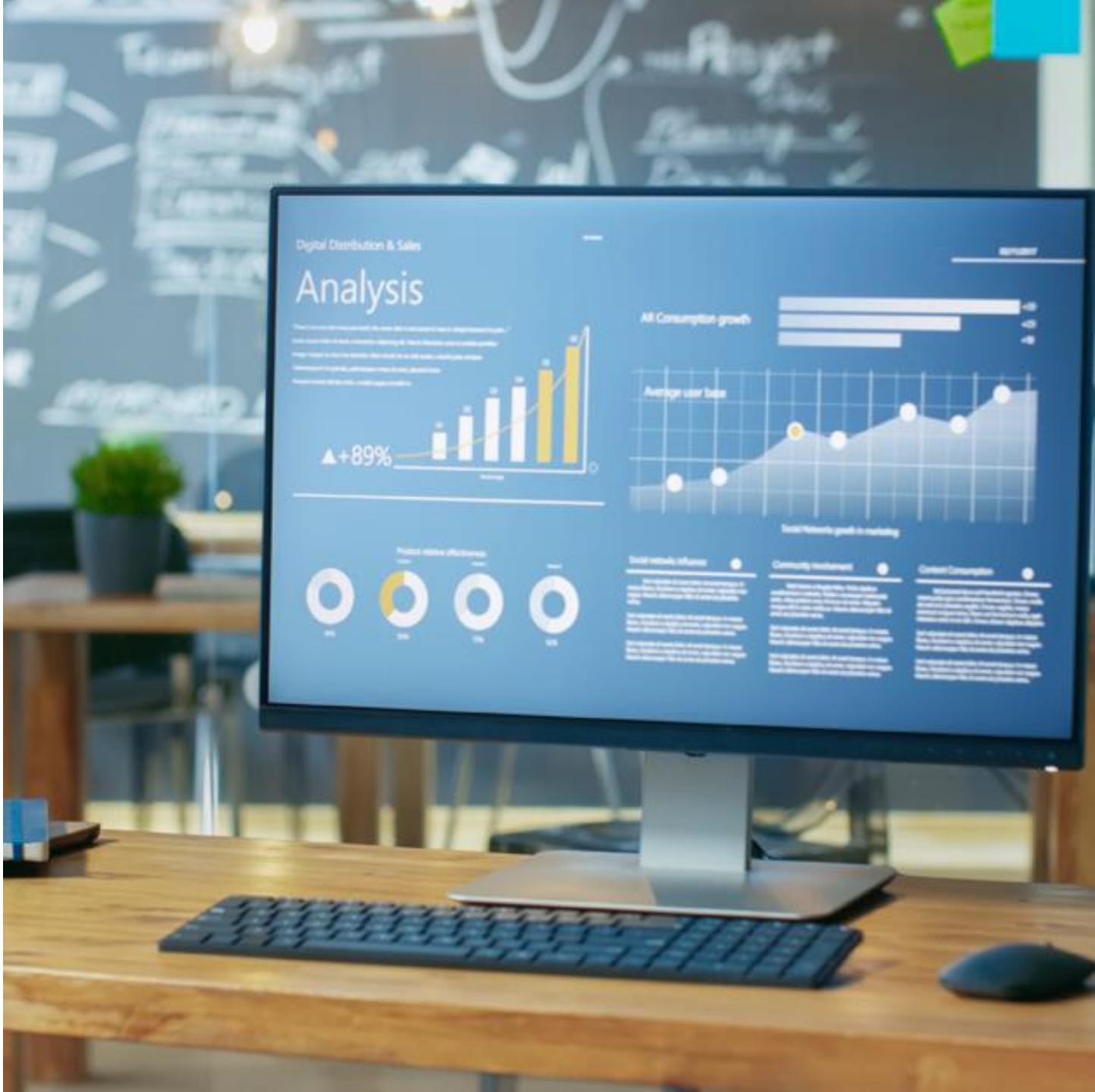
## Telemetry modules

- Collects a specific type of data and uses the core API to send it.
- Lots of built-in modules
- 3rd party modules exist; e.g. NServiceBus

## Telemetry processors

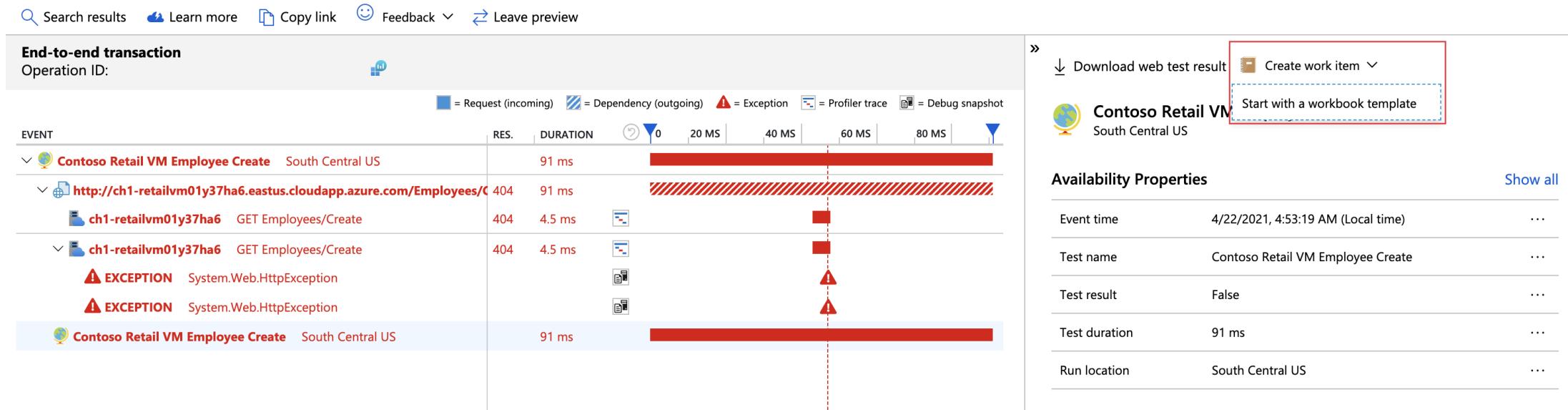
- Can filter and modify each telemetry item before it's sent to the portal.
- E.g. filter out all 400 responses

# AZURE DEVOPS/ GITHUB INTEGRATION



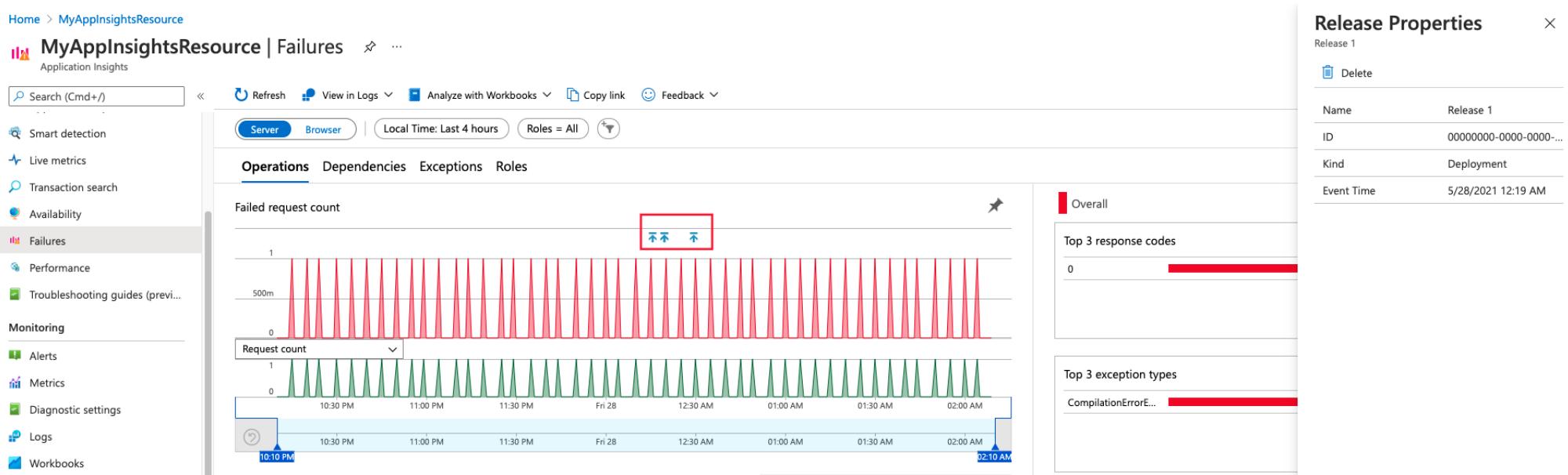
# WORK ITEM INTEGRATION

- Allows you to easily create work items in GitHub or Azure DevOps that have relevant Application Insights data embedded in them.
- Works for both Azure DevOps and Github

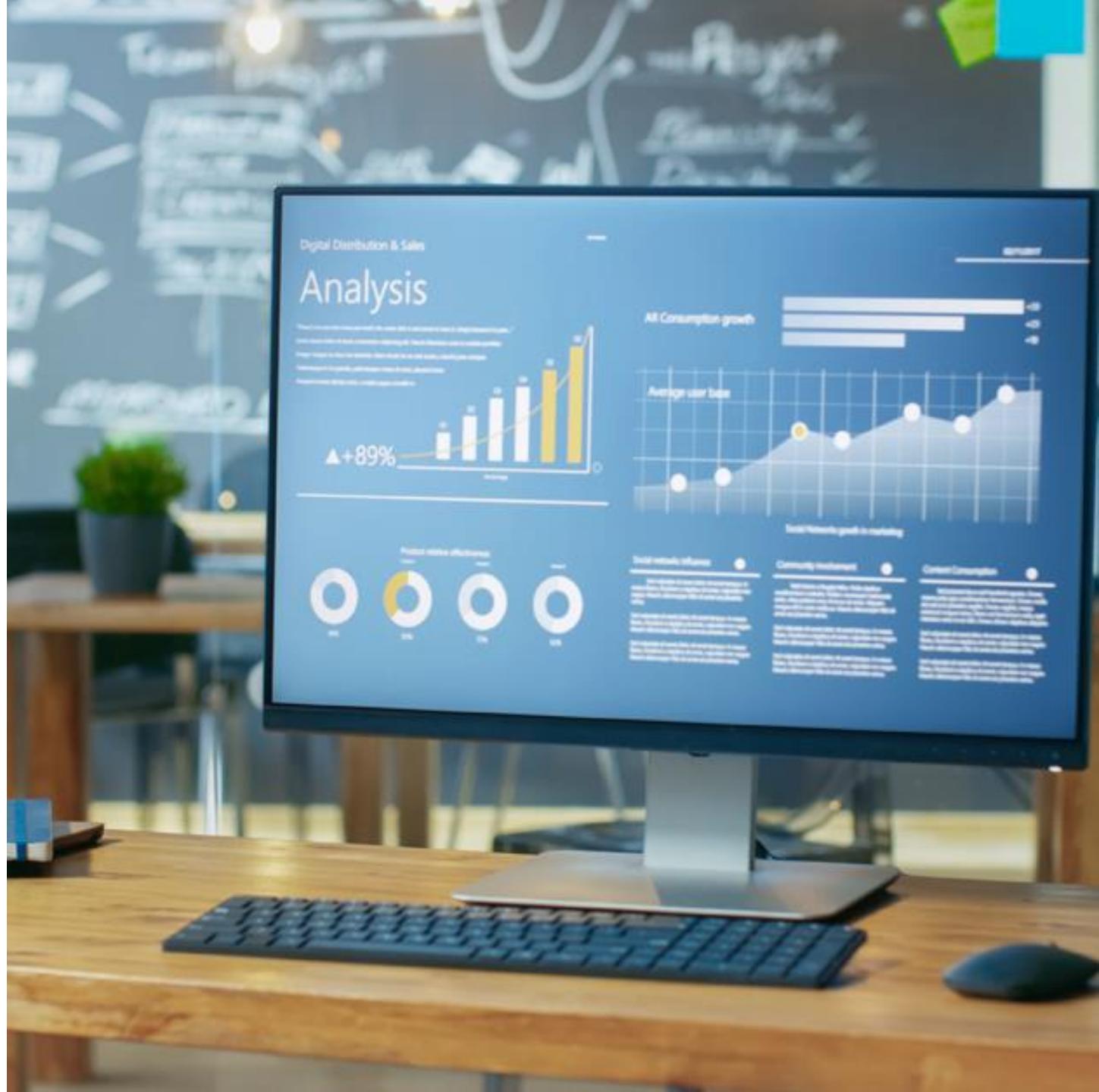


# RELEASE ANNOTATIONS

- Annotations show where you deployed a new build, or other significant events.
- Annotations make it easy to see whether your changes had any effect on your application's performance.
- Use the ReleaseAnnotation.ps1 powershell script in your release pipeline



# SNAPSHOT DEBUGGER



# SNAPSHOT DEBUGGER

- Collect a debug snapshot from your live application when an exception occurs.
- The snapshot shows the state of source code and variables at the moment the exception was thrown.
- Tricky to get up and running 😊
- Only on Windows Service Plan(Azure)

The screenshot displays the 'Debug Snapshot' interface. At the top right, there is a purple button labeled 'Download Snapshot' with the subtext 'Opens in Visual Studio Enterprise'. The main area is divided into two sections: 'Call stack' and 'Locals'.

**Call stack:**

```
[Managed to Native Transition]
MyCompany.Visitors.Web.Controllers.VisitorsController.Get(int visitorId, MyCompany.Visitors.Model.PictureType pictureType)
System.Runtime.CompilerServices.AsyncMethodBuilderCore.MoveNextRunner.InvokeMoveNext(object stateMachine)
System.Threading.ExecutionContext.RunInternal(System.Threading.ExecutionContext executionContext, System.Threading.ContextCallback callback, object state, bool pres...
System.Threading.ExecutionContext.Run(System.Threading.ExecutionContext executionContext, System.Threading.ContextCallback callback, object state, bool preserveSyn...
System.Runtime.CompilerServices.AsyncMethodBuilderCore.MoveNextRunner.Run()
System.Threading.Tasks.AwaitTaskContinuation.InvokeAction(object state)
System.Threading.Tasks.AwaitTaskContinuation.RunCallback(System.Threading.ContextCallback callback, object state, ref System.Threading.Tasks.Task currentTask)
System.Threading.Tasks.SynchronizationContextAwaitTaskContinuation.Run(System.Threading.Tasks.Task task, bool canInlineContinuationTask)
System.Threading.Tasks.Task.FinishContinuations()
System.Threading.Tasks.Task.FinishStageThree()
System.Threading.Tasks.Task<System._Canon>.TrySetResult(System._Canon result)
System.Runtime.CompilerServices.AsyncTaskMethodBuilder<MyCompany.Visitors.Model.Visitor>.SetResult(MyCompany.Visitors.Model.Visitor result)
```

**Locals:**

NAME	VALUE	TYPE
exception	{System.IndexOutOfRangeException: Index was outside the bound...}	System.IndexOutOfRangeException
this	{MyCompany.Visitors.Web.Controllers.VisitorsController}	MyCompany.Visitors.Web.Controllers.VisitorsController
visitorId	3	int
[0]	1	sbyte

# ENABLE SNAPSHOT DEBUGGER

## Enable Application Insights Snapshot Debugger for your application

Snapshot collection is available for:

- .NET Framework and ASP.NET applications running .NET Framework [LTS](#) or later.
- .NET Core and ASP.NET Core applications running .NET Core [LTS](#) on Windows.
- .NET [LTS](#) applications on Windows.

We don't recommend using .NET Core versions prior to LTS since they're out of support.

The following environments are supported:

- [Azure App Service](#)
- [Azure Function](#)
- [Azure Cloud Services](#) running OS family 4 or later
- [Azure Service Fabric services](#) running on Windows Server 2012 R2 or later
- [Azure Virtual Machines](#) and [virtual machine scale sets](#) running Windows Server 2012 R2 or later
- [On-premises virtual or physical machines](#) running Windows Server 2012 R2 or later or Windows 8.1 or later

 Note

Client applications (for example, WPF, Windows Forms or UWP) aren't supported.

If you've enabled Snapshot Debugger but aren't seeing snapshots, check our [Troubleshooting guide](#).



**SNAPSHOT  
DEBUGGER**

# SNAPSHOT DEBUGGER

Debug Snapshot ...

[Help](#) [Send the team an email](#)

Snapshots «

Selected  
11/16/2022, 1:26:05 PM  
POST /Account/Login  
Duration 150.4 ms Response code: 500

Date 1 Group by

All  
11/16/2022, 1:26:05 PM  
POST /Account/Login  
Duration 150.4 ms Response code: 500

Call Stack

METHOD

[Managed to Native Transition]  
void Microsoft.Data.ProviderBase.DbConnectionPool.CheckPoolBlockingPeriod(System.Exception e)  
Microsoft.Data.ProviderBase.DbConnectionInternal Microsoft.Data.ProviderBase.DbConnectionPool.CreateObject(System.Data.Common...  
Microsoft.Data.ProviderBase.DbConnectionInternal Microsoft.Data.ProviderBase.DbConnectionPool.UserCreateRequest(System.Data.Com...  
bool Microsoft.Data.ProviderBase.DbConnectionPool.TryGetConnection(System.Data.Common.DbConnection owningObject, uint waitFor...  
void Microsoft.Data.ProviderBase.DbConnectionPool.WaitForPendingOpen()  
void System.Threading.Thread.StartHelper.Callback(object state)  
void System.Threading.ExecutionContext.RunInternal(System.Threading.ExecutionContext executionContext, System.Threading.ContextCa...

Locals

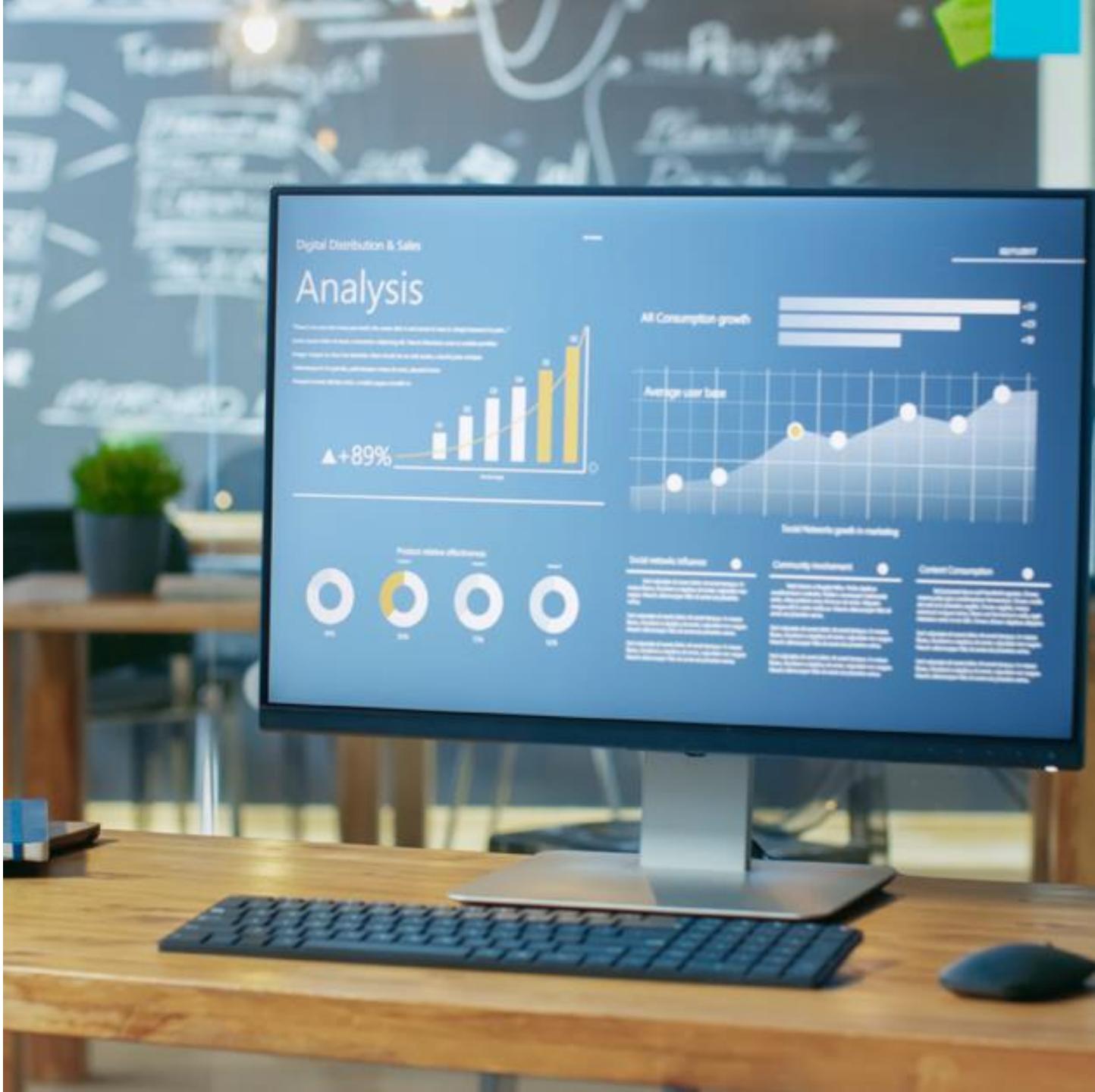
NAME	VALUE	TYPE
<b>Function Arguments</b>		
► e	{Microsoft.Data.SqlClient.SqlException}	System.Exception {Microsoft.Data.SqlClient.SqlExcep...
<b>Variables</b>		
► \$exception	{Microsoft.Data.SqlClient.SqlException}	Microsoft.Data.SqlClient.SqlException
► this	{Microsoft.Data.ProviderBase.DbConnectio...	Microsoft.Data.ProviderBase.DbConnectio...

Decompiled Code (Preview)

Not available

[Download Snapshot](#)  
Opens in Visual Studio Enterprise

# PERFORMANCE PROFILING



# PERFORMANCE PROFILING

Through Application Insights Profiler

## Capture and view performance traces

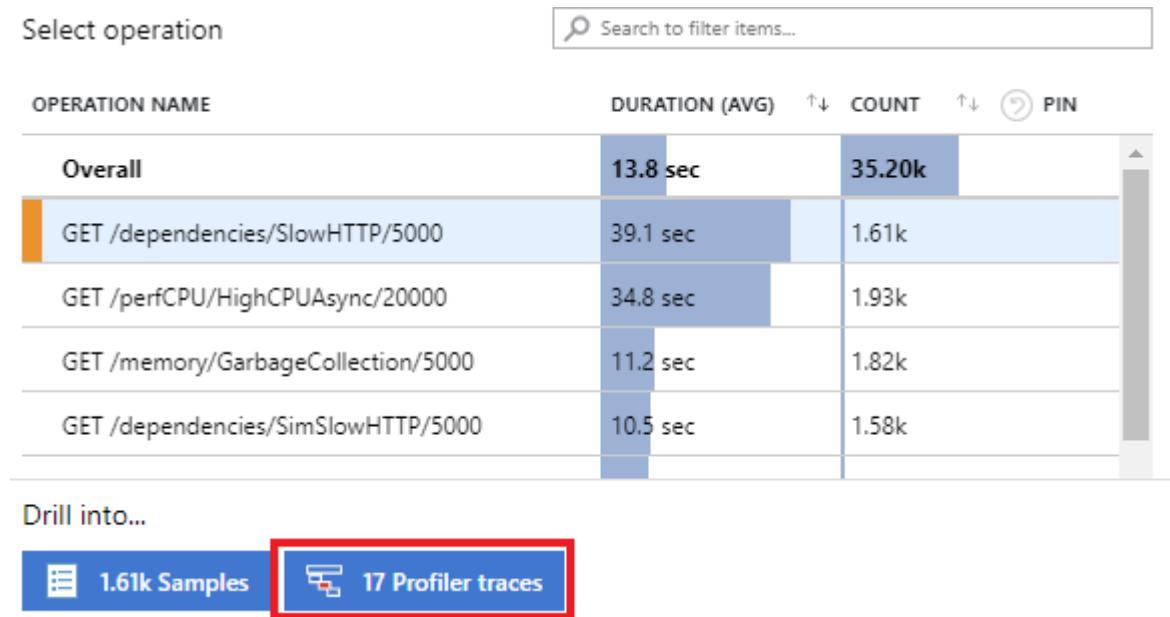
- Captures the following information:
- The median, fastest, and slowest response times
- The “hot” code path spending the most time handling a particular web request.

## Trigger based

- Sampling trigger: random profiling once an hour for 2min
- CPU trigger: starts profiling when CPU usage is above 80 %
- Memory trigger: starts profiling when memory usage is above 80 %

## Minimal overhead

- 5-15% CPU overhead





**PERFORMANCE  
PROFILING**

# PERFORMANCE PROFILING

Search results Learn more Copy link Feedback Leave preview

**End-to-end transaction**  
Operation ID: 40b94f298a8fb9873ace6a78a0bd1b73

EVENT RES. DURATION 0 2 MS 4 MS 6 MS 8 MS

cloudbreweshopweb GET / 307 8.5 ms

= Request (incoming) = Profiler trace

Open profiler traces Create work item

cloudbreweshopweb GET /

**Request Properties** Show all

Event time	11/16/2022, 1:49:31.018 PM (Local time)	...
Name	GET /	...
Response code	307	...
Successful request	true	...
Response time	8.5 ms	...
URL	http://cloudbreweshopweb.azurewebsites.net/	...

**Custom Properties**

AspNetCoreEnvironment	Production	...
-----------------------	------------	-----

**Related Items**

- Show what happened before and after this request in User Flows
- Show trend of this request over time
- All available telemetry 5 minutes before and after this event

1 All 0 Traces 0 Events

View all telemetry ^

# PERFORMANCE PROFILING

Home > cloudbreweshopweb | Transaction search > End-to-end transaction details >

GET / ...  
11/15/2022 1:49 PM to 11/17/2022 1:49 PM - cloudbreweshopweb

Help Send Feedback Configure Profiler

## Examples

Slowest wall clock time

503.25 ms	01:14:29 PM, 2022-11-16
69.04 ms	10:33:42 PM, 2022-11-15
49.59 ms	01:43:54 AM, 2022-11-16
42.39 ms	12:44:28 PM, 2022-11-16
22.61 ms	08:49:15 AM, 2022-11-16
17.26 ms	09:38:39 PM, 2022-11-15
16.05 ms	11:53:47 PM, 2022-11-15
12.67 ms	10:53:43 PM, 2022-11-15
9.48 ms	01:34:30 PM, 2022-11-16
3.28 ms	12:48:51 AM, 2022-11-16
1.83 ms	11:03:45 PM, 2022-11-15

Profiler now has CPU and Memory Triggers. [Click here to configure your trigger settings.](#) Dismiss.

Machine: rd00155d586e0dcloudbreweshopweb Process ID: 7152 Activity ID: #/7152/1/390/1/

Was this trace helpful?

[Download Trace](#)

Profile tree Flame graph

performance tip:  
73.61% of this request was spent in CPU time.  
CPU is executing instructions.  
[Show me](#)

Hot path  Framework dependencies

HIDE EVENTS ⓘ

Type in string to hide from the call tree and press ENTER

Add events Suggested events

No hidden events.

EVENTS	MODULE	THREAD TIME ↑↓	TIMELINE ↑↓
Request()		9.25 ms	
ActivityJsonStart(#/7152/1/390/1/1/)		9.25 ms	
Framework/Library _RtlUserThreadStart	ntdll	9.25 ms	
aspnetcorev2_inprocess!?	aspnetcorev2_inprocess	6.81 ms	
OTHER <<iiscore!?>>	iiscore	6.81 ms	
CPU Time		6.81 ms	
Waiting		2.06 ms	
CPU Time		0.39 ms	

# OPEN TELEMETRY



# OPEN TELEMETRY

## Open Telemetry provides you with:

- A single, vendor-agnostic instrumentation library
- An end-to-end implementation to generate, emit, collect, process and export telemetry data.
- Full control of your data with the ability to send data to multiple destinations in parallel through configuration.
- Open-standard semantic conventions to ensure vendor-agnostic data collection

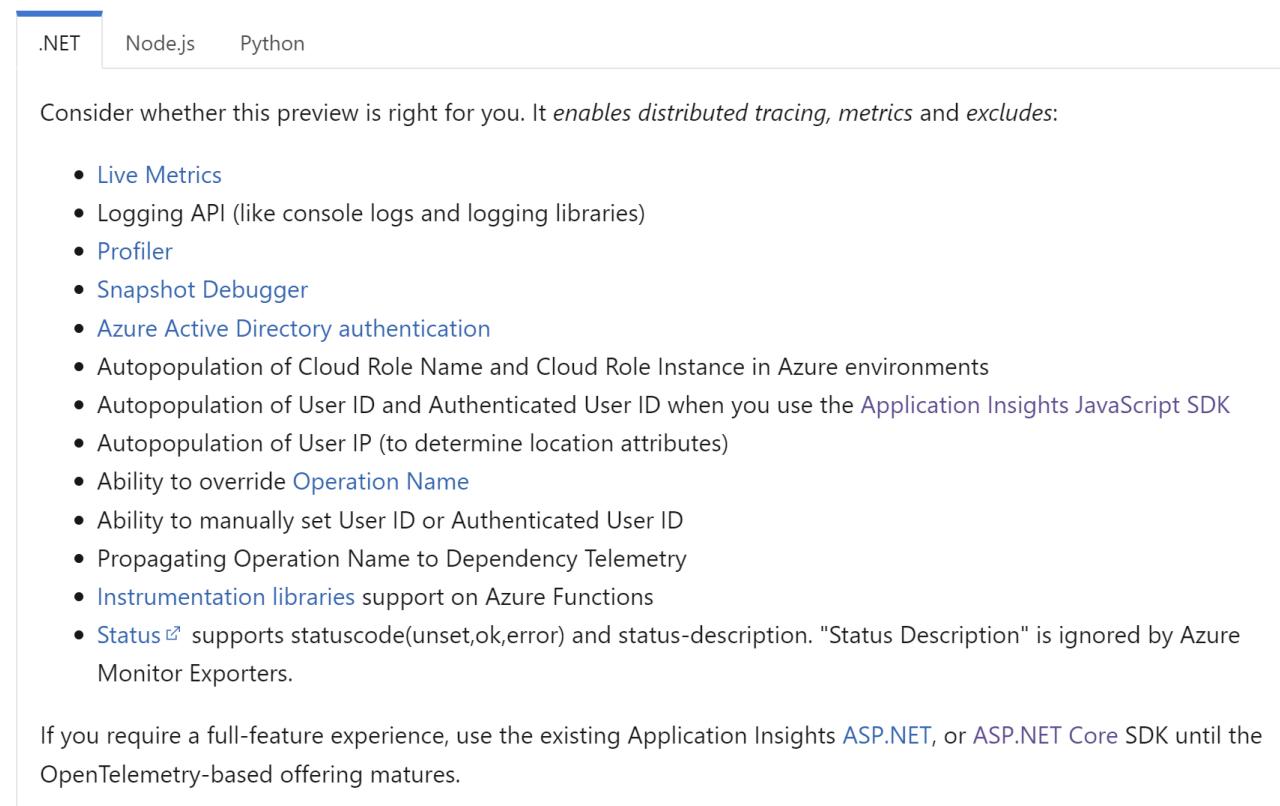


**OpenTelemetry**

# OPEN TELEMETRY APPLICATION INSIGHTS SUPPORT

- Supported through the Azure Monitor OpenTelemetry Exporter
- !!Still in preview!!

## Limitations of the preview release



The screenshot shows a user interface for selecting a technology stack. The '.NET' tab is selected, while 'Node.js' and 'Python' are shown as unselected options. Below the tabs, there is descriptive text and a bulleted list of limitations.

Consider whether this preview is right for you. It *enables distributed tracing, metrics and excludes:*

- [Live Metrics](#)
- Logging API (like console logs and logging libraries)
- [Profiler](#)
- [Snapshot Debugger](#)
- [Azure Active Directory authentication](#)
- Autopopulation of Cloud Role Name and Cloud Role Instance in Azure environments
- Autopopulation of User ID and Authenticated User ID when you use the [Application Insights JavaScript SDK](#)
- Autopopulation of User IP (to determine location attributes)
- Ability to override [Operation Name](#)
- Ability to manually set User ID or Authenticated User ID
- Propagating Operation Name to Dependency Telemetry
- [Instrumentation libraries](#) support on Azure Functions
- [Status](#) supports statuscode(unset,ok,error) and status-description. "Status Description" is ignored by Azure Monitor Exporters.

If you require a full-feature experience, use the existing Application Insights [ASP.NET](#), or [ASP.NET Core](#) SDK until the OpenTelemetry-based offering matures.



**OPEN TELEMETRY**

# SMART DETECTION



# SMART DETECTION

## Smart detection detects and notifies about various issues:

- **Failure Anomalies.** Notifies if the failure rate goes outside the expected envelope.
- **Performance Anomalies.** Notifies if response time of an operation or dependency duration is slowing down, compared to historical baseline.
- **General degradations and issues,** like Trace degradation, Memory leak, Abnormal rise in Exception volume and Security anti-patterns.

# SMART DETECTION

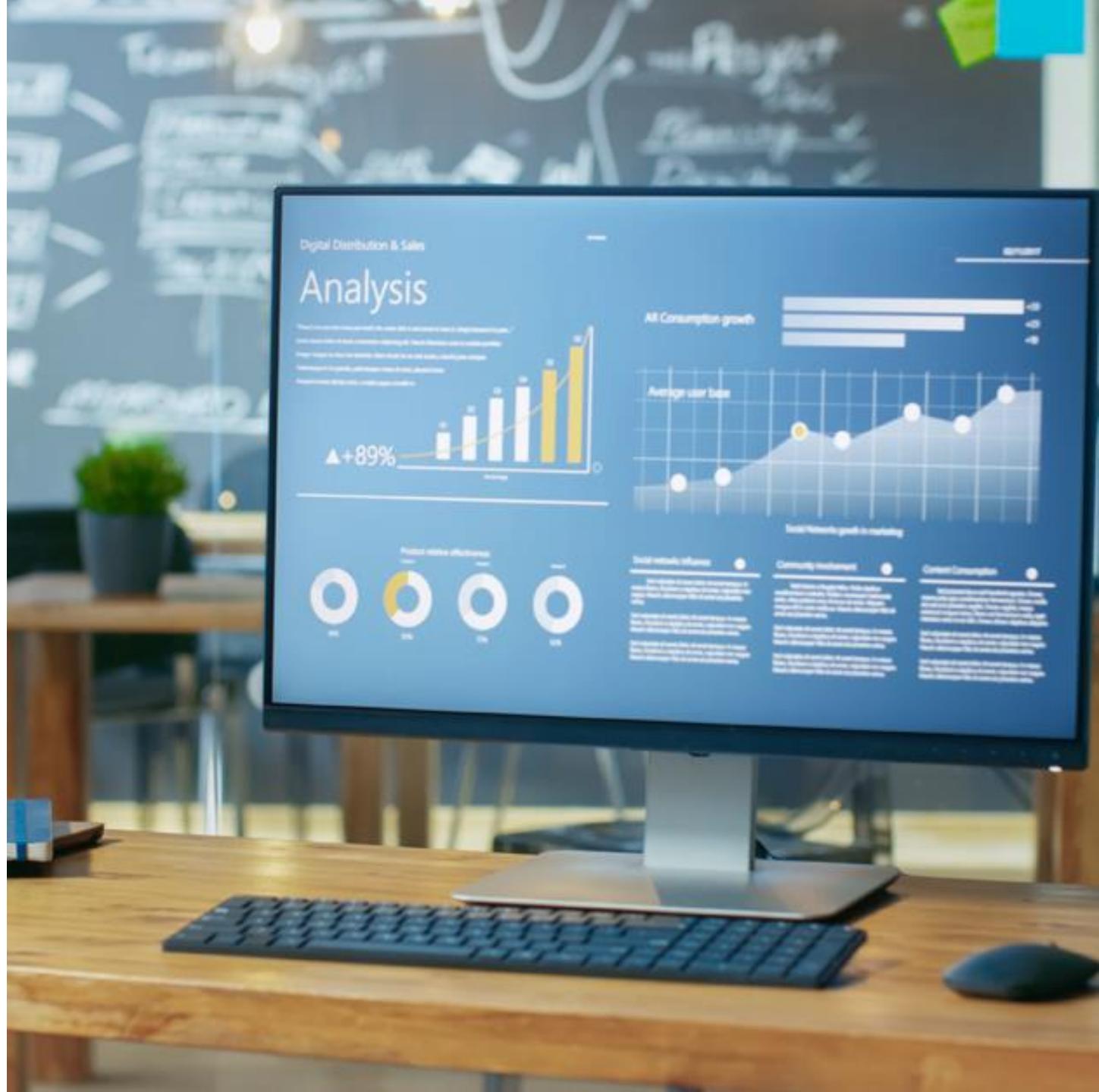
The screenshot shows the Microsoft Application Insights Smart Detection interface for the 'fabrikamprod' application. The left sidebar lists various monitoring and diagnostic tools, with 'Smart Detection' highlighted by a red box. The main pane displays a list of detected issues:

- Abnormal rise in exception volume (preview)**  
When: 7/28 5:30 AM - 7/29 5:29 AM  
What: 450% increase in 'System.Web.HttpException' volume compared to the previous 7 days
- Insecure form data transmission detected (preview)**  
When: 7/9 5:30 AM - 7/10 5:29 AM  
What: 2 operations or forms in your application submit data to insecure (non-HTTPS) URLs  
Note: Data provided by 1 user was potentially compromised due to this insecure data transmission
- Potentially insecure URL access detected (preview)**  
When: 7/9 5:30 AM - 7/10 5:29 AM  
What: 2 URLs were accessed by both HTTP and HTTPS protocols  
Note: 4 users accessed multiple URLs using HTTP instead of HTTPS
- Abnormal rise in exception volume (preview)**  
When: 6/29 5:30 AM - 6/30 5:29 AM  
What: 575% increase in 'System.Web.HttpException' volume compared to the previous 7 days
- Abnormal rise in exception volume (preview)**  
When: 6/14 5:30 AM - 6/15 5:29 AM  
What: Significant increase in 'Object doesn't support property or method 'Symbol.iterator'' volume compared to the previous 7 days



**SMART DETECTION**

# CONTINUOUS EXPORT



# CONTINUOUS EXPORT

- Metrics data is available for 90 days in Application Insights
- Sends telemetry data to Azure Blob Storage
- Files in JSON format

The screenshot shows the 'villagehall - Continuous export' configuration page in the Azure Application Insights portal. On the left, there's a sidebar with a search bar and a 'Continuous export' section. The main area displays a table with columns: STATUS, STORAGE ACCOUNT, and LAST EXPORT. A row is selected for 'mytestingstorage2' with a timestamp of '2 min ago'. To the right of the table is a large modal window titled 'Edit continuous export'. This modal has two tabs: 'Edit continuous export' (selected) and 'Data types to export'. The 'Edit continuous export' tab contains buttons for 'Enable', 'Disable', 'Delete', and a dropdown menu. The 'Data types to export' tab is titled 'Select the data types to export.' and lists several options with 'On' or 'Off' toggle switches. The listed data types include Availability, Custom Event, Dependency, and Exception.

STATUS	STORAGE ACCOUNT	LAST EXPORT
Configured	mytestingstorage2	2 min ago

**Data types to export**

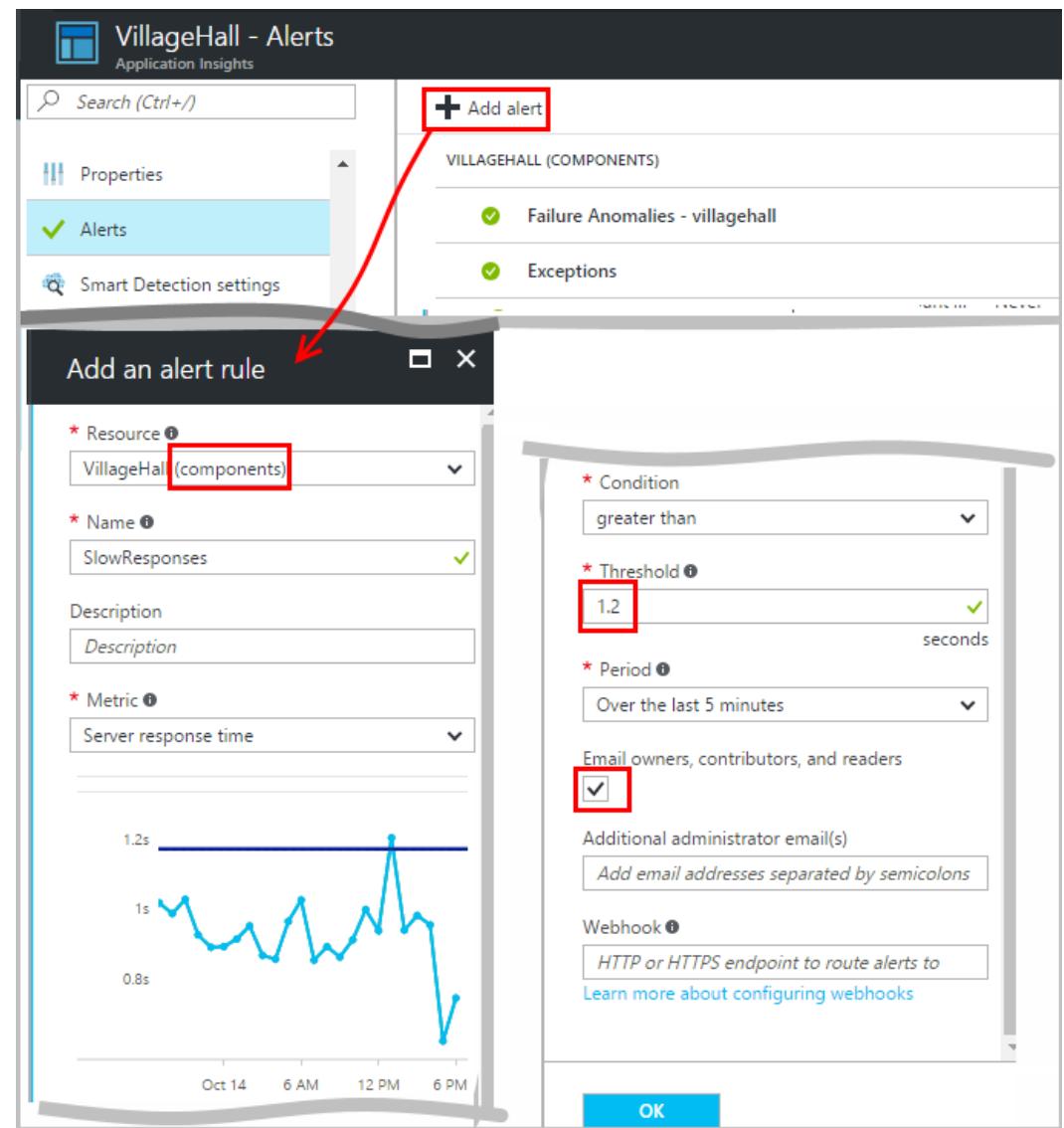
- Availability: On
- Custom Event: On
- Dependency: On
- Exception: On

# ALERTS



# ALERTS

- Define conditions on metrics
- Set problem interval
- Set e-mail address



# SITE AVAILABILITY



# SITE AVAILABILITY

- Define web tests to ping
  - Visual Studio web tests are also supported
- Ping from different countries
- Get alerts when unavailable

The screenshot shows the Microsoft Azure Application Insights interface for the 'fabrikamprod - Availability' resource. On the left, a sidebar lists various monitoring features: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Investigate, Application map, Smart Detection, Live Metrics Stream, Metrics, Search, and Availability. The 'Availability' item is highlighted with a red box. On the right, a modal window titled 'Create test' is open. It includes fields for 'Test name' (with placeholder 'Give your test a name'), 'Test type' (set to 'URL ping test'), 'URL' (set to 'http://fabrikamprod.azurewebsites.net'), and other configuration options like 'Parse dependent requests' (unchecked), 'Enable retries for availability test failures' (checked), 'Test frequency' (set to '5 minutes'), 'Test locations' (showing '5 location(s) configured'), 'Success criteria' (HTTP response: 200, Test Timeout), 'Alert type' (set to 'Near-realtime (PREVIEW)'), and 'Near-realtime alert status' (set to 'Enabled'). A note at the bottom of the modal says: 'You are using the new near-realtime alerting experience (preview). Please use View Alert button on the webtest details blade to configure the alert rule and notification settings'. A large orange arrow points from the 'Availability' link in the sidebar to the 'Add test' button in the 'Create test' dialog.

# CONCLUSION



# WRAPPING UP

- Application Insights is an easy-to-use telemetry service.
- Gives you a lot of useful information about your applications.
- Lots of features! Start using them to your advantage.



