APPLICATION INSIGHTS

SUCCINCTLY

BY ROBERTO ALBANO

SUCCINCTLY E-BOOK SERIES

Syncfusion®

www.dbooks.org

Application Insights Succinctly

By
Roberto Albano

Foreword by Daniel Jebaraj



Copyright © 2018 by Syncfusion, Inc.

2501 Aerial Center Parkway
Suite 200
Morrisville, NC 27560
USA
All rights reserved.

Important licensing information. Please read.

This book is available for free download from www.syncfusion.com on completion of a registration form.

If you obtained this book from any other source, please register and download a free copy from www.syncfusion.com.

This book is licensed for reading only if obtained from www.syncfusion.com.

This book is licensed strictly for personal or educational use.

Redistribution in any form is prohibited.

The authors and copyright holders provide absolutely no warranty for any information provided.

The authors and copyright holders shall not be liable for any claim, damages, or any other liability arising from, out of, or in connection with the information in this book.

Please do not use this book if the listed terms are unacceptable.

Use shall constitute acceptance of the terms listed.

SYNCFUSION, SUCCINCTLY, DELIVER INNOVATION WITH EASE, ESSENTIAL, and .NET ESSENTIALS are the registered trademarks of Syncfusion, Inc.

Technical Reviewer: Ed Freitas
Copy Editor: Courtney Wright
Acquisitions Coordinator: Tres Watkins, content development manager, Syncfusion, Inc.
Proofreader: Graham High, senior content producer, Syncfusion, Inc.

www.dbooks.org

Table of Contents

The Story behind the Succinctly Series of Books 7

About the Author 9

Chapter 1 Introduction 10

A first look at Application Insights 10

WAS or APM? Just telemetry! 10

A very brief history of Application Insights 11

Primary goals of Application Insights 11

Availability 12

Performance 12

Usage 12

Scope of this book 13

Chapter 2 How Do We Start? 14

Pricing options for Application Insights 14

Microsoft Azure subscription 15

Creating our first Application Insights resource 16

Chapter 3 The Dashboard 19

The instrumentation key 19

Getting started 20

The Application map 21

Live Metrics Stream 21

Chapter 4 Metrics Around Us 23

Metrics Explorer 23

What is the Metrics Explorer? 23

Metrics Explorer Favorites 24

Set the time range for the Metrics Explorer 28
Going deep into the details of a metric record 28
List of metrics 29
Usage metrics 30
Failure metrics 30
Server metrics 31
Availability metrics 31
Client metrics 31
Performance counter metrics 32
Chapter 5 Access Control 33
Chapter 6 Alert Rules 37
Chapter 7 Consuming and Exporting Data 40
Consuming data with Power BI 40
Application Insights REST API 42
Continuous export 45
Chapter 8 Visual Studio and SDKs 47
Where to find the right SDK 47
Integrating with Visual Studio 47
What happened to my project? 49
References 49
Connected services 50
Configuration 51
Scripts 52
Error handling 53
Chapter 9 Writing Your Code 55
The TelemetryClient class 55

www.dbooks.org

Tracking methods 55
The TelemetryConfiguration class 57
Using different instrumentation keys for various environments 58
Creating and configuring resources 58
Assigning instrumentation keys at run time 60
Chapter 10 Application Insights Analytics 62
Querying the data 64
Columns selected and columns displayed 66
Date and time expressions 68
Managing the results 69
Exporting the grid 69
Shape your data 71
Conclusion 75

The Story behind the *Succinctly* Series of Books

Daniel Jebaraj, Vice President
Syncfusion, Inc.

Staying on the cutting edge

As many of you may know, Syncfusion is a provider of software components for the Microsoft platform. This puts us in the exciting but challenging position of always being on the cutting edge.

Whenever platforms or tools are shipping out of Microsoft, which seems to be about every other week these days, we have to educate ourselves, quickly.

Information is plentiful but harder to digest

In reality, this translates into a lot of book orders, blog searches, and Twitter scans.

While more information is becoming available on the Internet and more and more books are being published, even on topics that are relatively new, one aspect that continues to inhibit us is the inability to find concise technology overview books.

We are usually faced with two options: read several 500+ page books or scour the web for relevant blog posts and other articles. Just as everyone else who has a job to do and customers to serve, we find this quite frustrating.

The *Succinctly* series

This frustration translated into a deep desire to produce a series of concise technical books that would be targeted at developers working on the Microsoft platform.

We firmly believe, given the background knowledge such developers have, that most topics can be translated into books that are between 50 and 100 pages.

This is exactly what we resolved to accomplish with the *Succinctly* series. Isn't everything wonderful born out of a deep desire to change things for the better?

The best authors, the best content

Each author was carefully chosen from a pool of talented experts who shared our vision. The book you now hold in your hands, and the others available in this series, are a result of the authors' tireless work. You will find original content that is guaranteed to get you up and running in about the time it takes to drink a few cups of coffee.

Free forever

Syncfusion will be working to produce books on several topics. The books will always be free. Any updates we publish will also be free.

Free? What is the catch?

There is no catch here. Syncfusion has a vested interest in this effort.

As a component vendor, our unique claim has always been that we offer deeper and broader frameworks than anyone else on the market. Developer education greatly helps us market and sell against competing vendors who promise to “enable AJAX support with one click,” or “turn the moon to cheese!”

Let us know what you think

If you have any topics of interest, thoughts, or feedback, please feel free to send them to us at succinctly-series@syncfusion.com.

We sincerely hope you enjoy reading this book and that it helps you better understand the topic of study. Thank you for reading.

Please follow us on Twitter and “Like” us on Face-book to help us spread the word about the *Succinctly* series!





About the Author

Roberto Albano is a passionate developer with a strong aptitude for learning and sharing.

He started as a professional developer using various development languages, but his favorite was Delphi, which he has been using for seven years.

In 2002, his discovery of .NET, C#, and Visual Studio changed his life. His passion for development technologies grew even more in the following years, and he used these techniques to implement all kinds of applications, from client-server applications, to real-time applications, including web services and web apps.

In the past few years, he's been involved in many projects related to web application development, using the ASP.NET MVC paradigm, and has benefitted from various Azure features, such as Application Insights for application telemetry.

Due to his natural inclination for team aggregation and knowledge sharing, he loves to speak at technical events in Italy. For the same reason, he co-founded dotNET{podcast}, a community focused on interviewing experts on Microsoft technologies.

He has been a Microsoft MVP since 2016 in the category *Visual Studio and Development Technologies*.

Chapter 1 Introduction

If your company's success is related to the reliability and uptime of a web application or web service, your mandatory mission is to keep track of what's happening with it.

It is particularly important to have full control of it, which requires understanding three distinct aspects:

- **Availability**: How your web application (or service) is reachable from different geographical areas.
- **Performance**: How your web application (or service) is working, and what are the best-performing and worst-performing parts of it.
- **Usage**: How people use your web application (or service), and which parts are more appreciated.

To achieve these points, we can use Application Insights, a service available on the Microsoft Azure platform.

A first look at Application Insights

With Application Insights, we can monitor every kind of web application, even if it is a web portal or a web service.

The technology or programming language used to develop the web application being monitored is not relevant, since many SDKs are available. This means we can cover every possible combination.

An important thing to know about Application Insights is that we can monitor web applications or services regardless of their hosting. This means we can use Application Insights to check the health of our ASP.NET-based web app hosted on Microsoft Azure in the same manner as a PHP website hosted on an Apache web server.

So, although this service is part of Microsoft Azure, we do not need to host our web application on it to use Application Insights.

WAS or APM? Just telemetry!

This kind of service is not new; there are a lot of other services like this on the web, usually known as web analytics software (WAS).

Within this classification, you will find other software tools such as Google Analytics, Matomo (formerly Piwik), Statcounter, and many others.

In any case, the official classification from Microsoft's documentation about Application Insights is Application Performance Management (APM).

In the end, however it has been classified, the sense of this service will not change: it is a telemetry service.

> ***Note: Telemetry is an automated communications process by which measurements and other data are collected at remote or inaccessible points and transmitted to receiving equipment for monitoring. (Source: Wikipedia)***

The first time I heard about telemetry was while watching a Formula 1 car competition. At that time, the speaker mentioned that with this technology the racing team was able, in every moment of the race, to control all the vital parameters of the car, such as tire pressure, fuel level, brake efficiency, engine temperature, and so on. All these values appeared on the team manager's screen as a real-time flow of data, on grids or charts.

Let me explain why, from my point of view, this is a similar scenario to a web application. Suppose you are leading a racing team (your company), you have a speedy and affordable car (your product), and you hire a professional driver (your development team) to win the race against your competitors (the market).

You need telemetry, you need that flow of data, to understand in real-time the changes that happen in order to evaluate the race's status and the conditions in which your car is running, applying the necessary corrections to achieve the best performance, or reach the finish line.

If you miss or don't correctly evaluate this information, your car's engine could break in the middle of the race, or something else could happen, and it may never reach the finish line.

A very brief history of Application Insights

This service was born a few years ago. In its first version it was part of Team Foundation Services Online (today known as Visual Studio Team Services), and its main scope was to provide telemetry not only for web applications, but also mobile apps.

After a while, the service became part of Microsoft Azure, and now, the focus of this service has been limited to web applications.

In the first half of 2016, a telemetry service specifically targeted toward mobile apps was created. This service is called HockeyApp.

A few months later, Microsoft announced the availability of the HockeyApp Bridge App. It enables you to access data collected by HockeyApp in Application Insights through its **Analytics and Continuous Export** feature.

Primary goals of Application Insights

As previously stated, availability, performance, and usage of our web application (or service) are the three most important goals for Application Insights.

But behind these three goals are more aspects to consider, which we will cover here.

Availability

When we monitor availability, we would like to understand:

- If our web application is up and running.
- How our web application is reachable from different geographical locations, in order to understand the market in which we are interested in investing.

Let me give an example: we have a company that produces short pants, and we want to sell these pants through a web commerce site throughout the world. The servers that host this website are based in Ireland.

Given the nature of the product, we will sell more in the hottest season of the year, but the season changes depending on the continent. For this reason, we can watch for the amount of time that a user from the hottest side of the globe takes to reach our website.

Starting from this data, we could discover that:

- The site is not reachable in a reasonable amount of time. For instance, a user from Tokyo must wait three seconds to open the main page, and up to five seconds to make a catalog search.
- The visitor count increased since last week, and the trend is significantly interesting in the Far East, particularly from China and South Korea.

Keeping this in mind, we can opt for a geographical replication of our website in a more easily reachable location for our potential customers, such as Japan.

This kind of decision is so easy in the cloud era, where a webmaster can simply modify geo-replication with a couple of clicks and get it sorted in a few minutes. But is important to know where to invest your money for your business, since geo-replication can be inexpensive, but in the end, it’s not free.

Performance

Using the web commerce site scenario, we should investigate the performance of our site to give our potential customers the best user experience.

In our example, we should be able to determine what pages or functions in our website slow our catalog search. Maybe we can discover a bottleneck in our search algorithm or in an external service we have integrated into our platform.

Usage

In our investment plan, we must estimate the effort (regarding money and time) to determine the best direction to achieve the business growth we are expecting.

To reach this goal, we need to understand the usage of our website, meaning how users spend their time on it. For instance, we should know if they prefer to search the catalog using filters, or browse the photo gallery more than the product detail page, or when they have found an active promotion on the main page whether or not they visit the promotion page.

This data can influence a large part of our future investments. To give an example, if we understand that our visitors usually appreciate a page we have underestimated, we can invest in the enhancement of that page or feature, instead of reinvesting in another, more expensive, and less attractive page for our potential customers.

Scope of this book

In this book, we will journey through the features available for the Application Insights resources. We will not cover all of them, but it will be a good starting point to understand the power of Application Insights.

www.dbooks.org

Chapter 2 How Do We Start?

The Application Insights service is available through Azure, Microsoft's cloud, so we need a subscription for this platform to begin.

> ***Tip: Microsoft Azure is a cloud computing service that provides Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS) to individuals and companies to help them perform better in their business. Visit azure.microsoft.com for more information.***

There are various kinds of subscriptions for Azure, and it's easy to start a free one just to try Application Insights, or start a pay-as-you-go subscription to use the free Basic pricing option for Application Insights.

If you are a developer using Microsoft's technologies, you should already have Visual Studio 2013 or a newer version installed. In this book, we will be using Visual Studio Community 2017.

If you own a Visual Studio with MSDN subscription, please refer to the terms of use and remember that the credits are available for development and testing only, so it cannot be utilized in a production environment.

There is no minimum configuration regarding hardware because all the information from Application Insights can be gathered through a web browser.

Pricing options for Application Insights

At the time of writing this book, there are two pricing options available for Application Insights, known as Basic and Enterprise. The details are shown in the following figure.

	BASIC	ENTERPRISE
Base monthly price	Free	$15 per node
Included Data	1 GB **per month**	200 MB per node **each day**
Additional data	$2.30 per GB	$2.30 per GB
Data retention (raw and aggregated data)	90 days	90 days
Application Performance Management (APM) and Analytics features	Included	Included
Continuous export	$0.50 per GB	Unlimited
Connector for OMS Log Analytics	Not Included	Included
Multi-step Web Tests	$10 per test per month	$10 per test per month

Figure 1: Pricing Options

As you can see from Figure 1, the Basic option is free but limited, and can be used to create Application Insights resources for debugging or for low-traffic applications.

When we have more traffic on our site, we can decide to switch to the Enterprise option, which has a monthly cost and more powerful features, such as a greater amount of data included and an unlimited “continuous export” feature. We will explore the continuous export feature in the following chapters.

Microsoft Azure subscription

In this chapter, we’ll explore how to create an Application Insights resource within the Azure portal and connect it to an existing web application. Then we will repeat this step, creating an ASP.NET web application with an Application Insights resource directly from Visual Studio. Even if the end result of both procedures is the same, the latter mode is more convenient if you usually work with Visual Studio.

To create your first Application Insights resource, navigate to the Azure portal.



Tip: Bookmark the URL https://portal.azure.com since we’ll use it as the entry point to the Azure portal throughout this book.

If you have no subscription for Azure, you will need to create a new one, as explained previously. The Basic option is usually enough to get started with Application Insights.

www.dbooks.org

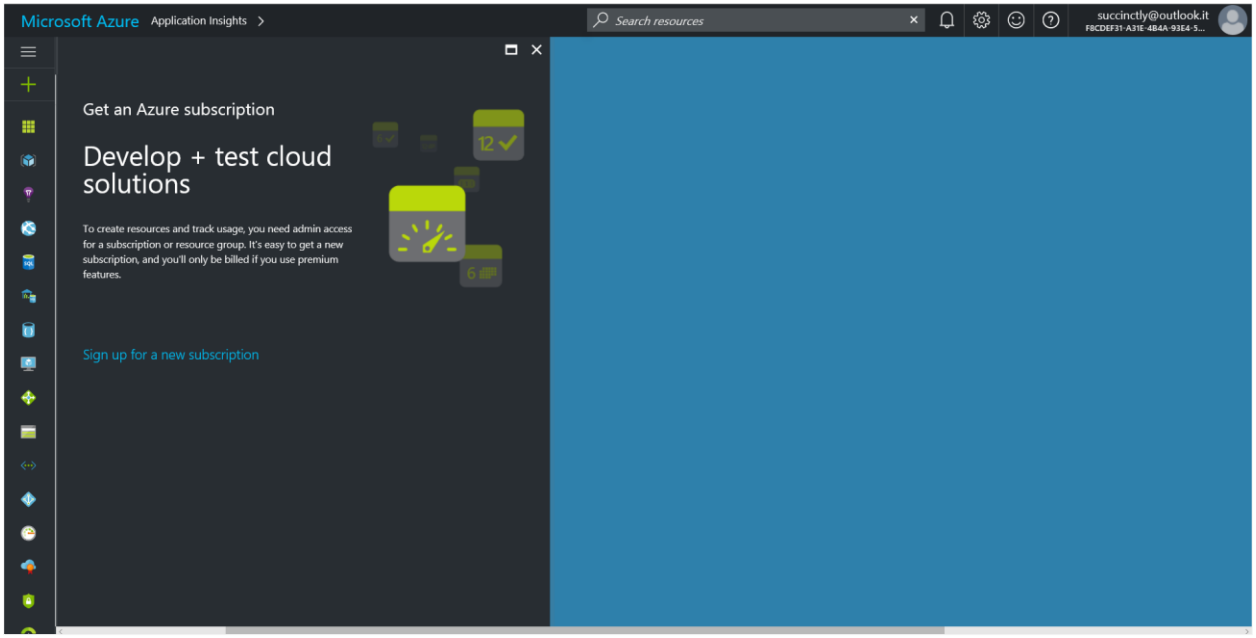


Figure 2: Azure Portal

Note: Since Azure is not really the topic of this book, it is assumed that you have already signed up for this service.

Creating our first Application Insights resource

On the left-hand side of the Azure portal, you will find a vertical bar with a green **+** symbol—this will open the **Marketplace** blade. Here you can locate the **Application Insights** item located in the **Developer Tools** section.

Note: The term "blade" on Microsoft Azure represents a vertical panel that is usually a container for other information. The drill-through navigation on this portal is designed using a progressive blade opening on the right-hand side of the previous blade.

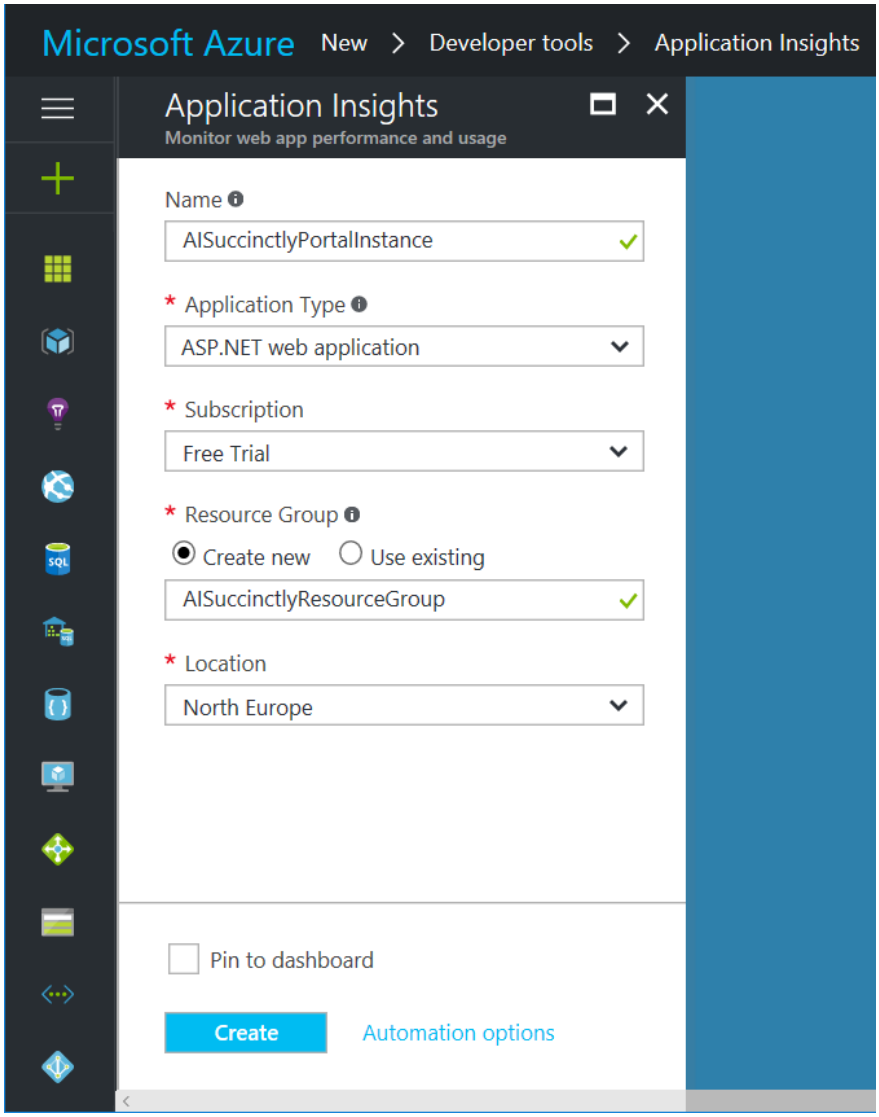


Figure 3: The Application Insights New Resource Blade

The following information is required to create an Application Insights resource:

- **Name**: This is the name of the resource. It must be unique in your Azure subscription.
- **Application Type**: This is the kind of implementation that we will monitor. It could be one of the following values:
  - **ASP.NET web application**: A typical ASP.NET web application, such as an ASP.NET MVC application, a Web Forms application, or a WCF Service.
  - **Java web application**: A typical Java web application.
  - **HockeyApp bridge application**: This option is to allow Application Insights to use data gathered with HockeyApp. When you select this option, two additional fields will be displayed (HockeyApp Token and HockeyApp Application) to complete the connection between Application Insights and the HockeyApp environment.
  - **General**: An HTML-based application, regardless of the development language.
- **Subscription**: The Azure subscription the resource is associated with.

www.dbooks.org

- **Resource Group**: This is the resource group in which the resource is collected. It is a sort of resource container, useful for Azure resource management.
- **Location**: The geographic location where the resource will be running. Currently the available options are East U.S., South Central U.S., North Europe, and West Europe.

Once you click the **Create** button, the resource will be created in seconds.



Tip: Since the Azure portal interface is designed to be as simple as possible, you can explore the different sections yourself to look at what is available for each resource.

At this point you should have your own Application Insights resource ready to go. In the next chapter we will explore the dashboard to understand the available features for that resource.

Chapter 3 The Dashboard

Once the Application Insights resource is available in the **All resources** blade, we can access all the information and configuration details available for the resource.

In this chapter, we will explore some interesting sections that will help us understand the fundamental concepts of Application Insights.

The instrumentation key

The first thing to explore is the **INSTRUMENTATION KEY** field, which is available on the **Properties** blade.

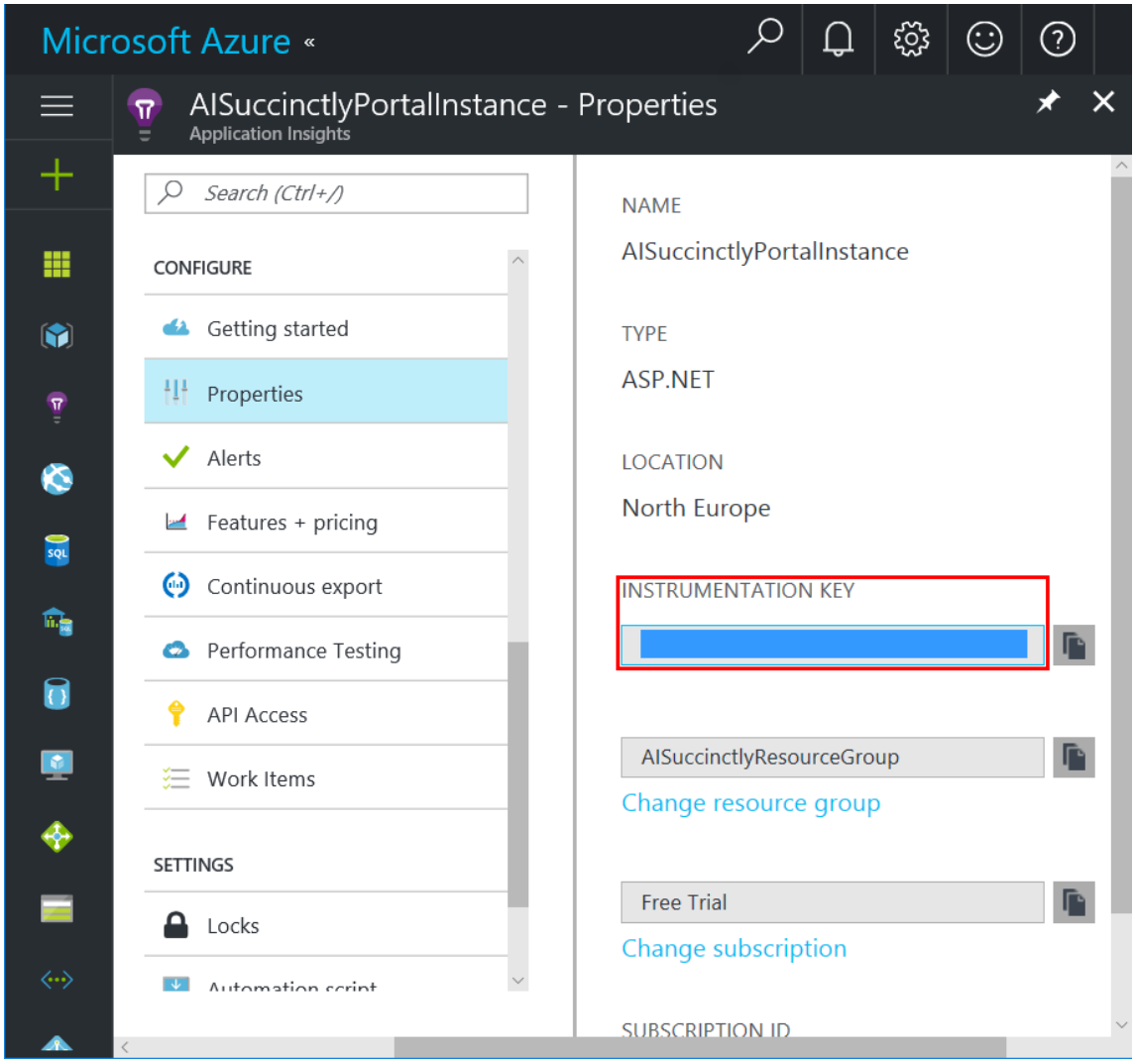


Figure 4: The Instrumentation Key

www.dbooks.org

The instrumentation key is a GUID that identifies a resource in every web application we will use. It is generated automatically by Azure during the resource's creation process.

> ***Note: GUID is short for Globally Unique Identifier. It is a unique identifier that can be created in several ways, usually with a combination of a few unique OS settings or components, such as current IP address or network MAC address, clock date/time, etc.***

When a resource has been created on the Azure portal, you need to copy the instrumentation key and paste it in the configuration of the web application that will use it. If the resource is created within Visual Studio during the creation of a web application, this step is not needed since the IDE will do this for us.

Getting started

In this section we will find a lot of information (and code) that is useful to integrate this resource into our application, both on the server side and the client side. This means you could use it on the server side to track the performance of the business logic, and you could use it on the client side to understand the usage of your web pages, regardless of the language used (this is very important).

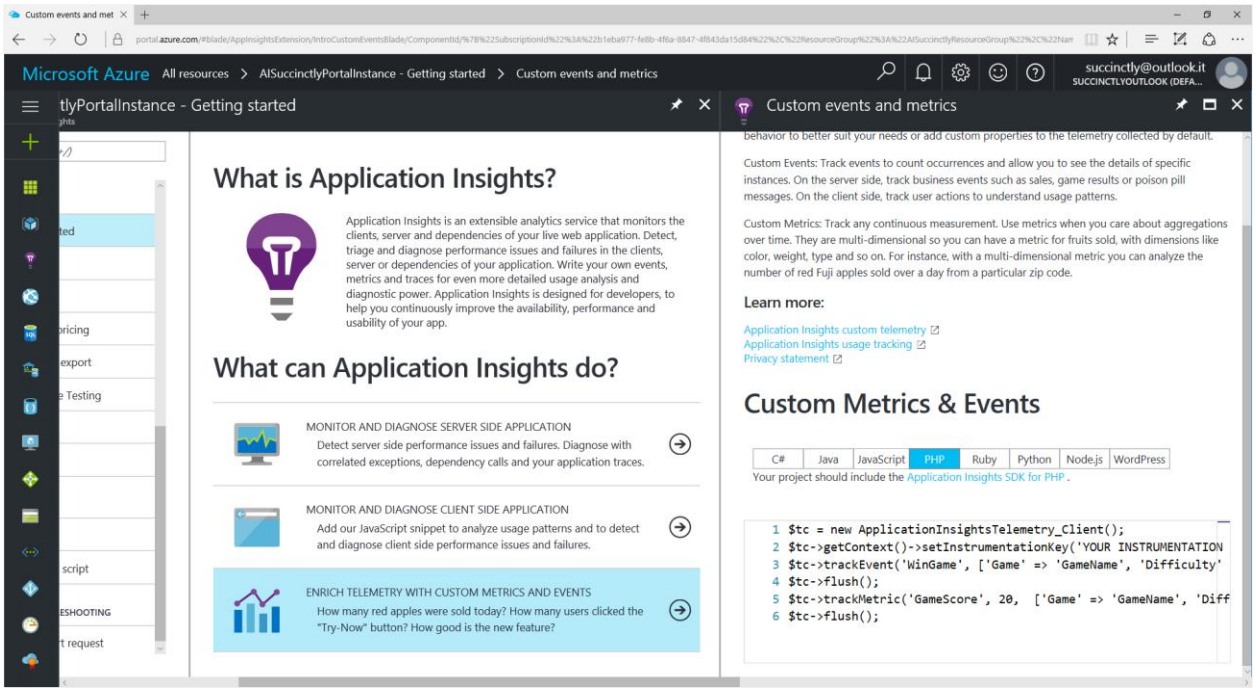


Figure 5: The Getting Started Section

For instance, if you want to use this resource in a PHP-based web application, here you can learn the basics of how to use it. There is the example code to copy, which represents the starting point for a newbie on Application Insights.

The Application map

The **Application map** is a visual feature that shows the dependency relationships between the components of a monitored web application.

In a brand new Application Insights resource, it will appear as a blank canvas. In a running environment, you will see something like this:

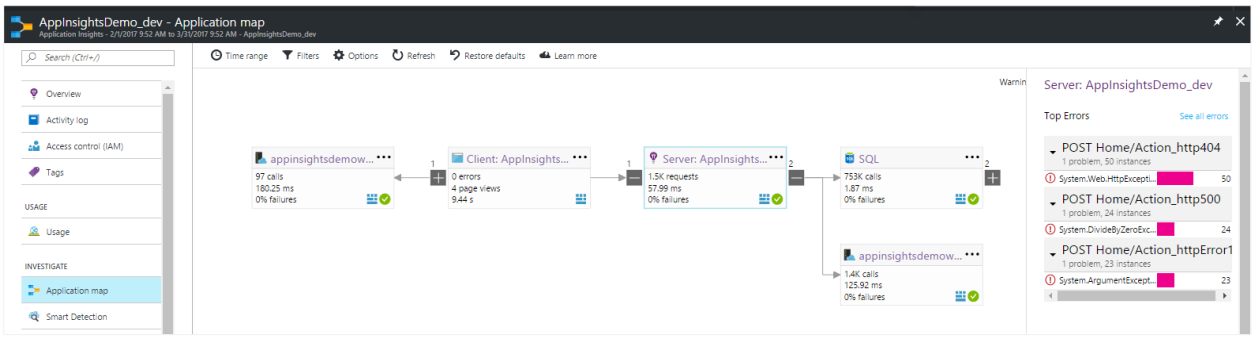


Figure 6: The Application Map Canvas

As you can see in Figure 6, we have the graph of the relationship in which we can find a SQL Server (on the right) and the hosted website (first item on the left).

Selecting one of the components, we can see the related items in a separate pane on the right (in Figure 6, these are requests), and we can choose items from the context menu (**…**) on every component to see a filtered set of information.

Another interesting thing to know is the **time range** filter, which allows us to see the part of the relationship defined for that given period (if it is available in the resource's data, of course).

Note: Every now and then we can see this icon  ***appearing somewhere on the screen. It is a direct link to a powerful feature that we will see in Chapter 10, Application Insights Analytics.***

Live Metrics Stream

Another interesting feature within the Azure portal is the **Live Metrics Stream**—the name says it all.

www.dbooks.org

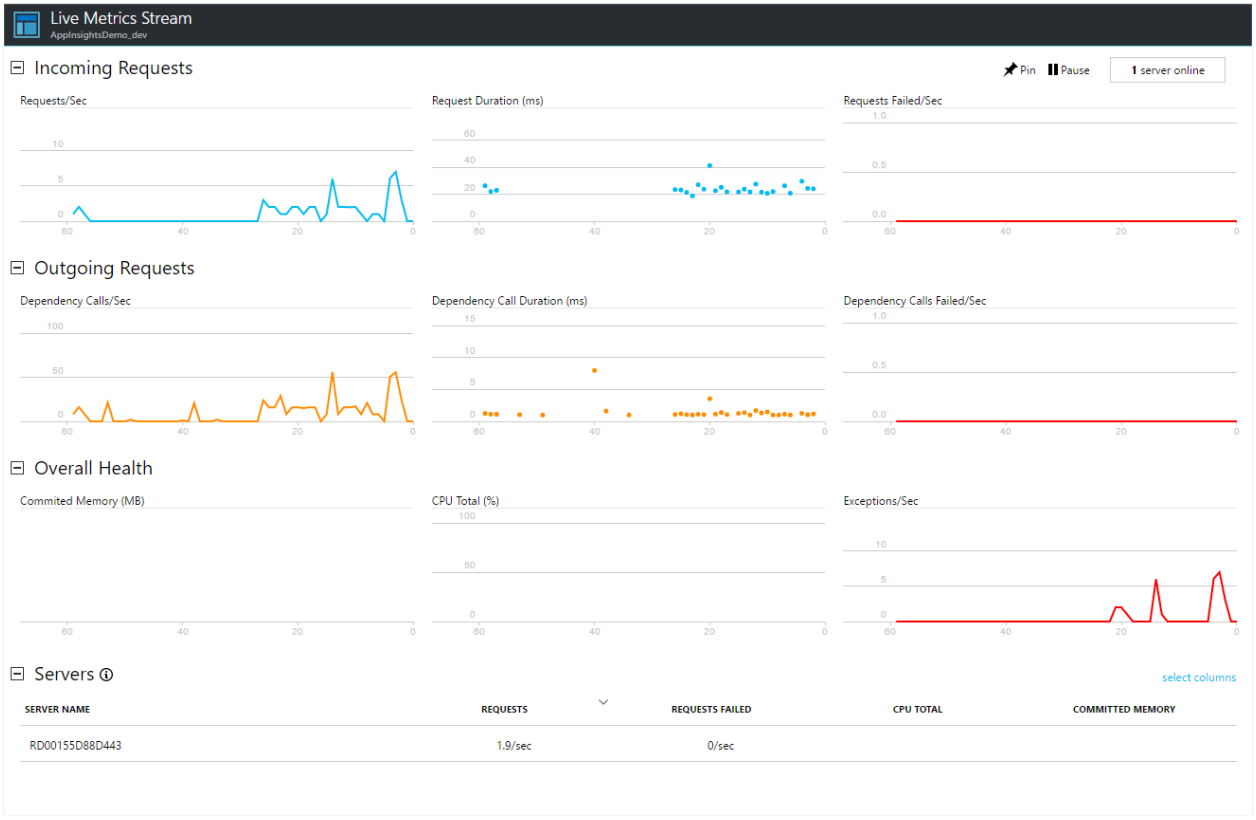


Figure 7: Live Metrics Stream

Like a heartbeat, the Live Metrics Stream will allow us to see what is happening in near-real time on the web application or service we are monitoring. It's in "near-real time" because there is a minimal latency, around one second, but for its purpose, we can consider it a real-time monitor.

This feature can be handy if you are releasing a newer version of your web application in a production environment to apply a fix (e.g., for slow performance) and you want to understand immediately if this release is working as you expected.

Another example is if you are monitoring a specific moment of your web application, to understand how it is doing—for instance, in a live-event scenario where you would like to see a connected users counter. Another example is when you are verifying for repeated failures.

> ***Note: When using the SDK version 2.4.0 or later, there will be additional filters on the right-hand side of the Live Metrics Stream panel for creating and using customized KPIs.***

Keep in mind that this feature doesn't provide search capabilities.

Now that you have seen a little more about the features available with your Application Insights resource, it's time to take a look at metrics.

Chapter 4 Metrics Around Us

Every bit of information we gather from the web application or service that we are monitoring is called a *metric*. It may stand for a measured value or a count of some event or behavior related to that web application. Each metric could contain multiple bits of information, called *data points*.

In this chapter, we will dive into all the assorted flavors of metrics available within Application Insights.

Metrics Explorer

A metric, in the terminology of Application Insights, is the measurement of the performance or efficiency of the software being monitored, over one or more of its features.

One of the most impressive blades that we can use for an Application Insights resource is the Metrics Explorer. You can reach it from the **Overview** section by clicking the **Metrics Explorer** button.

What is the Metrics Explorer?

The Metrics Explorer might be the first place to land when you want to track for an exception or other data gathered about your web application. Here you can quickly find accurate information in a few clicks.

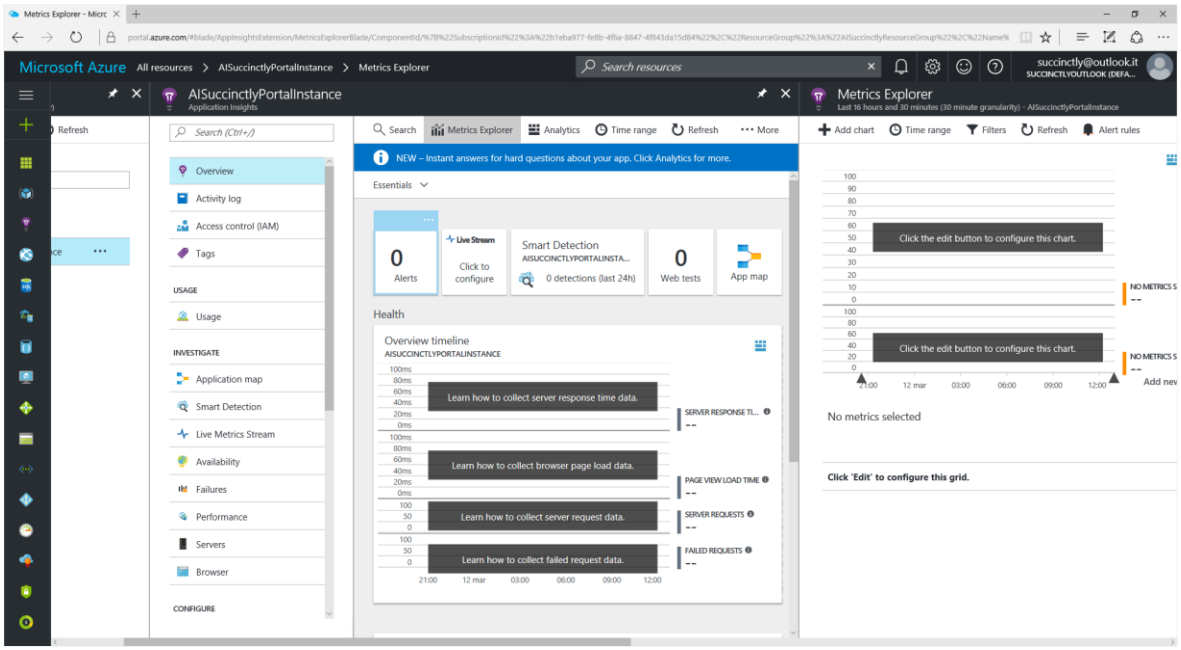


Figure 8: The Metrics Explorer Blade

www.dbooks.org

In Figure 8, we have no data since the resource is brand new. Once the resource has collected data from the web application, we will see something like what's shown in Figure 9.

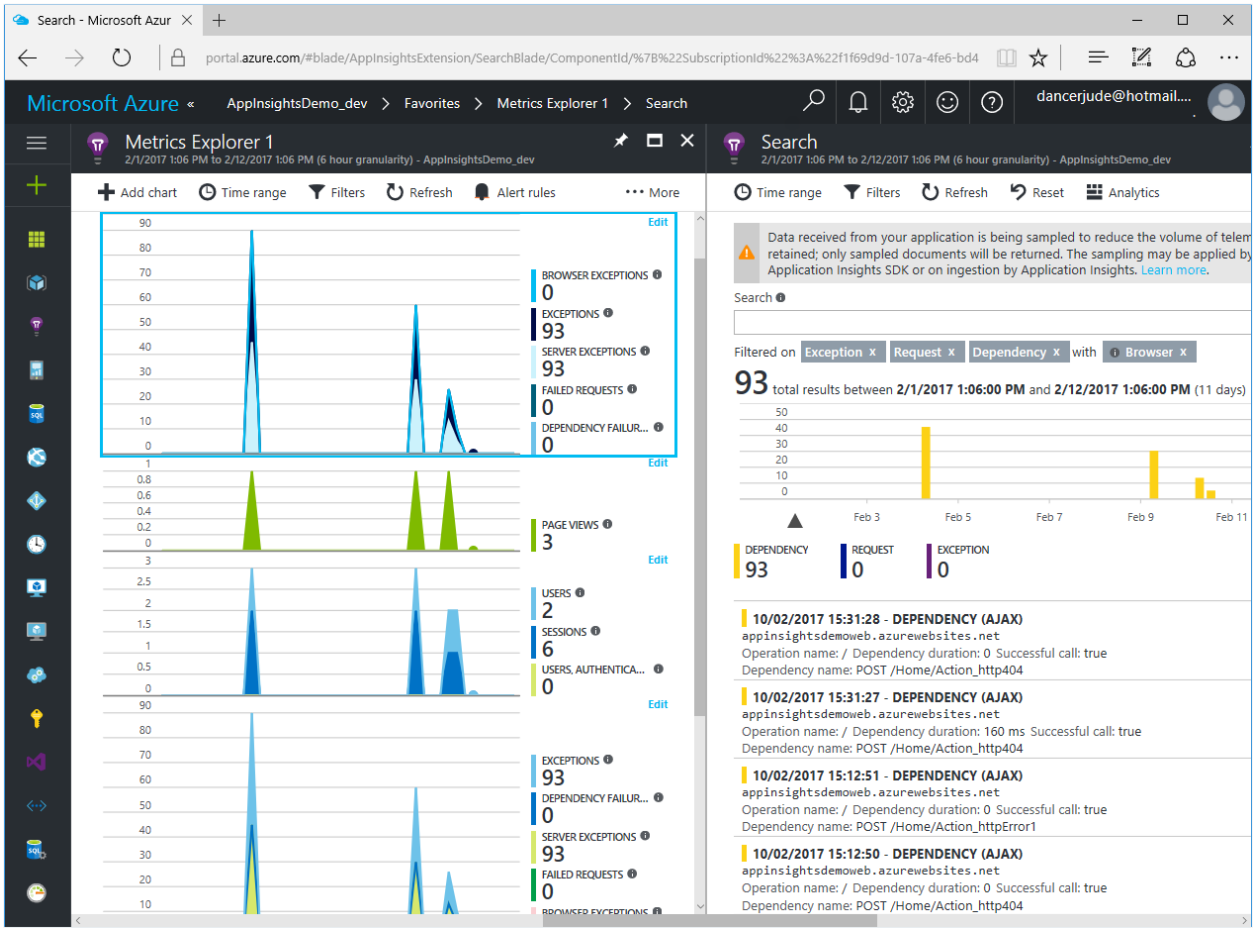


Figure 9: The Metrics Explorer Blade for a Demo Resource

The Metrics Explorer blade can be configured using specific information we desire to create a view of metrics, as you can see in Figure 9.

> ***Note: The term "view" might not be proper for this feature, and it's not part of the official nomenclature, but we will use it here to better explain the idea behind this blade.***

Metrics Explorer Favorites

It is possible to create more views by saving them to the **Favorites** list. For instance, we could save a view named "exceptions" in our favorites with all the metrics related to the exceptions we want to see there, or another view called "visitors" with the details about page views and other information about site visits.

This ability to save and reuse views could be beneficial if you will be intensively using the Metrics Explorer.

Creating a favorite view

Starting from the Metrics Explorer, you could find a situation like in Figure 8, with an empty blade that is ready to host your metrics. Every placeholder can host a chart for a metric.

Select the metrics to host for the desired destination chart by clicking the **Edit** label in the top-right corner of the graph. In this edit mode, a new blade will appear on the right. Here, you can choose one of the metrics available from the list or use the search text box to filter out specific metrics.

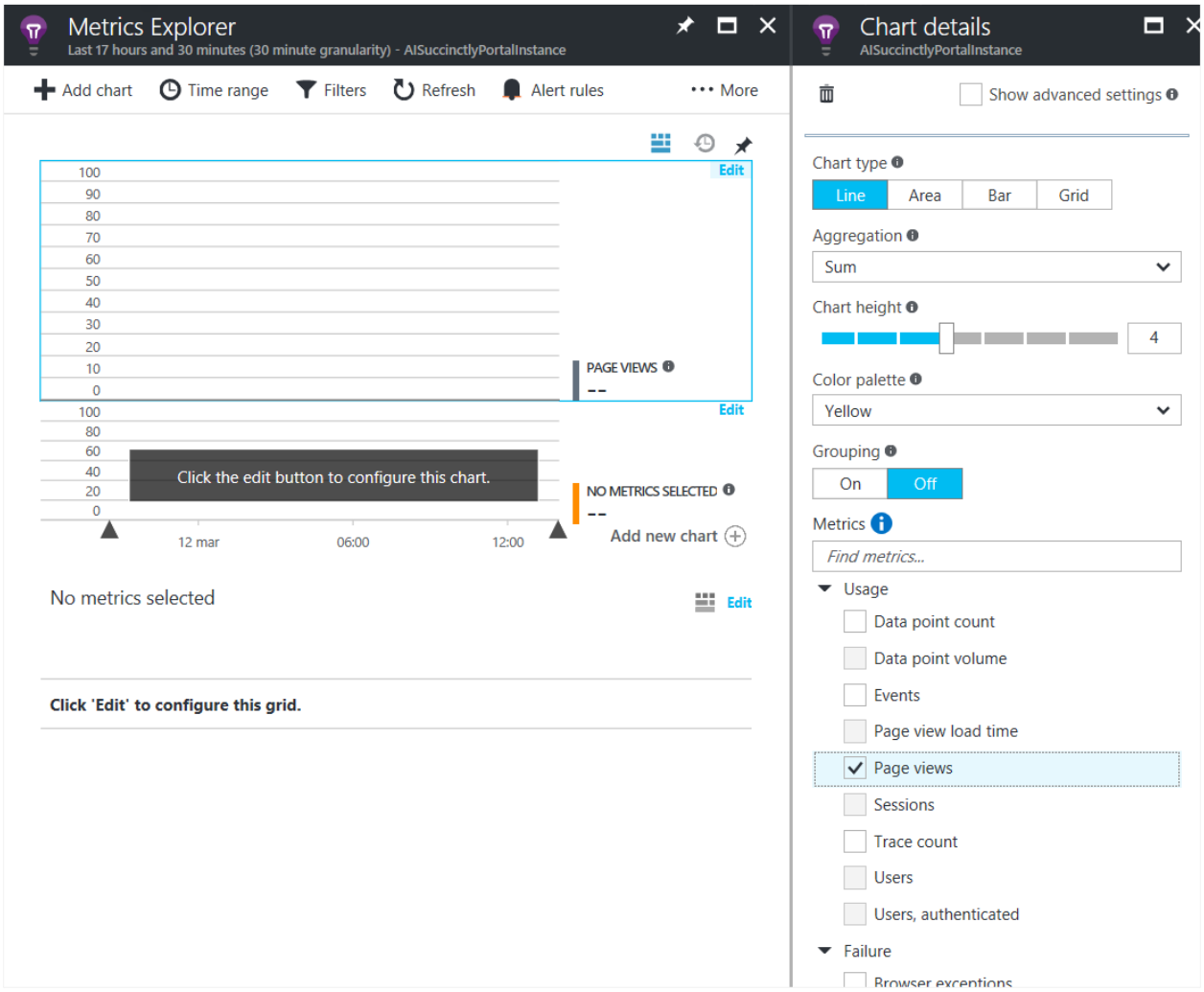


Figure 10: Creating a Metric Explorer View

In a single chart placeholder, we can display more than one metric, but when we select a metric in the **Chart details** blade, some others will be unavailable, as you can see in Figure 10. This behavior happens because the metrics that are unavailable don't have the same scale and aggregation types, so they cannot appear on the same chart.

www.dbooks.org

As you can see in the upper-right side of Figure 10, we can choose how to shape the data seen on the screen by selecting one of the **Chart type** values:

- Line
- Area
- Bar
- Grid

To show the different chart types, the following figure shows the same **Page Views** metric used to create four charts, one per chart type:

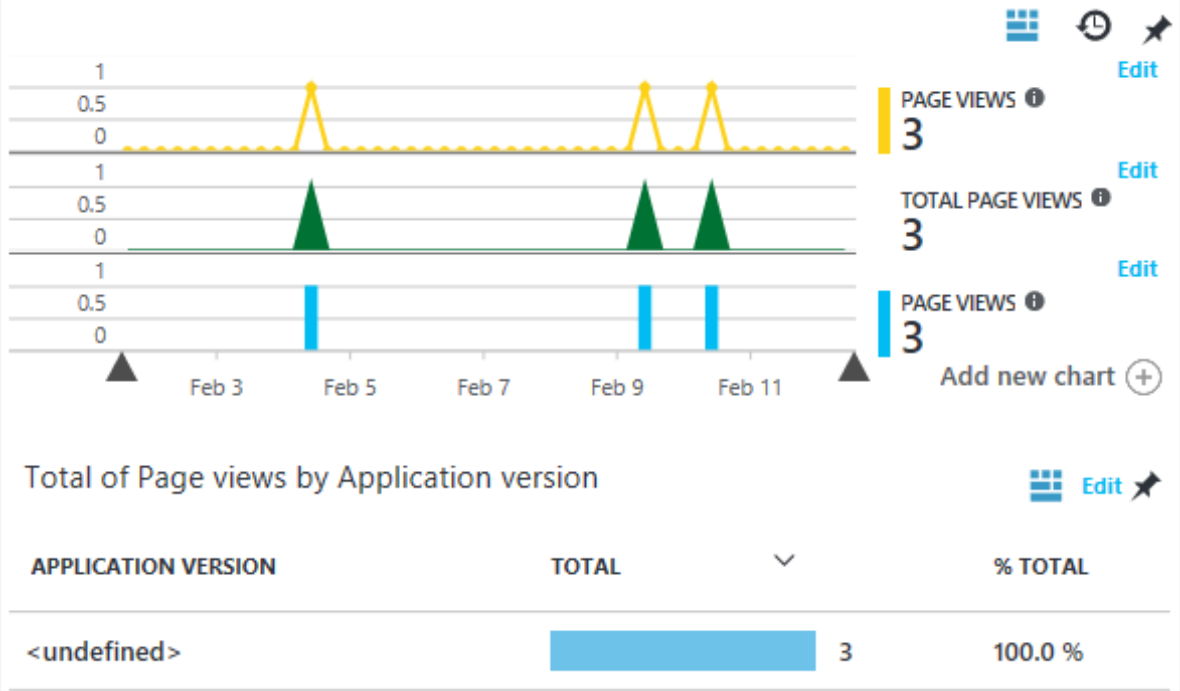


Figure 11: Page Views Metric in Four Different Chart Types

Depending on the chosen metric and chart type, there could be other settings available, such as a graph height or an aggregation field.

In the Metrics Explorer blade (as well as in a view that can be saved as a favorite), we can add more placeholders using the **Add new chart** option, as shown in Figure 11. There could be up to nine chart placeholders in a single blade, regardless of the chart type or height.

A view can be saved to the Favorites list as a shared or personal one, using the option seen in Figure 12.

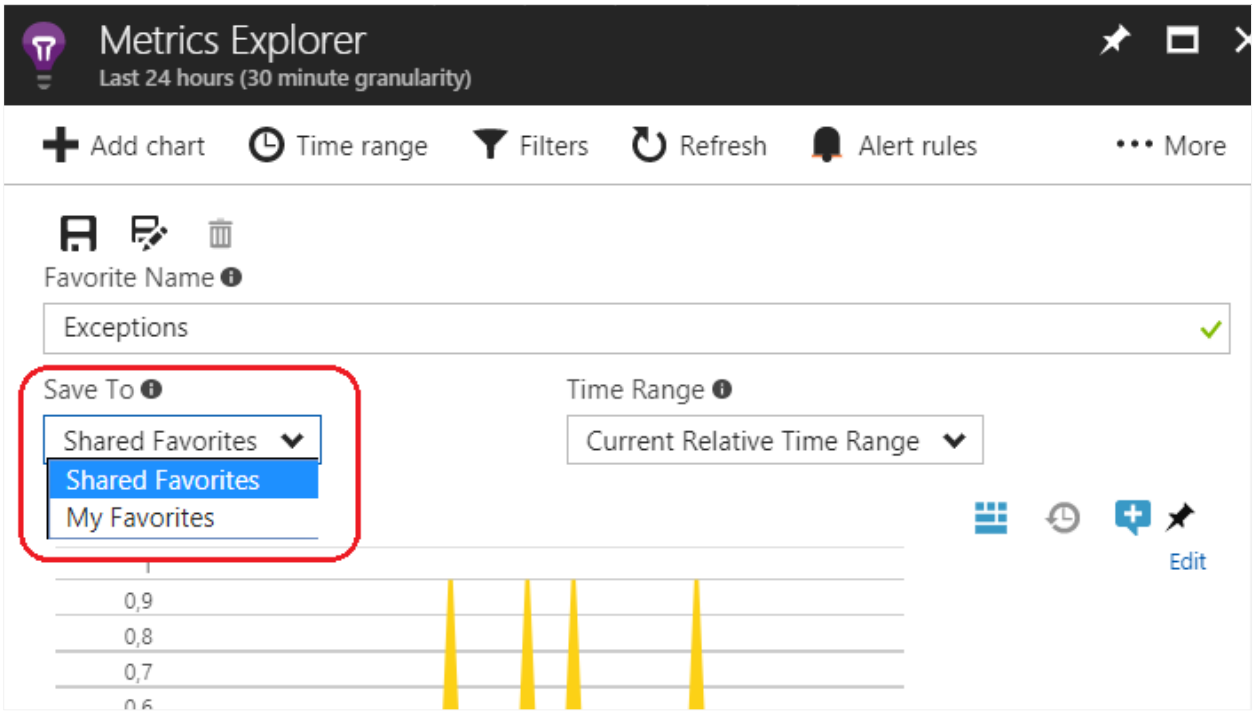


Figure 12: Saving a Favorite View

The term *shared* means that any other user that has access to the resource can see and edit this view. Otherwise, the view is only visible to its creator.

Loading a favorite view

From the **Overview** blade, you can reach the list of **Favorites**.

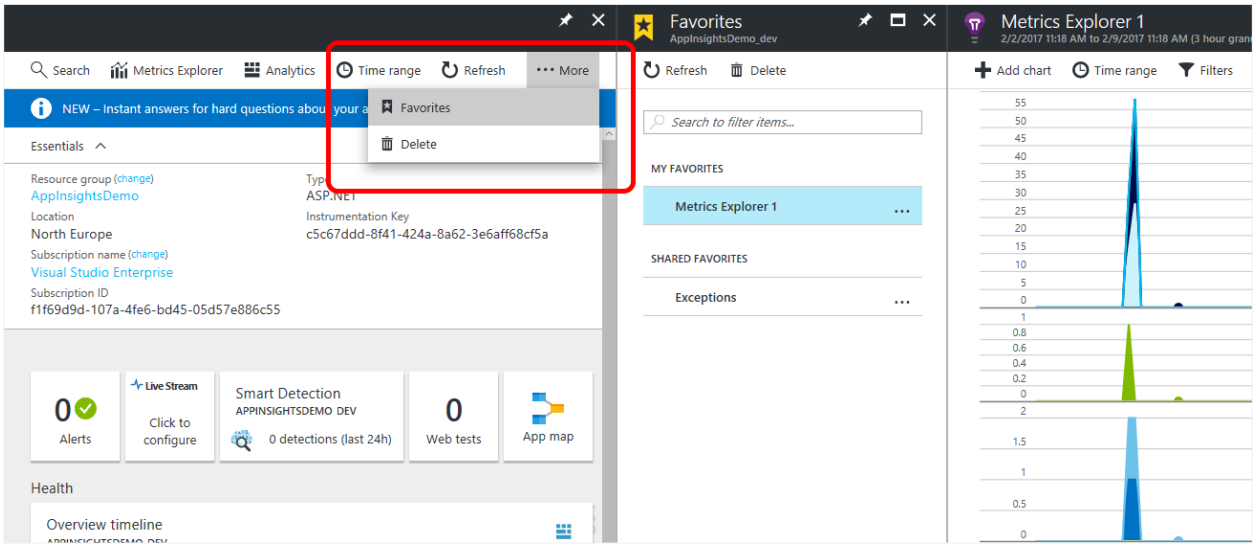


Figure 13: Favorites List

www.dbooks.org

In the **Favorites** blade, there are two sections where we will find all our favorites: **My favorites** and **Shared favorites**. In the first section, the user will find their own favorites, while the second section contains the favorites created and shared by other users. The shared favorites can be used and edited as well.

Here we can choose a favorite view to load and display with a single click. Once selected, the usage is the same as using the Metrics Explorer directly.

Set the time range for the Metrics Explorer

Once we have chosen the metrics to display, we can select the time range for which we want to display data within the Metrics Explorer.

The possible values are:

- Last 30 minutes
- Last hour
- Last 12 hours
- Last 24 hours
- Last 48 hours
- Last three days
- Last seven days
- Last 30 days
- Custom (choose your custom start time and end time)

There is also the option of selecting the granularity of the data to display; the values here depend on the time range selected. As you can see in the Pricing options for Application Insights section, the data retention is 90 days (from the current day) in both the Basic and Enterprise pricing options.

In Chapter 7, Consuming and Exporting Data, we will see more about exporting features, which are useful when data must be preserved past the retention period.

Going deep into the details of a metric record

All that we have seen up until now for the Metrics Explorer section was related to displaying available data, but what about the composition of the data itself? Using one of the charts, we can retrieve a set of records about the metrics included. Selecting one of them, we will see more details about it.

To give an example, let's see the **Page views** metric for a page of the web application we are monitoring.

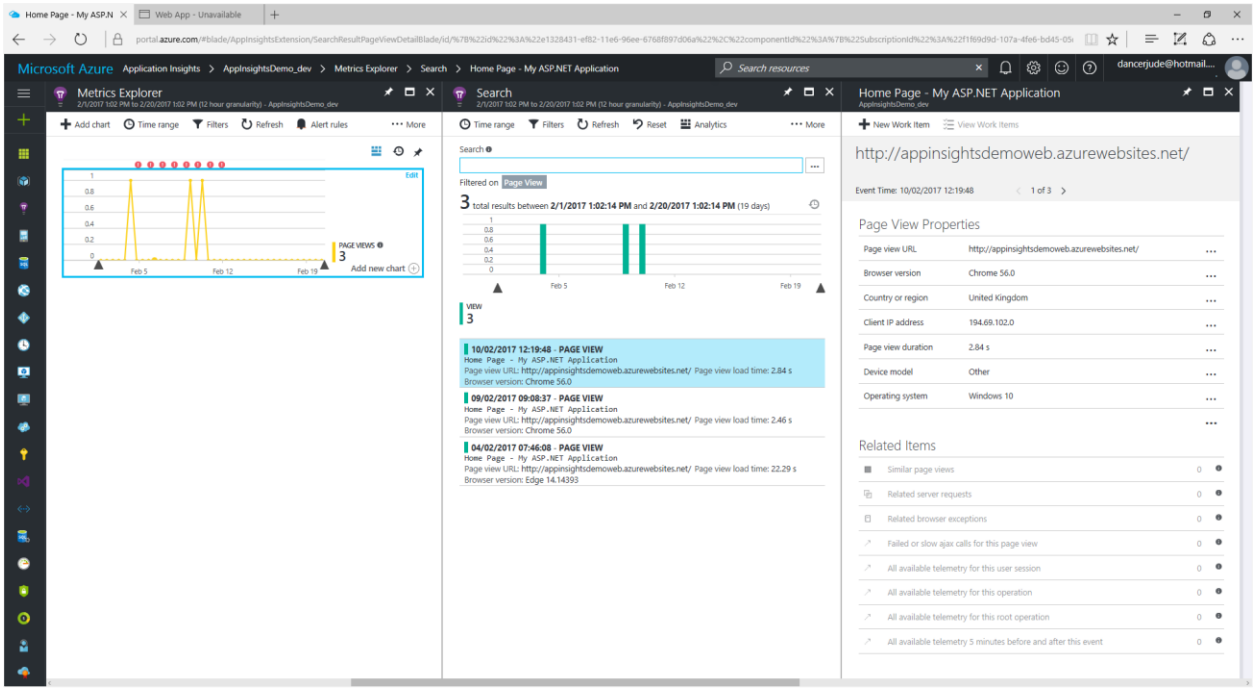


Figure 14: Details for a Page View Metric Record

As you can see in Figure 14, the details about the selected record appear on the right-hand blade. In this case, you can see that the page was visited by someone from the United Kingdom using a Chrome browser on Windows 10. The visitor remained on the page for less than three seconds.

The information present on this metric is related to its scope. So, by choosing other metrics, we will collect other types of information.

List of metrics

When we search for a metric to chart with Metrics Explorer, we find these groups:

- Usage
- Failure
- Server
- Availability
- Client
- Performance Counter

Each group in this list can contain several metrics to detect usage and track behavior of a web application. I will provide a brief description of these metrics one by one to give you a better understanding of each.

www.dbooks.org

Usage metrics

The following table shows a list of the usage metrics.

Table 1: Usage Metrics

Metric	Description
Data point count	Number of data points sent
Data point volume	Volume of data sent
Events	Count of custom events
Page view load time	Page view load time
Page views	Count of page views
Sessions	Count of user sessions
Trace count	Count of trace documents
Users	Count of distinct users
User, authenticated	Count of distinct users who signed in

Failure metrics

The following table shows a list of the failure metrics.

Table 2: Failure Metrics

Metric	Description
Browser exceptions	Count of uncaught exceptions thrown in the browser
Dependency failures	Count of failed calls made by the application to external resources
Exceptions	Combined count of all uncaught exceptions
Failed requests	Count of HTTP requests marked as failed (response code >= 400, except 401)
Server exceptions	Count of uncaught exceptions thrown in the server application

Server metrics

The following table shows a list of the server metrics.

Table 3: Server Metrics

Metric	Description
Dependency calls	Count of calls made by the application to external resources
Dependency duration	Duration of calls made by the application to external resources
Server requests	Count of HTTP requests completed
Server response time	Time between receiving an HTTP request and finishing sending the response

Availability metrics

The following table shows a list of the availability metrics.

Table 4: Availability Metrics

Metric	Description
Availability	Availability (average)
Test duration	Test duration (average)

Client metrics

The following table shows a list of the client metrics.

Table 5: Client Metrics

Metric	Description
Browser page load time	Time from user request until DOM, stylesheets, scripts, and images are loaded
Client processing time	Time between receiving the last byte of a document until the DOM is loaded (asynchronous requests may still be processing)
Page load network connect time	Time between user request and network connection (includes DNS lookup and transport connection)

www.dbooks.org

Metric	Description
Receiving response time	Time between the first and last bytes, or until disconnection
Send request time	Time between network connection and receiving the first byte

Performance counter metrics

The following table shows a list of the performance counter metrics.

Table 6: Performance Counter Metrics

Metric	Description
Available memory	Physical memory immediately available for allocation to a process or for system use
Exception rate	Count of handled and unhandled exceptions reported to Windows, including .NET exceptions and unmanaged exceptions that are converted into .NET exceptions
HTTP request execution time	Execution time of the most recent request
HTTP request rate	Rate of all requests to the application per second from ASP.NET
HTTP requests in application queue	Length of the application request queue
Process CPU	The percentage of elapsed time of all process threads used by the processor to execute the application process
Process IO rate	Total bytes per second read and written to files, network, and devices
Process-private bytes	Memory exclusively assigned to the monitored application’s processes
Processor time	The percentage of time that the processor spends in non-idle threads

In this chapter, you have hopefully learned a little bit more about the available metric types. In the next chapter, you will learn how to assign users the appropriate rights to access and use each one.

Chapter 5 Access Control

Usually, data collected from Application Insights resources can be analyzed by people with different skills to check a web application from various points of view, covering aspects from performance to marketing.

For this reason, in a large team, we could have more users accessing the dashboard and treat the same data in multiple ways. This means that every user could have different privileges on different parts of that data.

With the **Access control (IAM)** feature, we can assign distinct roles to different people, defining limited access to specific information.

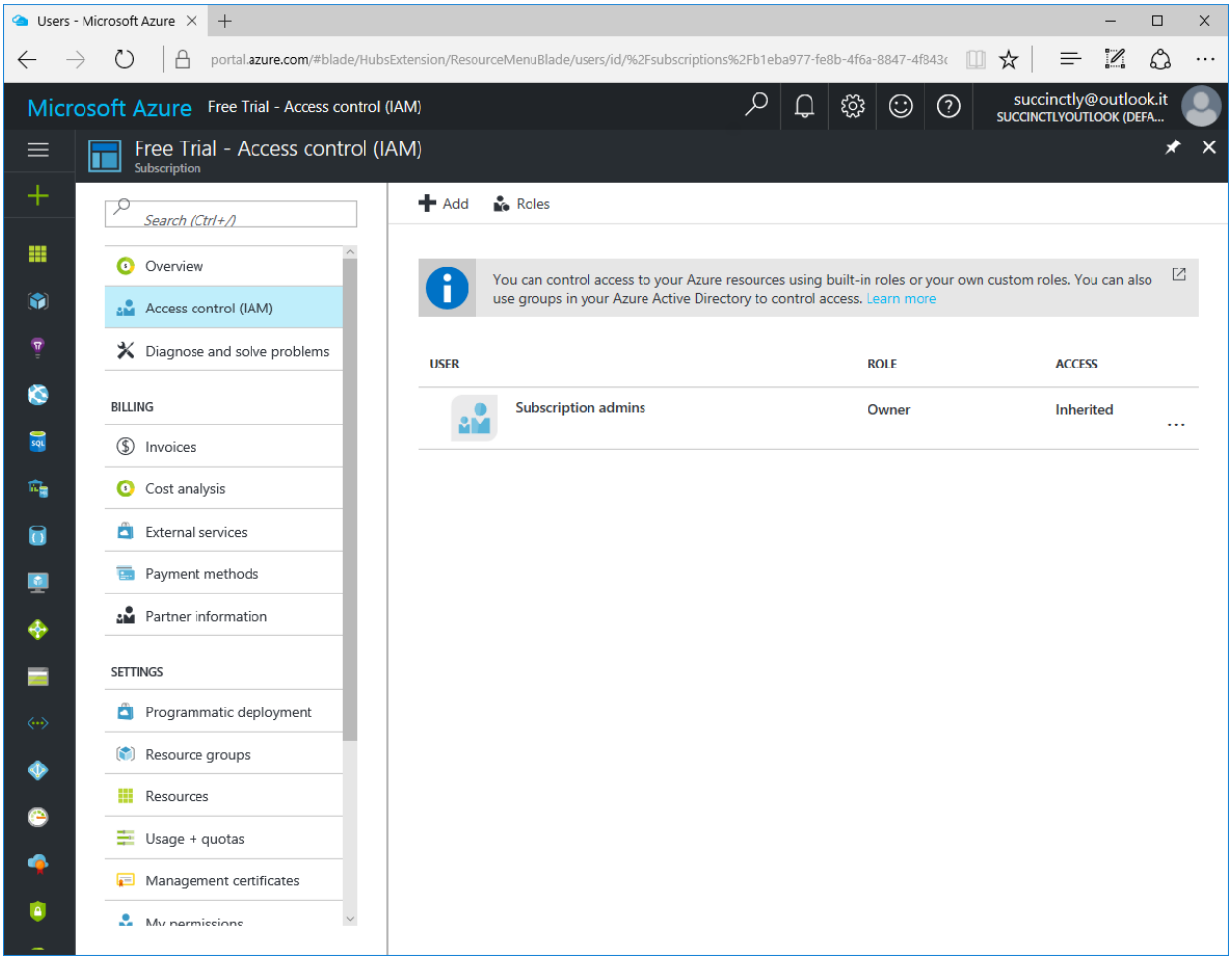


Figure 15: The Access Control (IAM) Feature

When you create a new Application Insights resource, the **Subscription admins** user is defined by default, with the **Owner** role.

www.dbooks.org

By clicking the **Roles** button at the top of the **Access control (IAM)** blade, as seen in Figure 15, we will see the built-in roles for Application Insights and its resources.

This list contains the following roles:

- **Owner**: Can manage everything about Application Insights resources, including access rights.
- **Contributor**: Can manage everything about Application Insights resources, excluding access rights.
- **Reader**: Can view everything about Application Insights resources, but cannot manage anything.
- **Application Insights Component Contributor**: Can manage Application Insights components.
- **Application Insights Snapshot Debugger**: Can use Application Insights Snapshot Debugger features.
- **Log Analytics Contributor**: Can read all monitoring data and edit monitoring settings.
- **Log Analytics Reader**: Can view and search all monitoring data as well as view monitoring settings.
- **Monitoring Contributor**: Can read all monitoring data and update monitoring settings.
- **Monitoring Reader**: Can read all monitoring data.
- **Resource Policy Contributor**: Can create and modify resource policies and create support tickets.
- **User Access Administrator**: Can manage user access to Azure resources.
- **Website Contributor**: Can manage websites, but cannot access them.

To add a user to the access list, you can use the **Add** button or the **Roles** button at the top of the **Access control (IAM)** blade. In both cases, a new blade on the right will open where you can choose the role to assign.

To be sure that the role fits user activities, you can use the **Permissions** button at the top of the **Roles** blade to see the full list of permissions assigned to a role as shown in Figure 16.

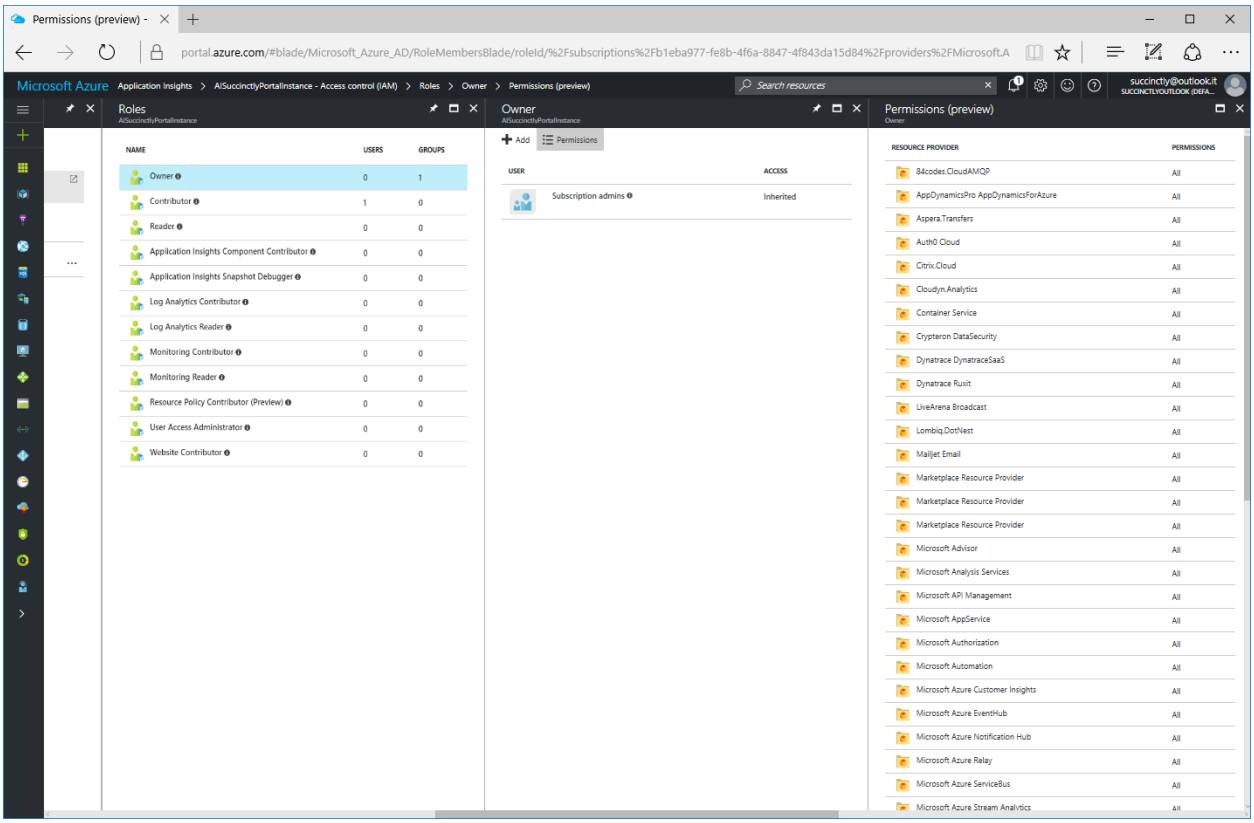


Figure 16: The Permissions List

By selecting a permission from that list, you can see in detail what actions that role can execute as in Figure 17.

www.dbooks.org

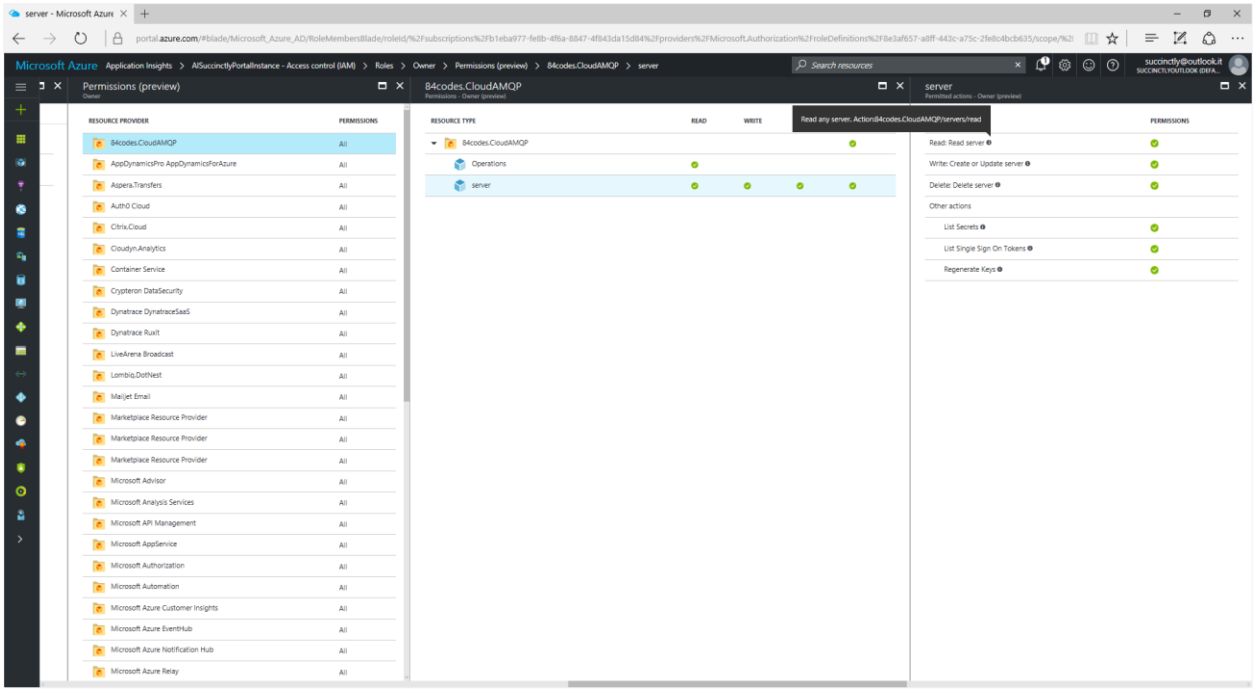


Figure 17: Actions Allowed for a Single Role

In the next chapter, we will see how to define alert rules and warnings to notify users.

Chapter 6 Alert Rules

Every monitoring system has alerting capabilities, using either internal or external subsystems, and Application Insights has its own, too. In this case, it's an internal subsystem.

This alerting subsystem is ready to send a notification or an email warning when a particular event or an interesting scenario happens.

To manage alert rules, we need to go to the **Configure** section and select the **Alerts** feature.

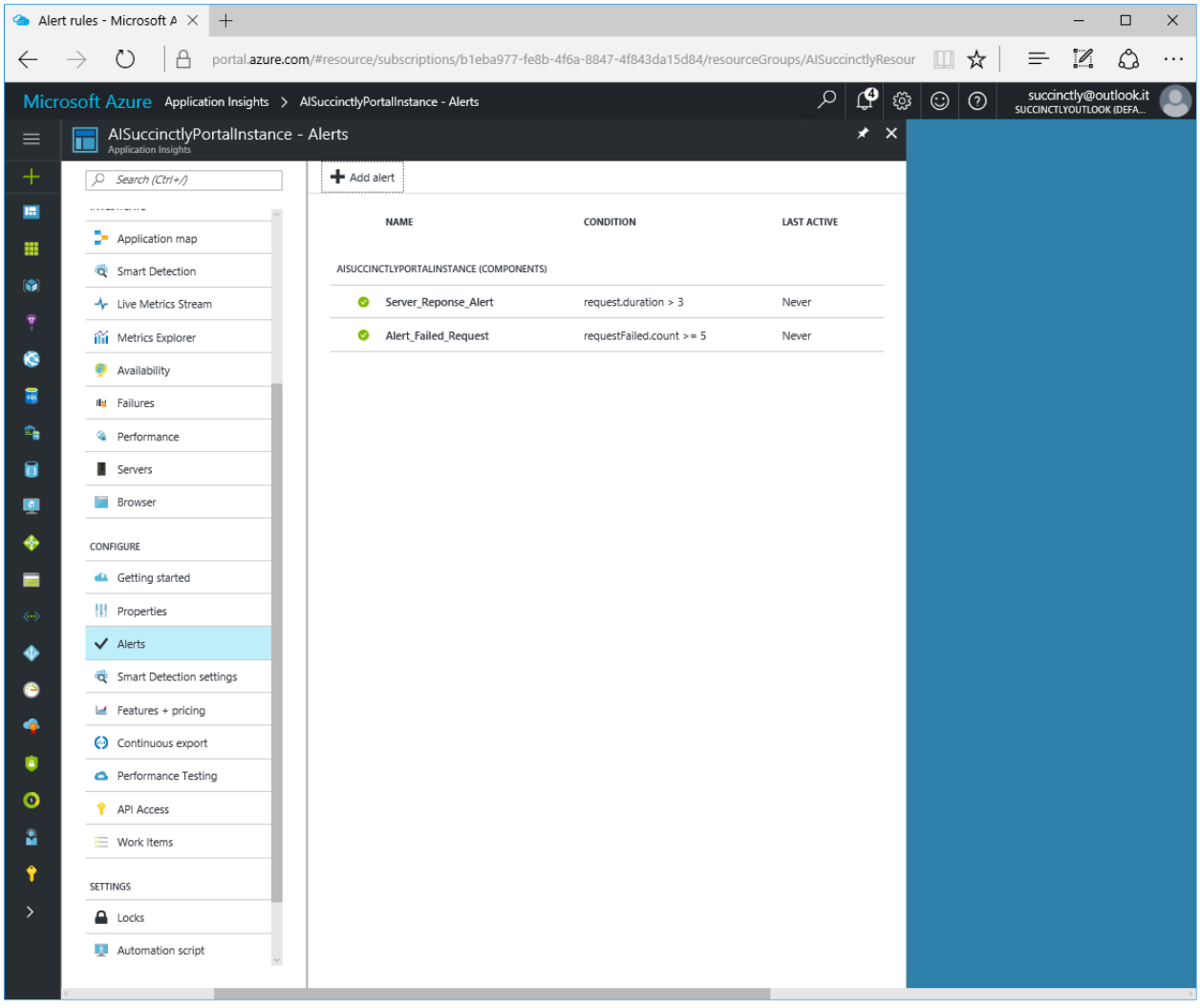


Figure 18: The Alerts Blade

To reach the **Alerts** blade, we can also use the **Alert rules** button within **Metrics Explorer**.

www.dbooks.org

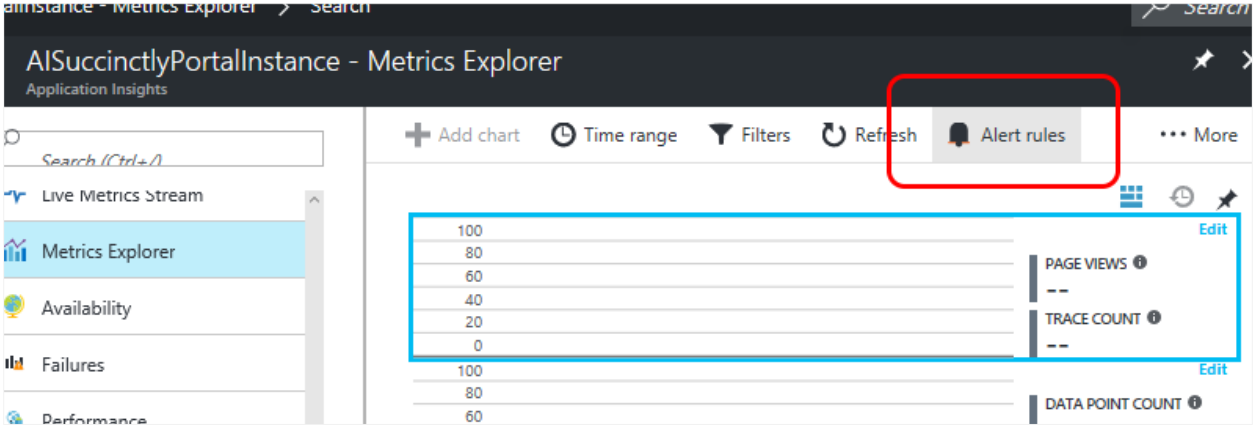


Figure 19: The Alert Rules Button

Once the alert rule matches the condition, an email will be sent to the recipients defined within the alert rule.

When we create a new alert, we must enter some information, such as **Name** and **Description**, and define the scenario we want to monitor by selecting the **Metric**, the **Condition**, the **Threshold**, the reference **Period**, and the destination **Email(s)**.

The **Condition** is a conditional operator (such as *greater than*, *greater than or equal to*, etc.) and the **Threshold** is the comparison parameter.

The reference **Period** is a time span in which this alert can monitor information and, when necessary, send an email. The **Period** cannot be shorter than the last five minutes, and not longer than the last 24 hours.

Recipients for email alerts can be all users—owners, contributors, readers—and other email addresses.

There is also the possibility to use a webhook if we want to route the alert to another system or tool that supports this feature.

Similarly to the Live Metrics Stream, this is not a real-time system.

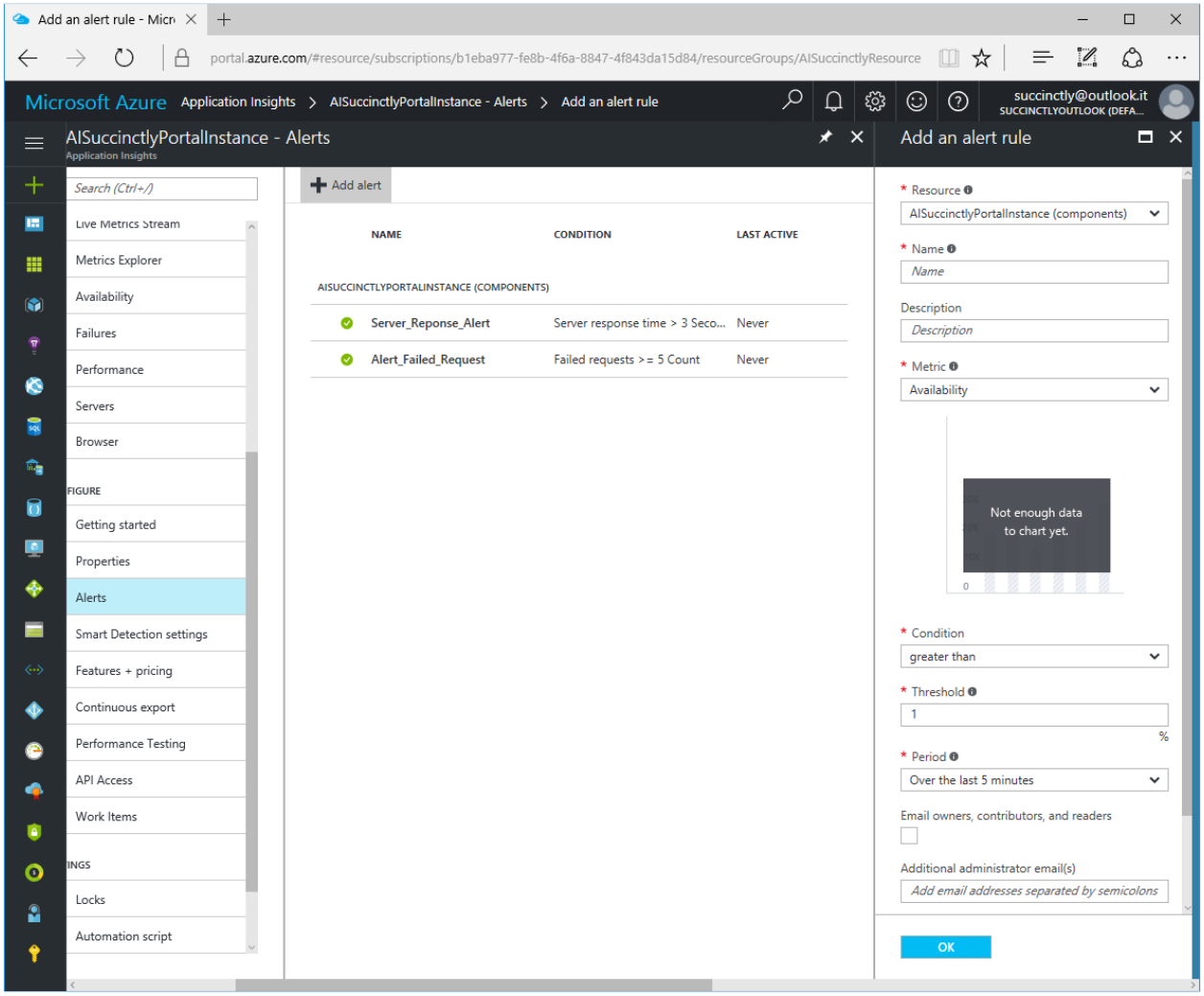


Figure 20: New Alert Creation

Just to give some examples, we can check if the server response time is greater than three seconds within the last five minutes, or if in the last few hours there have been five or more failed requests.

Of course, it is important not to monitor every single metric, but only the ones that are relevant for your business, and set the right parameters for the alerts. Otherwise, you might receive tons of emails.

Also, we can monitor not only for a failure or malfunction, but also for the accomplishment of a goal (for instance, when you reach a million page views on your website).

Now it's time to move the focus to the data. In the next chapter, you will see how to consume and export data from an Application Insights resource.

www.dbooks.org

Chapter 7 Consuming and Exporting Data

If you take a look back at Figure 1, you can see that whatever the pricing option is, the data redemption policy is no longer than 90 days. Therefore, if you want to keep a history of your web application's usage, you need to export the data from the Application Insights resource and store it in another place. Various features help to accomplish this goal, so we should consider all the available options to get the telemetry data out of Application Insights.

Consuming data with Power BI

If you're a Power BI user, you can connect to your Application Insights resource directly with it.

> ***Note: Power BI is a suite of business analytics tools available from Microsoft to analyze data. Using this suite, a company can monitor its business using rich dashboards available on every device. For more information, visit powerbi.microsoft.com or check out Syncfusion's e-book, Power BI Succinctly by Pierstefano Tucci.***

To connect Power BI to Application Insights, you need an account on Power BI.

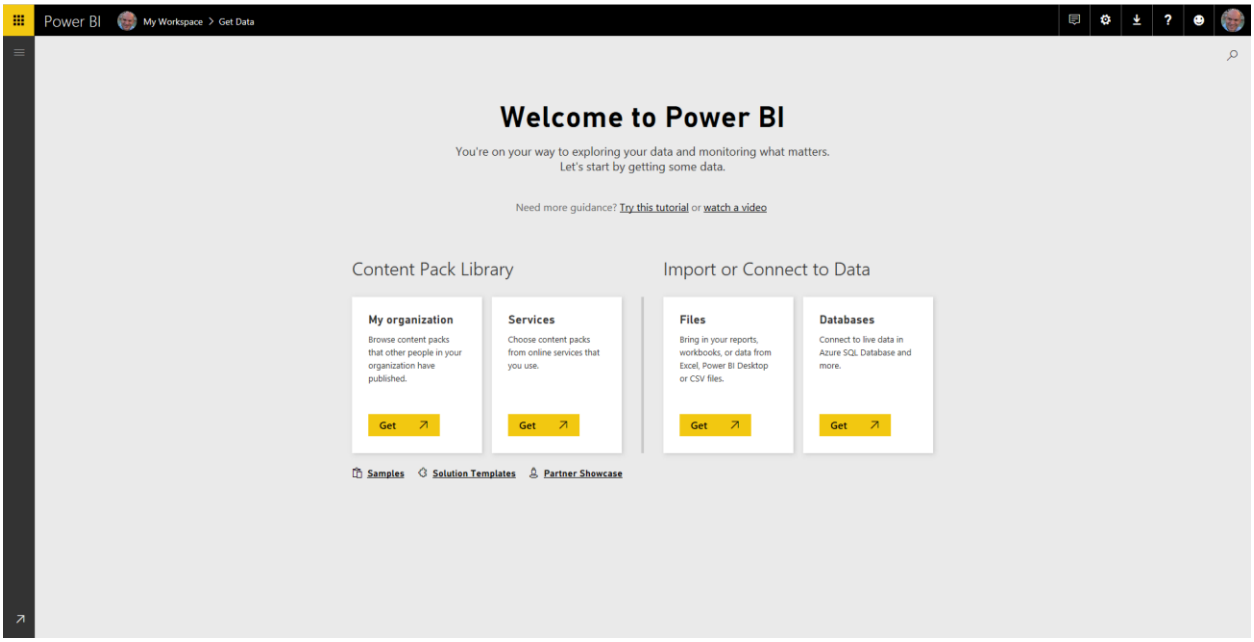


Figure 21: The Power BI Main Page

On the main page, you must use **Services**, and filter out the content packs available to find the right one in order to connect to **Application Insights**.

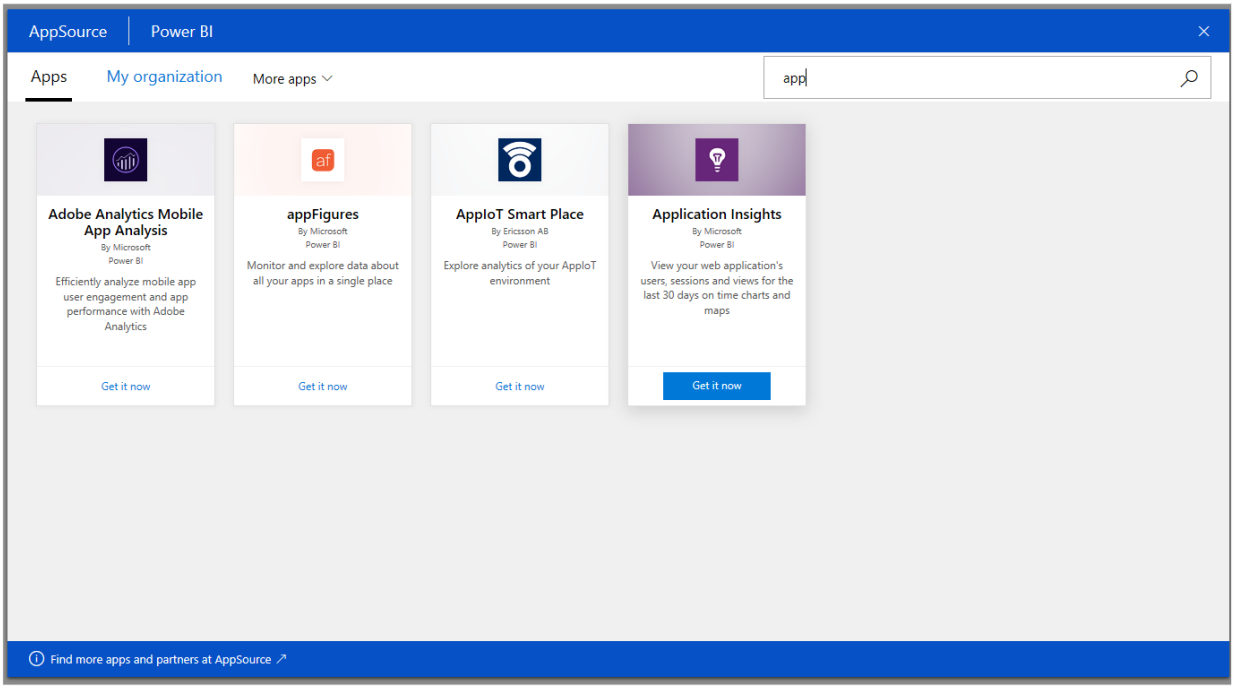


Figure 22: Power BI Content Packs

Once you've selected the right content pack, you need to enter the information required (**Resource Name**, **Resource Group**, and **Azure Subscription ID**) to connect to the Application Insights resource.

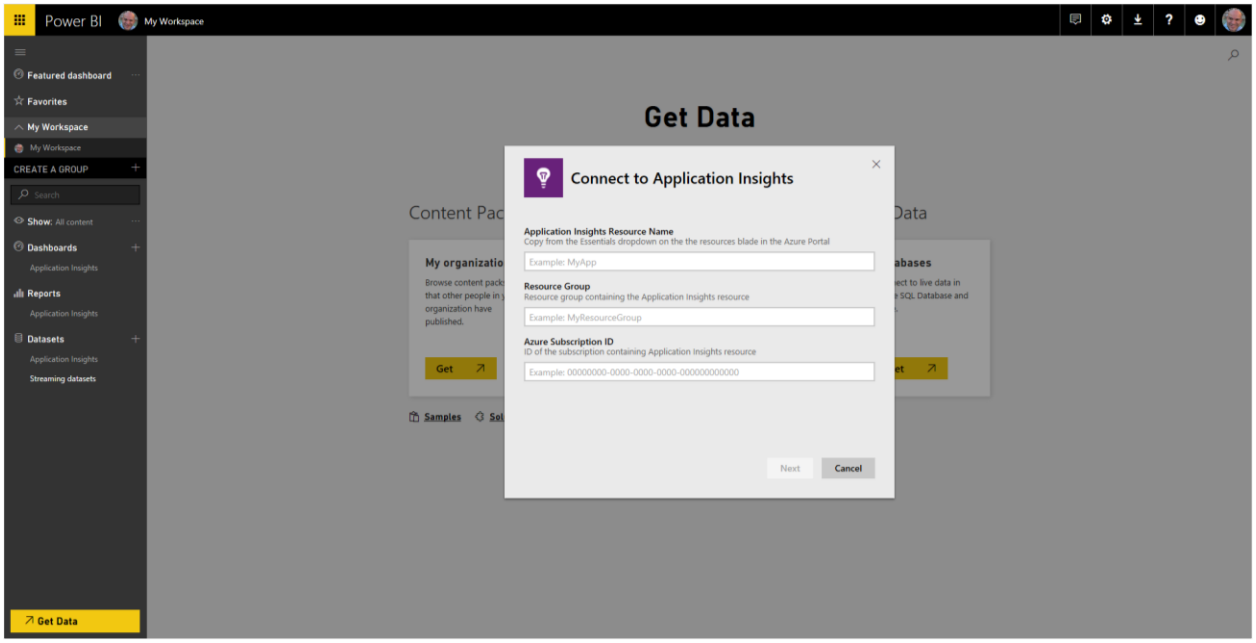


Figure 23: Connecting to the Application Insights Resource

Finally, once you complete the sign-in, you will have access to Application Insights data from Power BI, just as you would with any other source type.

www.dbooks.org

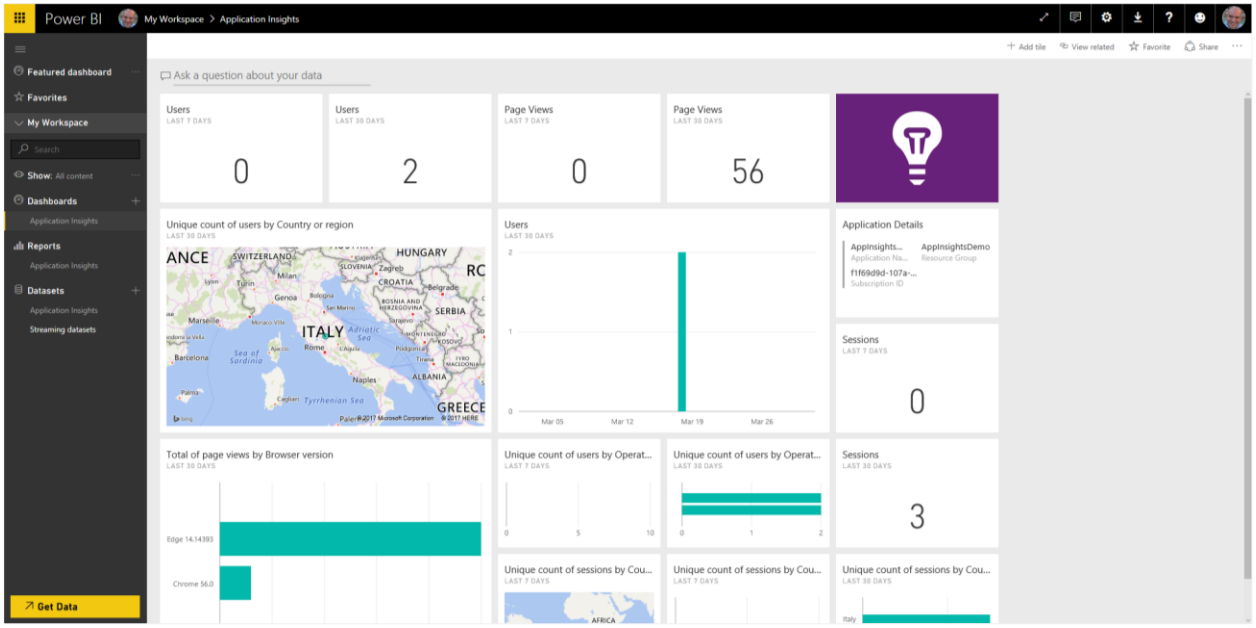


Figure 24: Power BI Connected to an Application Insights Resource

Application Insights REST API

The Application Insights REST API is a programmatic way to get data from Application Insights using a REST API service. It is available here.

> ***Tip: At the time of writing this book, this feature is available in public preview. This means the development is still in progress, so things can change (more or less significantly) on the road to the final release. For information on changes, refer to this changelog.***

This REST API lets you query data about your Application Insights resource.

Before you can use this API, you must configure access to it. To do this, open the **API Access** blade in the **Configure** section to create an API key.

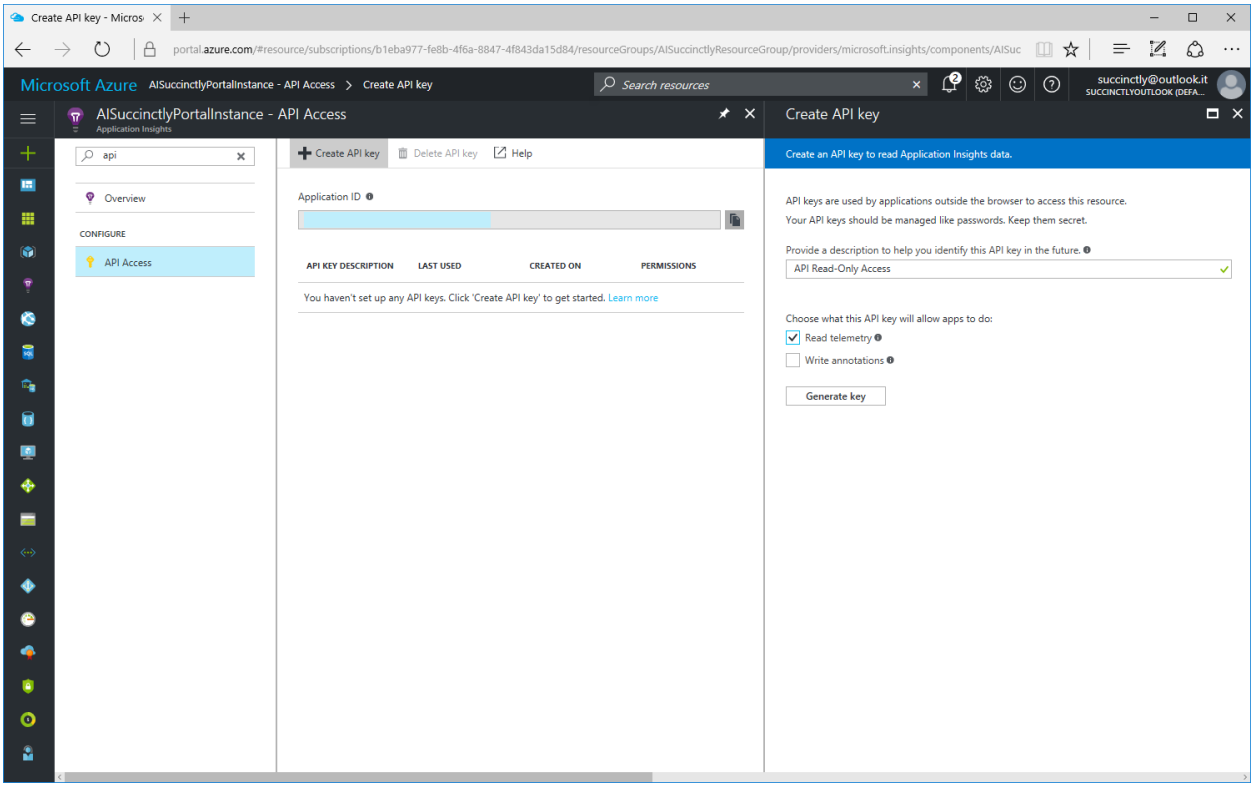


Figure 25: The API Access Blade

When you create access for your data, you can specify a name for your API key, and one or both of the following permissions:

- **Read telemetry**: Allows you to read data from your Application Insights resource.
- **Write annotations**: Allows you to write annotations (not data) in your Application Insights resource. Annotations are additional information about a significant event.

With the **Generate key** button, you will be able to complete the operation and receive an API key with the name and permissions you provided to it.



Tip: You may need multiple API keys. For instance, you can create one key per business application using that data, even if they have the same permission. This way, you can change or revoke rights at any time without affecting the other business applications involved.

Before using this API key (for instance, from within a consumer application), you can try it from the **API Explorer** page.

www.dbooks.org

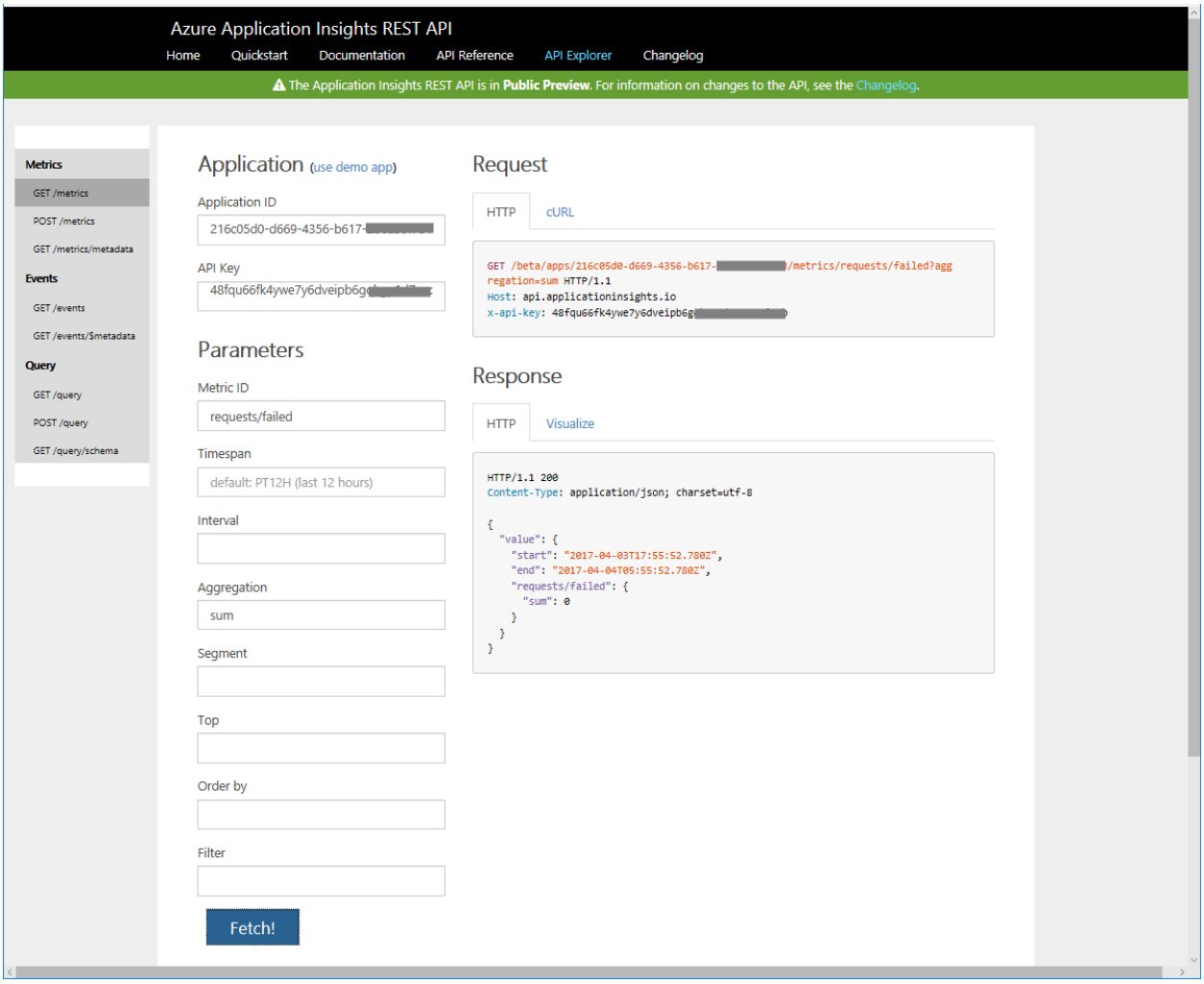


Figure 26: The API Explorer Page

To use this testing tool, you need an **API Key** and **Application ID**. The application ID is available in the **API key** blade in the Azure portal.

Using the **Metric ID** field, you can choose the appropriate one from a list of metrics to get data from the API. On this form, most of the fields are straightforward and easy to understand since you only have to choose one of the suggested items from the list.

The goal behind this feature is to let you see if the data coming back from the API is exactly what you are looking for before you use the API in a business application. Here you can set the different parameters to filter out the result, such as the aggregation type or time span.

As you can see in Figure 26, you can preview both the API request and response.

Once you've tested the appropriate API, you can use it in your business application like any other REST API service.

Continuous export

Since data retention is not unlimited, we should export data from our Application Insights resource at a scheduled time if we want to preserve information.

This capability is available in both the Basic and Enterprise pricing plans, but while there is a fee per gigabyte used in the first (free) option, the second option can be used without any limitation.

To use this feature, select **Continuous Export** from the **Configure** section of the Application Insights features blade. To create a continuous export item, we must configure a few parameters, as you can see in Figure 27.

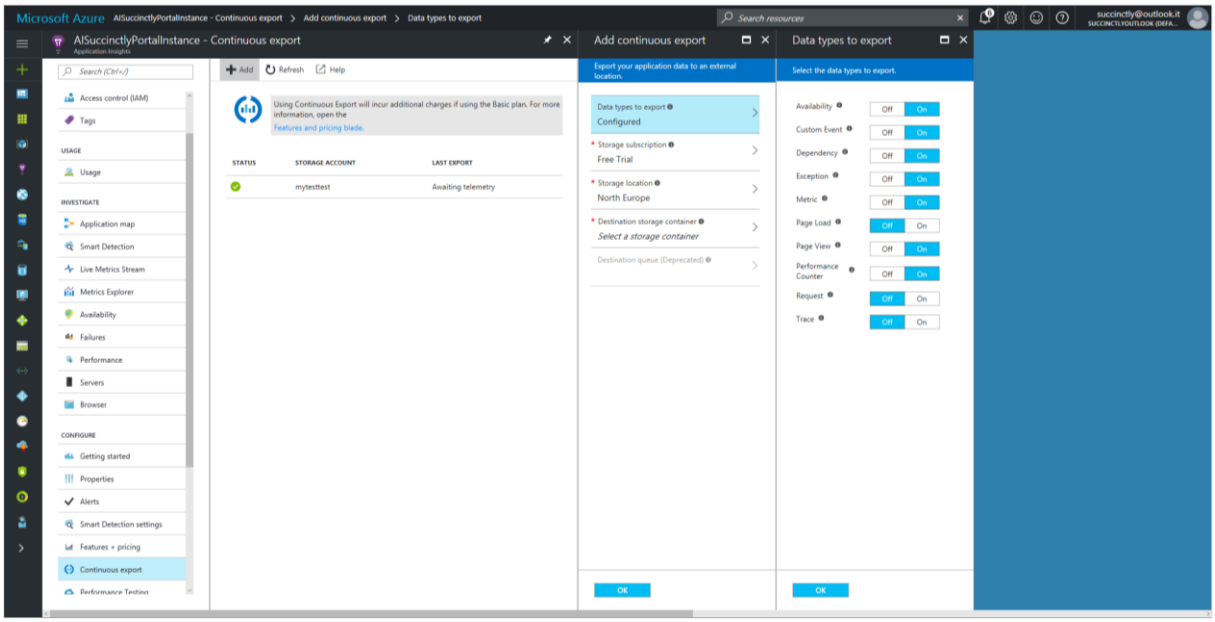


Figure 27: The Continuous Export Feature

The first of these parameters represents the data to export, so we can define the exact kind of information we would like to collect:

- **Availability**: Data results for all the web tests of our application.
- **Custom Event**: All the information registered with the **`TrackEvent()`** method (see Chapter 9).
- **Dependency**: All the time usage data related to external resources (DBs, services, etc.).
- **Exception**: Data results for all server exceptions and exception data registered with the **`TrackException()`** method (see Chapter 9).
- **Metric**: All the information recorded with the **`TrackMetric()`** method (see Chapter 9).
- **Page Load**: The time taken to load a page in the browser.
- **Page View**: The page view count of our web application.
- **Performance Counter**: Performance counters collected by Application Insights Status Monitor.
- **Request**: Information about HTTP requests received by the server.
- **Trace**: Log telemetry and all the data registered with the **`TrackTrace()`** method (see Chapter 9).

www.dbooks.org

Since we need to use an Azure Storage resource (which is the only container type supported), all other parameters are related to the storage in which we want to export the data collected.



Note: We will not cover the creation of storage on Azure, which is beyond the scope of this book. If you'd like to know more about this, refer to this documentation.

- **Storage location** parameter: The value suggested is the same as the Application Insights resource area, and if you change it, you can incur additional charges.

Once we have selected the destination storage, we must choose (or create) a container for it. This feature will work immediately after you create it, while you may stop it by using the **Disable** button on the blade.



Tip: Remember that using continuous export may cause you to incur additional charges; refer to the Pricing options for Application Insights section for further details. Note that Visual Studio MSDN subscribers cannot use this feature.

The exported data will be in JSON format and will represent the raw telemetry that Application Insights receives from the web application.

The exported data could appear in the storage up to an hour after it has been recorded, so **don't consider this feature a real-time export**.

Exported data can stay within the storage for as long as you want, although you'll be paying a fee for its use. Of course, you can export data to your local storage to avoid high storage payments.

The next step is interacting with the Application Insights resource from within your code. In the next chapter, you will see Visual Studio in action.

Chapter 8 Visual Studio and SDKs

It's time to see some code. After all those chapters about configuring and using various features, we can finally see a developer's point of view.

If you are involved with software development that relates to web applications or services, Application Insights has an SDK for almost any programming language or technology, so any developer can easily feel comfortable with it.

Where to find the right SDK

You can find the right SDKs for Application Insights here. As you can see, there are various languages and frameworks supported, including some by third-party vendors. Each item connects you to a specific page where you can begin integrating with Application Insights. This contains useful information on which SDK to use.

In this book, the focus is on Microsoft technologies, so we will explore the process on how to integrate Application Insights into a web application based on ASP.NET MVC, with Visual Studio.

Integrating with Visual Studio

Using Application Insights with Visual Studio is very simple due to the high integration between both products and technologies.

In the following examples, we will use Visual Studio Enterprise 2017. Any other edition (such as the free Community edition) would work fine as well. There are some minor differences when using Visual Studio 2015, but the process is almost the same in either version.

We will start by creating a new ASP.NET MVC project from within Visual Studio.

www.dbooks.org

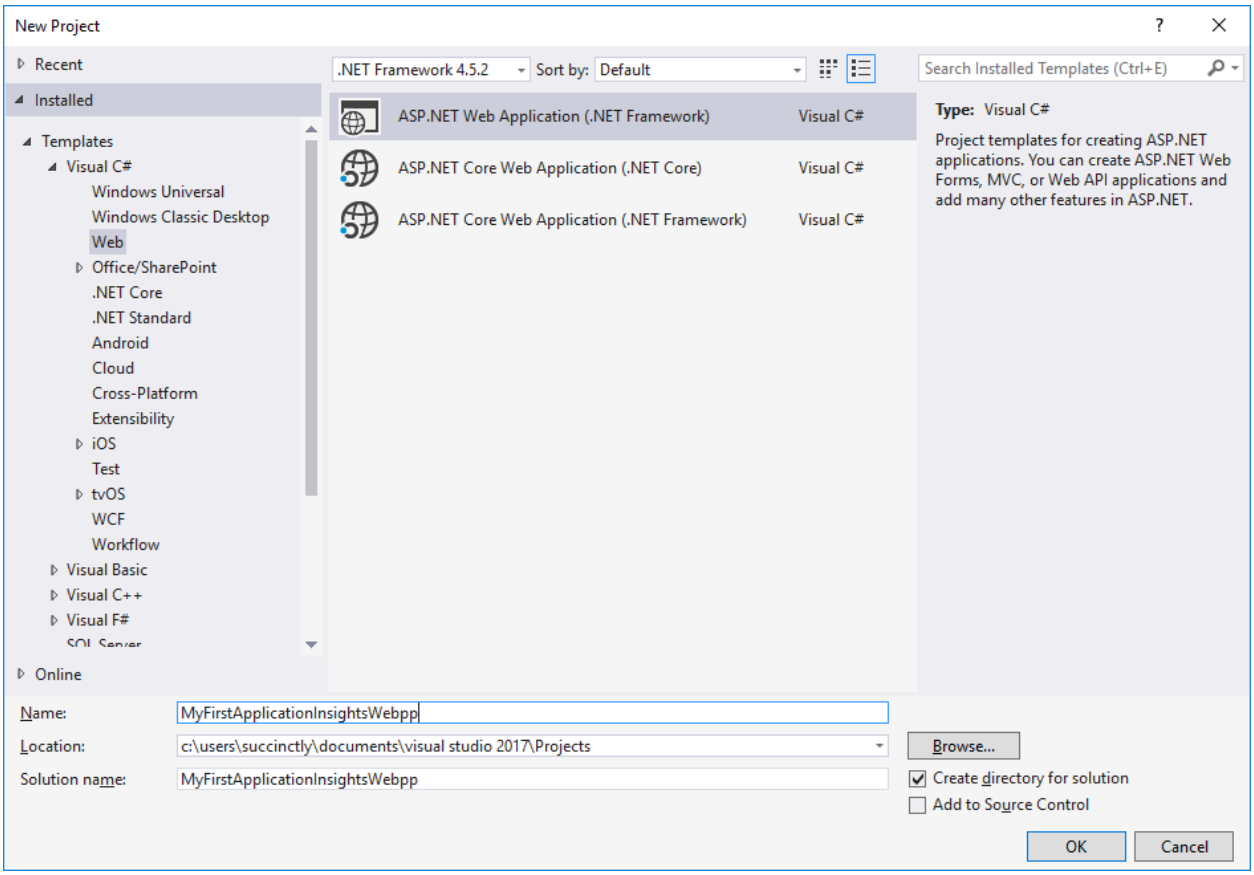


Figure 28: Starting a New Project

For this particular example, choose ASP.NET MVC with no authentication.

Once the project is created, we can add Application Insights by right-clicking on the project in the **Solution Explorer** window and choosing **Configure Application Insights** from the context menu.

A page will appear, and from there you will be able to choose the subscription and other information to register your app with Application Insights. At the bottom of this page, there is the option to use a free account.

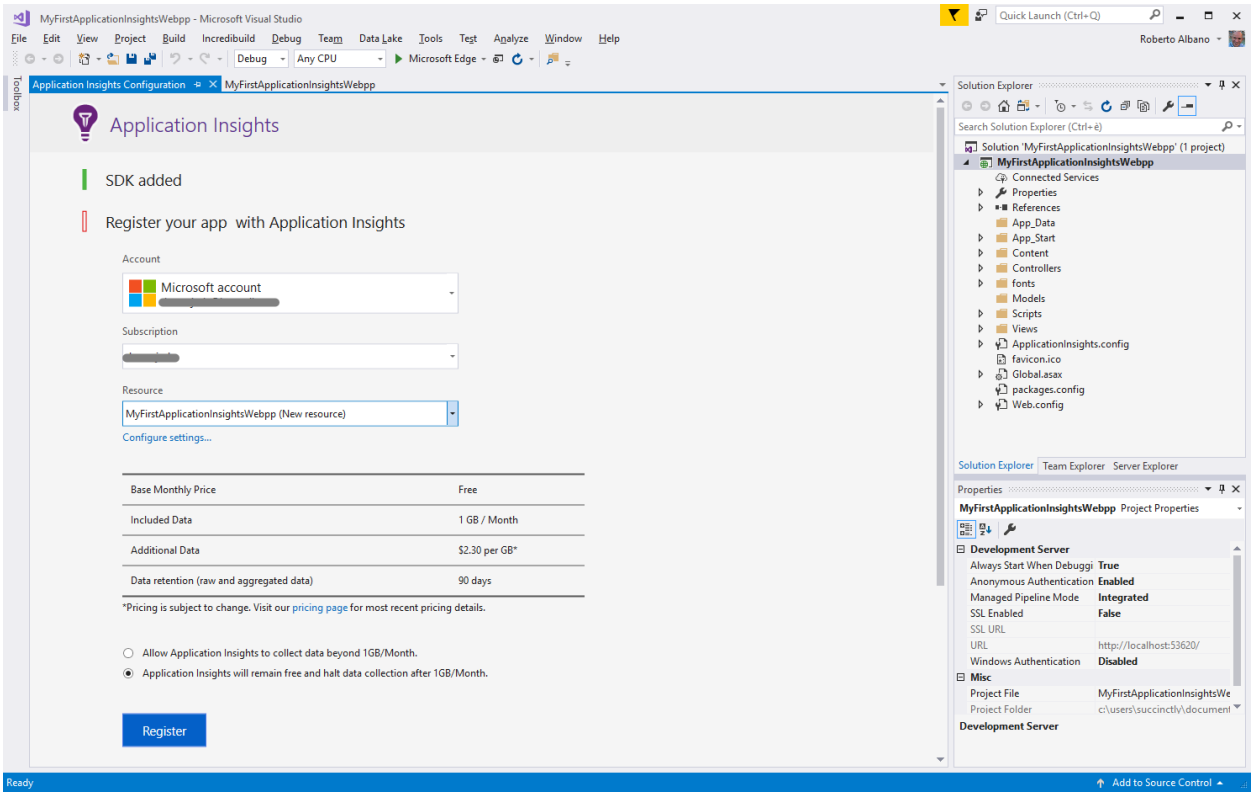


Figure 29: The Web Application Registration

Once you have activated an Application Insights resource, every call you execute from your code using one of the following lines will be collected into Application Insights and will be visible from the Dashboard on the Azure portal:

```
System.Diagnostics.Trace.TraceInformation("Information");
System.Diagnostics.Trace.TraceWarning("Warning");
System.Diagnostics.Trace.TraceError("Error");
```

After a few moments, the web application will be registered to use telemetry locally in Visual Studio instead of using a brand-new Application Insights resource. You can return to this page to activate an Application Insights resource on the Azure portal at any point.

What happened to my project?

If you followed the previous steps to register Application Insights, you'll see some changes in the project. The references that you will see in the following sections are related to a project in Visual Studio. If you use Application Insights from within other languages or IDEs, the behavior could be different.

References

First, we need to add the required Application Insights assemblies to our project.

www.dbooks.org

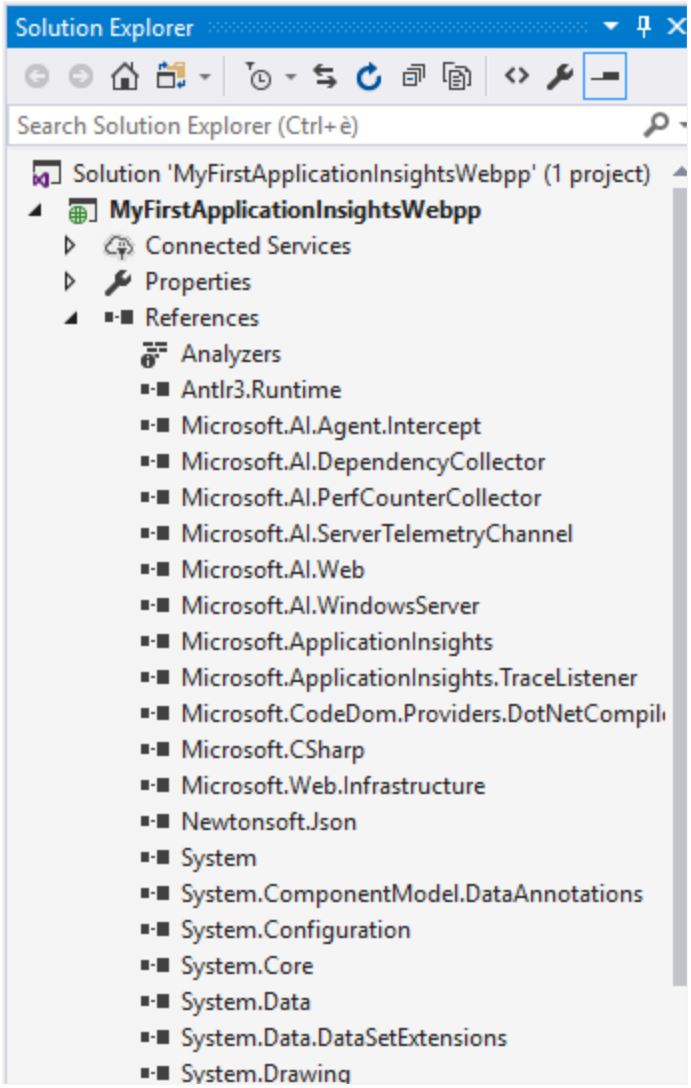


Figure 30: Project References

These references have been added behind the scenes by NuGet. It has added the references **Microsoft.AI.*.dll** and **Microsoft.ApplicationInsights*.dll** to the project.

Connected services

In the project, a new file will be added to the **Connected Services** section. This will be the **ConnectedService.json** configuration file.

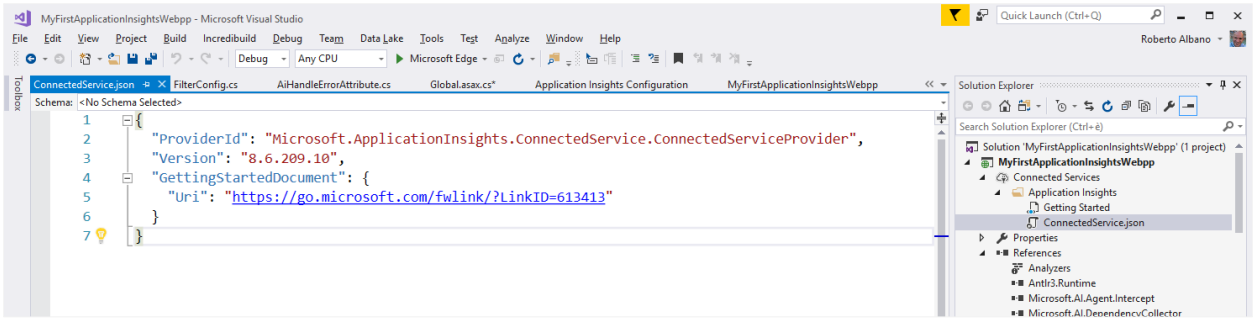


Figure 31: The Connected Service Configuration File

Configuration

A new **ApplicationInsights.config** file will appear in our project.

This is an XML file that holds all the information related to the resource used in the project. Here we will find the instrumentation key (which we talked about in the section The instrumentation key) that uniquely identifies the resource within the whole project.



Figure 32: The ApplicationInsights.config File

Apart from this key, there are other parameters such as modules and assemblies to consider within the telemetry data collection process.

www.dbooks.org

Another part of the configuration is placed inside the **Web.config** file. In this XML file, you can find a couple of elements that register the Application Insights modules used to track every exception that occurs.

Code Listing 1: Changes to the Web.config File

```
<system.web>
    ...omitted...
    <httpModules>
      <add name="ApplicationInsightsWebTracking"
           type="Microsoft.ApplicationInsights.Web.
                 ApplicationInsightsHttpModule, Microsoft.AI.Web" />
    </httpModules>
  </system.web>
    ... omitted...
  <system.webServer>
    ... omitted...
    <modules>
      <remove name="ApplicationInsightsWebTracking" />
      <add name="ApplicationInsightsWebTracking"
           type="Microsoft.ApplicationInsights.Web.
                 ApplicationInsightsHttpModule, Microsoft.AI.Web"
           preCondition="managedHandler" />
    </modules>
</system.webServer>
```

These parameters are important for error handling with Application Insights.

Scripts

Since there is a lot of information we can gather in a repetitive way (for instance, page visits and origin of requests), Visual Studio helps us retrieve this information directly from the master page of our project by placing inside this view a JavaScript snippet that is called directly from the Application Insights SDK.

In our example, we are using an ASP.NET MVC web application, and the snippet can be found inside the **_Layout.cshtml** view (which is sort of a master page for this kind of web application).



Figure 33: The Application Insights JavaScript Snippet

Placing the snippet here allows us to track all the visits to every page that uses this layout, without the annoyance of having to paste this code into every view.

> ***Tip: When working regularly with ASP.NET MVC, I usually strip off this snippet from its original location and place it into a partial view. This strategy will allow you to reuse this code more efficiently when there are more layout views within your project.***

As you can see, the snippet refers to the **`InstrumentationKey`** that is used within the code as a copy of the key registered within the **Web.config** file. We will explore another way to do this in the Using different instrumentation keys for various environments section.

Error handling

One of the files that Visual Studio automatically adds to our web application project for us is the **AiHandleErrorAttribute.cs** file (contained within the **ErrorHandler** folder).

This file defines the following class.

Code Listing 2: The AiHandleErrorAttribute Class

```
[AttributeUsage(AttributeTargets.Class | AttributeTargets.Method,
Inherited = true, AllowMultiple = true)]
public class AiHandleErrorAttribute : HandleErrorAttribute
{
    public override void OnException(ExceptionContext filterContext)
    {
        if (filterContext != null
            && filterContext.HttpContext != null
            && filterContext.Exception != null)
        {
```


www.dbooks.org

```
            // If customError is Off,
            // then AI HTTPModule will report the exception
            if (filterContext.HttpContext.IsCustomErrorEnabled)
            {
                var ai = new TelemetryClient();
                ai.TrackException(filterContext.Exception);
            }
        }
        base.OnException(filterContext);
    }
}
```

This class handles exceptions and inherits from the **`HandleErrorAttribute`** class. It can be used to decorate other classes or methods in which we can potentially catch an exception and send it to our Application Insights resource.

However, if we search for the **FilterConfig.cs** file (in the **App_Start** folder) we can see that this class is used in the **`RegisterGlobalFilters`** method, whose purpose is to register the filters to use for incoming requests.

This means an instance of this class is present in the stack of filters that the ASP.NET MVC application will use for every request, so when an exception is thrown, this class will register it with Application Insights for us.



Tip: Be careful that the exception isn't trapped when using this class if the** custom error **setting in the Web.config file is turned off. In that case, the exception will be held by the Application Insights HTTP module.

In Code Listing 2, we can see the **`TelemetryClient`** class for the first time. We will discuss it in detail in the next chapter.

Let's go ahead and start writing code from a web application and record data into Application Insights.

Chapter 9 Writing Your Code

In the last chapter, we demonstrated how Visual Studio creates a project that is ready to be used as an Application Insights resource. Now it's time to gather the points we want to keep track of.

As we have seen, with Application Insights we can register anything that happens in our web application. You can do this programmatically through code, using the available SDKs for most web development languages and frameworks.

In every SDK, there is a standard entry point to register something on our Application Insights resource, and this is the **`TelemetryClient`** class.

The TelemetryClient class

To start using the **`TelemetryClient`** class, we only need a few lines of code. In the following example, we track an exception.

Code Listing 1: Tracking an Exception Using a TelemetryClient Instance

```
var ai = new TelemetryClient();
ai.TrackException(ex);
```

In this example, the first line of code creates a new instance of the **`TelemetryClient`** class. It establishes a connection to the Application Insights resource linked by the instrumentation key used.

In the next line, the **`TrackException`** method is invoked, passing an exception instance.

With these two lines of code, we can send the entire exception to the Application Insights resource. There we will find all the information related to it, including the origin of the exception, the stack trace, and so on.

As you can see, this example is written in C#, and so it can be executed from the *server side*. The interesting thing to know is that we can have the **`TelemetryClient`** class even on the *client side*, using the same method, with the same input and output provided.

Tracking methods

Table 7 contains the methods of the **`TelemetryClient`** class. Some of these methods are invoked directly using the JavaScript snippet (see the Scripts section).

www.dbooks.org

Table 7: TelemetryClient Methods

Method	Description
TrackAvailability	Sends an **AvailabilityTelemetry** object with information about the availability of an application
TrackDependency	Sends a **DependencyTelemetry** object with information about external dependencies called in an application
TrackEvent	Sends an **EventTelemetry** object for display in Metrics Explorer
TrackException	Sends an **ExceptionTelemetry** object for display in Diagnostic Search
TrackMetric	Sends a **MetricTelemetry** object for aggregation in Metrics Explorer
TrackPageView	Sends a **PageViewTelemetry** object with information about the page viewed in the application
TrackRequest	Sends a **RequestTelemetry** object with information about a request handled by the application
TrackTrace	Sends a **TraceTelemetry** object with a trace message for display in Diagnostic Search

Every method sends an instance of a particular class to the Application Insights resource. The instance sent has all the information related to the event or metric to enable tracking. All these methods are overloaded to send the original information as parameters instead of the instance.

To give an example, suppose you have the following variables.

Code Listing 4: Defining Parameters for a RequestTelemetry Instance

```
var ai = new TelemetryClient();
string name = "myRequest";
DateTimeOffset startTime = new DateTimeOffset(DateTime.Now);
TimeSpan duration = new TimeSpan(0,0,0,0,250);
string responseCode = "200";
bool success = true;
```

You can track a request by sending an instance of the **RequestTelemetry** class.

Code Listing 5: Tracking a Request Using TrackRequest with a RequestTelemetry Instance

```
RequestTelemetry requestTelemetryInstance = new RequestTelemetry(name,
startTime, duration, responseCode, success);
ai.TrackRequest(requestTelemetryInstance);
```

Alternatively, this can be done by sending the values as parameters using an overload of the constructor.

Code Listing 6: Tracking a Request Using TrackRequest with Parameters

```
ai.TrackRequest(name, startTime, duration, responseCode, success);
```

The TelemetryConfiguration class

`TelemetryConfiguration` is another important class to understand. It holds all the information related to the resource of Application Insights that will use the web application.

As stated previously, the instrumentation key is stored in the **ApplicationInsights.config** file.

You can create an instance of the **`TelemetryClient`** class using the default constructor (without any input parameters), or use an overload of it.

In both constructors, the class uses an instance of **`TelemetryConfiguration`** to connect to the right Application Insights resource.

If you use the default constructor for **`TelemetryClient`**, it implies the usage of a singleton instance of the **`TelemetryConfiguration`** class. If you require more than one instance, you must manage your own **`TelemetryConfiguration`** instance.

For example, say we create an instance of **`TelemetryClient`** using the default constructor.

Code Listing 7: TelemetryClient Instantiation Using a Default Constructor

```
var ai = new TelemetryClient();
```

Behind the scenes, this class will use the following singleton instance of **`TelemetryConfiguration`**:

Code Listing 8: TelemetryConfiguration Singleton Instance

```
TelemetryConfiguration.Active
```

This uses the instrumentation key present in the ApplicationInsights.config file.

However, since **`InstrumentationKey`** is a property of this class, we can change this value to connect to a different key. We will discuss this in the next section and see how we can manage different instrumentation keys depending on different scenarios.

Returning to the **`TelemetryClient`** class, another constructor accepts an instance of a **`TelemetryConfiguration`** class as an input parameter.

www.dbooks.org

Code Listing 9: TelemetryClient Instantiation Using an Overloaded Constructor

```
TelemetryConfiguration config = TelemetryConfiguration.CreateDefault();
config.InstrumentationKey = "myInstrumentationKey";
var ai = new TelemetryClient(config);
```

By using this constructor, you have the responsibility to set the correct references for the resource in the configuration class.



Tip: In the previous listing, we used the `CreateDefault` initializer (that creates an instance starting from the *ApplicationInsights.config* parameters) due to the long list of properties to initialize by hand using a default constructor.

Using different instrumentation keys for various environments

If you are working on a web application and want to use Application Insights resources to monitor both the debug and the release version of your project, it is recommended that you use two different resources to avoid influencing the production environment with data coming from the development environment.

You can achieve this with little effort and only a few lines of code.

Creating and configuring resources

Let's start creating the Application Insights resources. Of course, we will create two different resources; in this example, we'll identify these as **ai_dev_resource** for development and **ai_rel_resource** for release.



Note: We'll create this example using an ASP.NET MVC web application and Visual Studio. For different languages and IDEs, there might be an equivalent way.

In this type of application, we have a **Web.Config** file with all the information about the web application configuration.

Using the solution configurations feature within Visual Studio, we'll have a debug and a release configuration created by default (but we can have more solution configurations if needed).

For every *solution configuration*, Visual Studio generates a corresponding *configuration file,* so having debug and release configurations will result in two XLST files in the Solution Explorer: **Web.Debug.config** and **Web.Release.config**. This allows us to transform the **Web.config** file with the correct parameters for each environment.

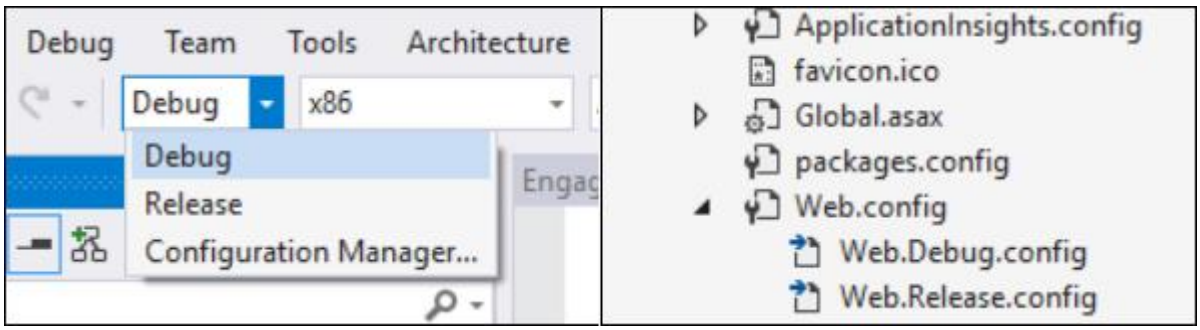


Figure 34: Solution Configurations

For this example, suppose we have the following instrumentation keys:

- **ai_dev_resource** => a11111111-a111-a111-a111-a11111111111
- **ai_prod_resource** => b22222222-b222-b222-b222-b22222222222

First, in your **Web.config** file, you need to add the development instrumentation key in an **`appSettings`** key:

Code Listing 10: Instrumentation Key Setting in the Web.config File

```
<appSettings>
  <!-- Application Insights -DEVELOPMENT- Instrumentation Key -->
  <add key="ApplicationInsightsInstrumentationKey" value=" a11111111-
   a111-a111-a111-a11111111111" />
</appSettings>
```

Then you must add the other instrumentation key in the **Web.Release.config** file, with an XML transformation rule, as follows.

Code Listing 11: Instrumentation Key Transformation Setting for Release (Production)

```
<appSettings>
  <!-- Application Insights -PRODUCTION- Instrumentation Key -->
  <add key="ApplicationInsightsInstrumentationKey" value=" b22222222-
   b222-b222-b222-b22222222222" xdt:Transform="SetAttributes"
   xdt:Locator="Match(name)"/>
</appSettings>
```

Using this configuration replacement, you ensure that when you deploy your code in a production environment (by using the release solution configuration), the instrumentation key will have the correct value.

Remember that these settings **will not automatically change** the instrumentation key at run time. It is only a mode to guarantee that, in every context, the correct key will target the right destination environment. It must be loaded by code, as we will see in the next section.

www.dbooks.org

Assigning instrumentation keys at run time

Once these configurations have been added, we need to understand how to assign the instrumentation key at run time.

Since the Application Insights SDK is very flexible, there is a possibility to change the value of the current resource's instrumentation key at run time.

To do this, use the **Global.asax** file and add two lines of code at the end of the **`Application_Start`** method of the class inside this file.

Code Listing 12: Customizing Application Insights' Resource Loading

```
public class MvcApplication : System.Web.HttpApplication
{
    protected void Application_Start()
    {
        AreaRegistration.RegisterAllAreas();
        FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
        RouteConfig.RegisterRoutes(RouteTable.Routes);
        BundleConfig.RegisterBundles(BundleTable.Bundles);

        // Load the instrumentation key from the configuration file
        var currentTelemetryKey System.Web.Configuration.
            WebConfigurationManager.
            AppSettings["ApplicationInsightsInstrumentationKey"];
        // Assign the instrumentation key to the running resource
        Microsoft.ApplicationInsights.
            Extensibility.TelemetryConfiguration.Active.
            InstrumentationKey = currentTelemetryKey;
    }
}
```

The last line of code in this method will assign the value of the instrumentation key by reading it from the **Web.config** file; therefore, if you are in a production environment and using the Release solution configuration, you should have the **`InstrumentationKey`** value of the **ai_prod_resource** resource.

There is another area where you need to make a small change: the JavaScript snippet we talked about in the Scripts section.

In this snippet, instead of using the **`InstrumentationKey`** as a fixed value, we will load it using the Razor syntax with the variable we mentioned that holds the key.

```
Microsoft.ApplicationInsights.Extensibility.TelemetryConfiguration.Active.
InstrumentationKey
```

The result is something like this:

Code Listing 13: JavaScript Snippet for Page View Tracking

```
<script type="text/javascript">
    var appInsights=window.appInsights||function(config){
        function r(config){t[config]=function(){var
i=arguments;t.queue.push(function(){t[config].apply(t,i)})}}var
t={config:config},u=document,e=window,o="script",s=u.createElement(o),i,f;f
or(s.src=config.url||"//az416426.vo.msecnd.net/scripts/a/ai.0.js",u.getElem
entsByTagName(o)[0].parentNode.appendChild(s),t.cookie=u.cookie,t.queue=[],
i=["Event","Exception","Metric","PageView","Trace"];i.length;)r("track"+i.p
op());return
r("setAuthenticatedUserContext"),r("clearAuthenticatedUserContext"),config.
disableExceptionTracking||(i="onerror",r("_"+i),f=e[i],e[i]=function(config
,r,u,e,o){var s=f&&f(config,r,u,e,o);return
s!==!0&&t["_"+i](config,r,u,e,o),s}),t
    }({
        instrumentationKey:
"@(Microsoft.ApplicationInsights.Extensibility.TelemetryConfiguration.Activ
e.InstrumentationKey)"
    });
    window.appInsights=appInsights;
    appInsights.trackPageView();
</script>
```

Using this code, you will ensure that every HTML page will use the right instrumentation key to track page views.

From now on, you can debug your code using the **ai_dev_resource**, and without influencing the production environment, which will be monitored by the **ai_prod_resource**.

www.dbooks.org

Chapter 10 Application Insights Analytics

Application Insights Analytics is a powerful tool that enables users to search across large amounts of data that can then be collected within an Application Insights resource. It uses a specific language to query terabytes of data in seconds.

In the following sections, we will be exploring this tool in more detail and obtain greater understanding of this querying language.

This tool is available from the Azure portal by selecting the **Overview** page of the Application Insights resource and searching for the Application Insights **Analytics** icon.

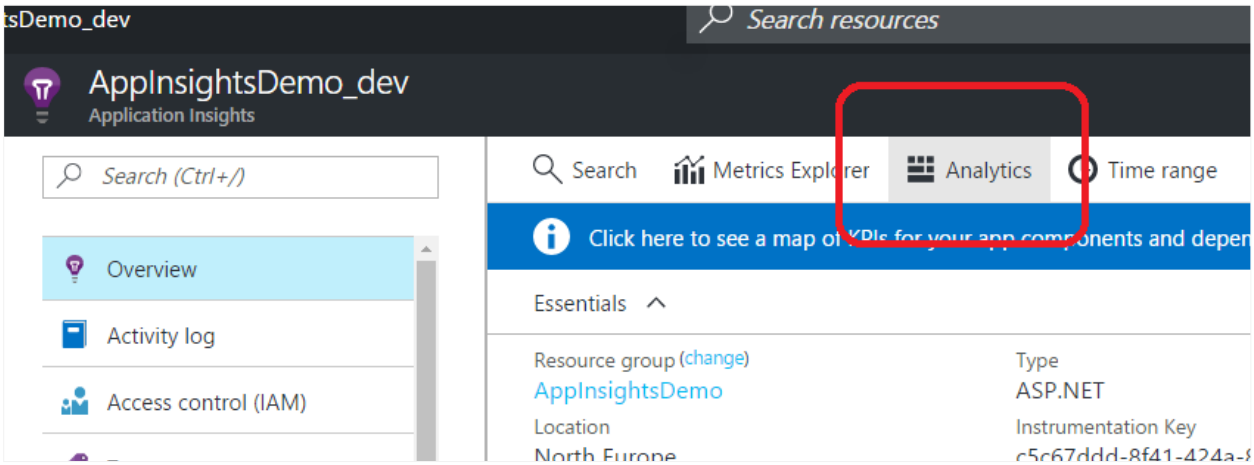


Figure 35: The Application Insights Analytics Button

Using this button, you will land on a separate portal, focused solely on Application Insights Analytics, as shown in Figure 36.

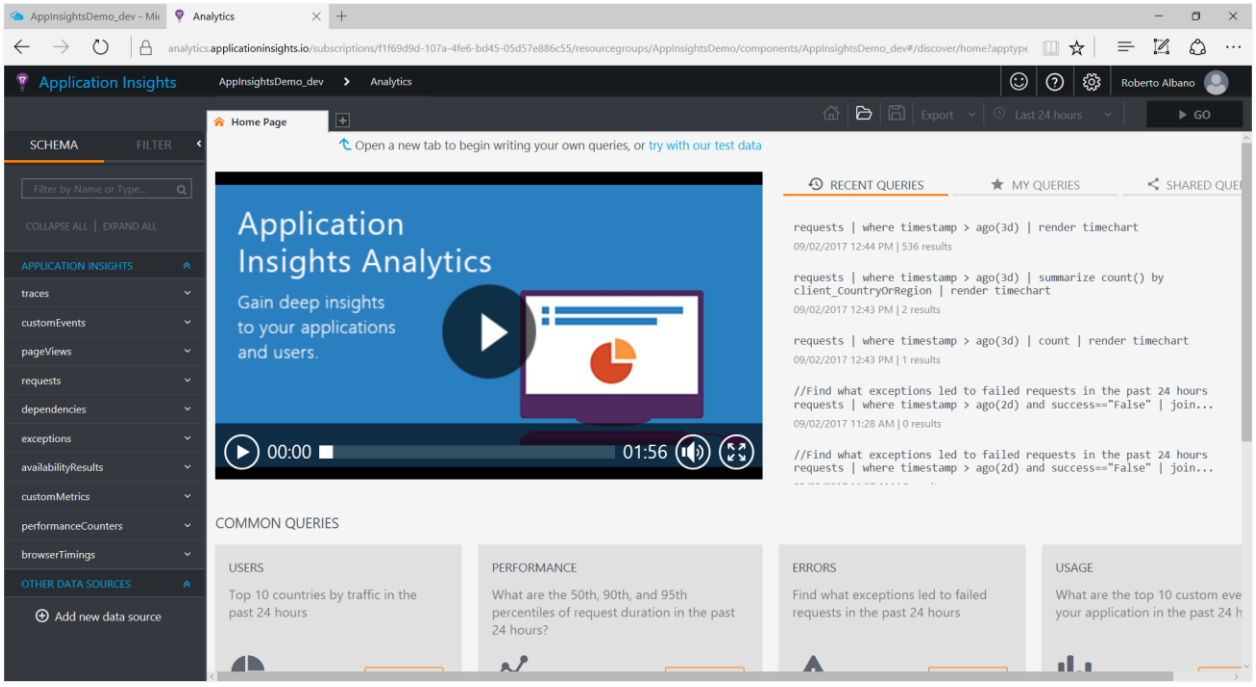


Figure 36: The Application Insights Analytics Portal

Here you will be able to query data from your telemetry and export results.

On this page, we can find a section with common queries for *users*, *performance*, *errors*, and *usage* topics. These are useful samples to start exploring the query language.

By understanding the basics of the query language, you will be able to retrieve more data than by using the available dashboard features. Before we dig deeper into the query language, we will preview some of the main sections that you will find on the query page.

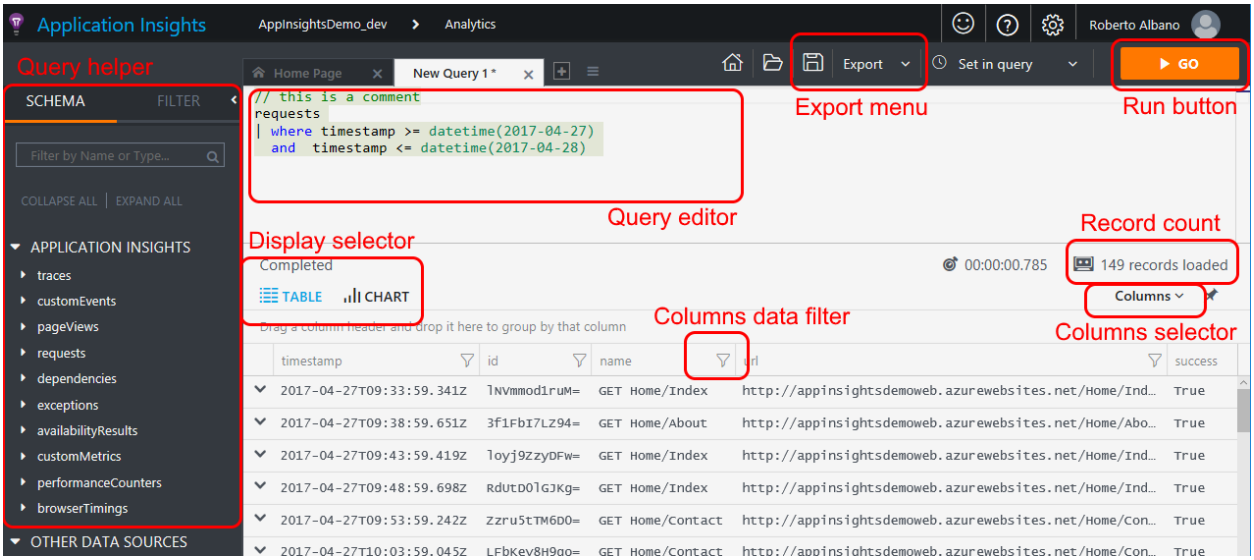


Figure 37: A Preview of the Query Page

www.dbooks.org

Querying the data

Let's start with a simple example of a query. Suppose you would like to get information on the last five requests received from your web application.

You need to query your Application Insights resource to get back the last five HTTP requests registered in your telemetry.

The syntax is as follows:

Code Listing 14: A Simple Query to Return the Last Five Requests

```
requests | limit 5
```

In simple terms, all the telemetry data is stored in a big database. The first line of this command indicates the table from which we want to retrieve the data (in this case, the **request** table). Then, separated by the pipe character (**|**), are the filter clauses we want to apply on that table to get the proper data from the query. You can use one or more filters, but every query needs to start with the pipe character.

Note: It is not mandatory to write a filter clause per row to execute the query, but writing one per row improves readability and performance.

To run the query, click the orange **GO** button in the top-right corner of the page. Figure 38 displays the results.

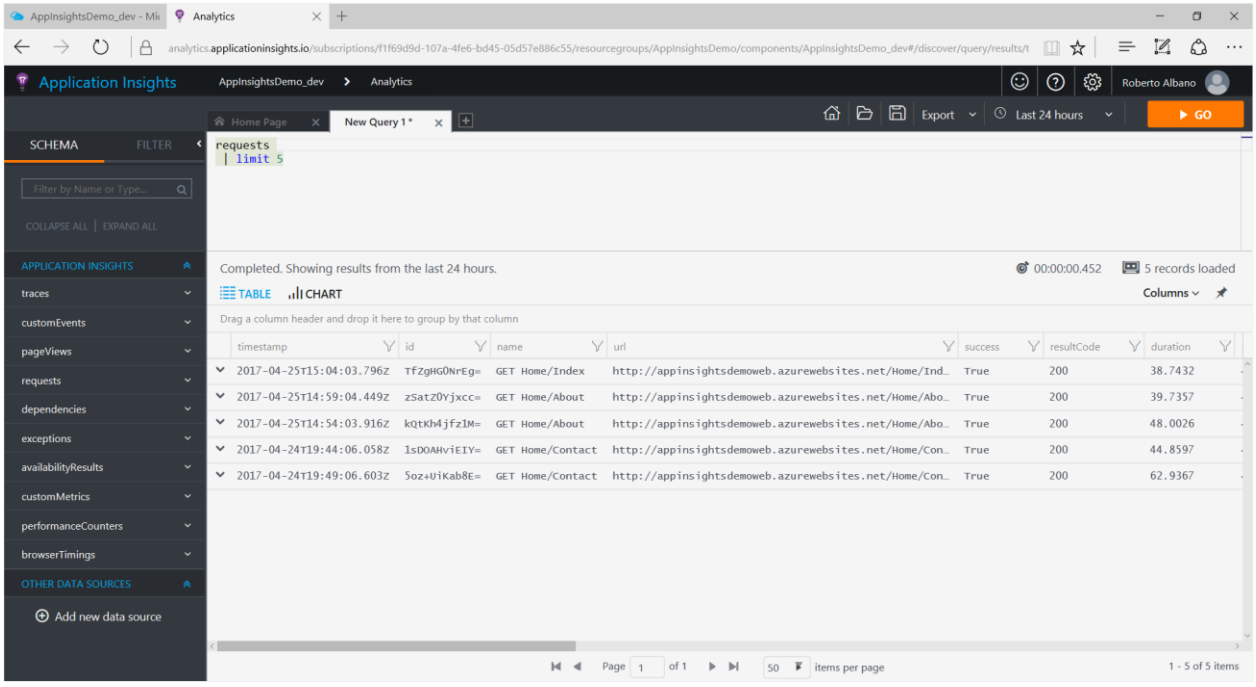


Figure 38: An Elementary Example of a Query

As you can see, the page shows the results for the query (if it is correct and there is data to display). The grid displays the records found. The first column of every line contains a chevron icon; when you click this icon, the row will expand to show all the details for that record.

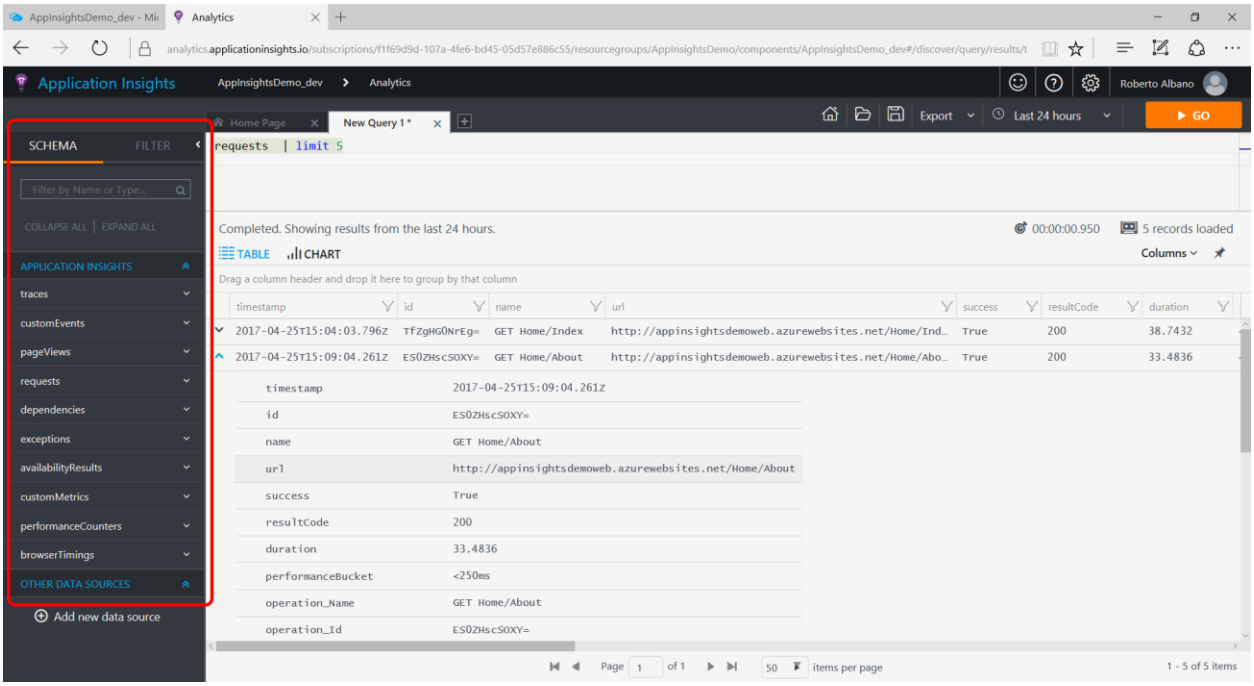


Figure 39: Record Details

It is possible to retrieve a lot of data within seconds, no matter the date range you choose. Remember that the retaining period for data is 90 days.

As you can see in Figure 39, on the left of the query editor, there is a toolbox with all the entities you can query that are available in the schema. You can expand each one to see the list of data points (or fields, if you think of your data as a database) that you can find within the entity.

You can put any item from this toolbox directly into the query editor by double-clicking on it instead of typing it into the editor. This toolbox is useful for formulating standardized commands and provides a good starting point for understanding data points per entity.

If you want to add more filters or, more generally, need to build your statement, the autocompletion feature can help you. You can see this feature in action when writing code in the query editor.

www.dbooks.org

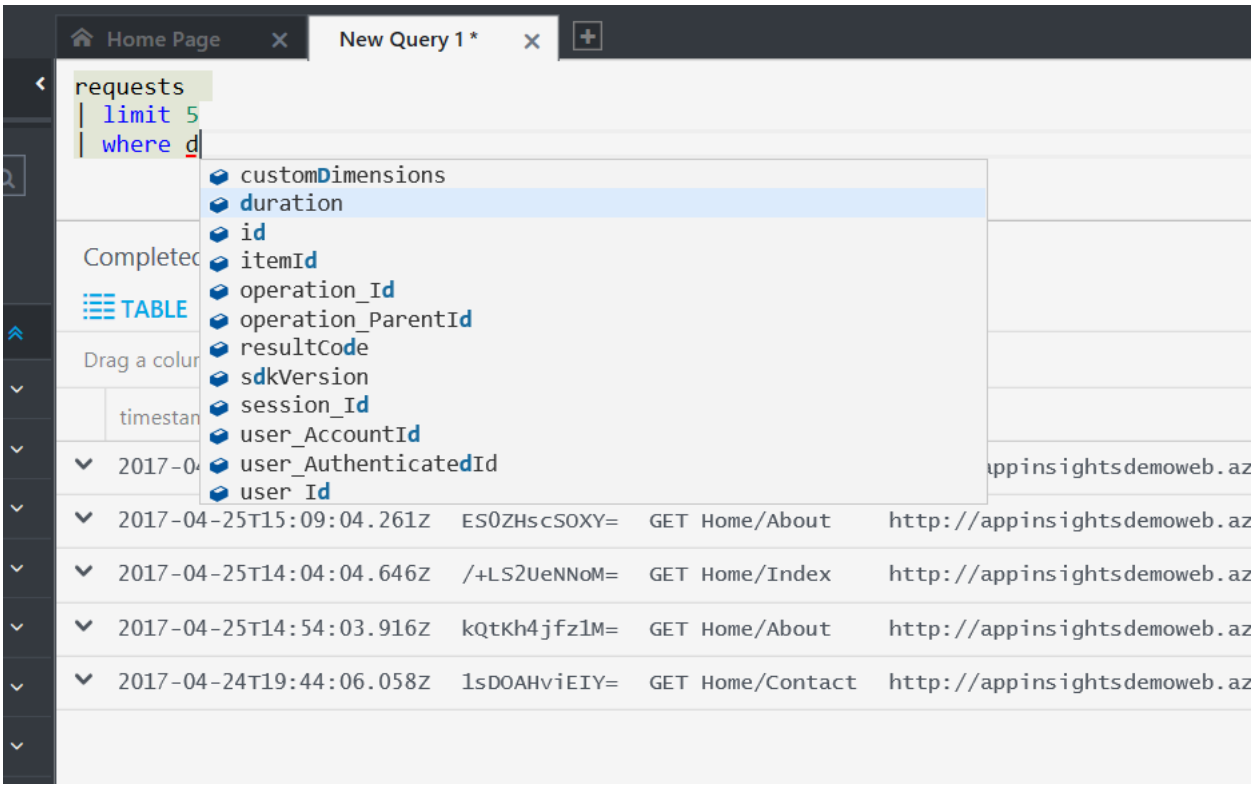


Figure 40: Query Autocompletion

Columns selected and columns displayed

When you execute a query, you will retrieve all the information available on the table you are querying. But if you want to specify the fields that you want to receive in your result, you should use the **`project`** keyword. For instance, if you want to receive from the query only two fields, such as **`timestamp`** and **`url`**, you can write the following query.

Code Listing 15: A Query Returning Only Two Fields

```
requests | limit 5 | project timestamp, url
```

The keyword **`project`** allows you to fetch only the fields you specify in the query. These will be displayed as columns, and you will see that in your grid.

In addition, if you want to display only a subset of the columns returned from the query, you can use the selector available on the top, right-hand side of the grid.

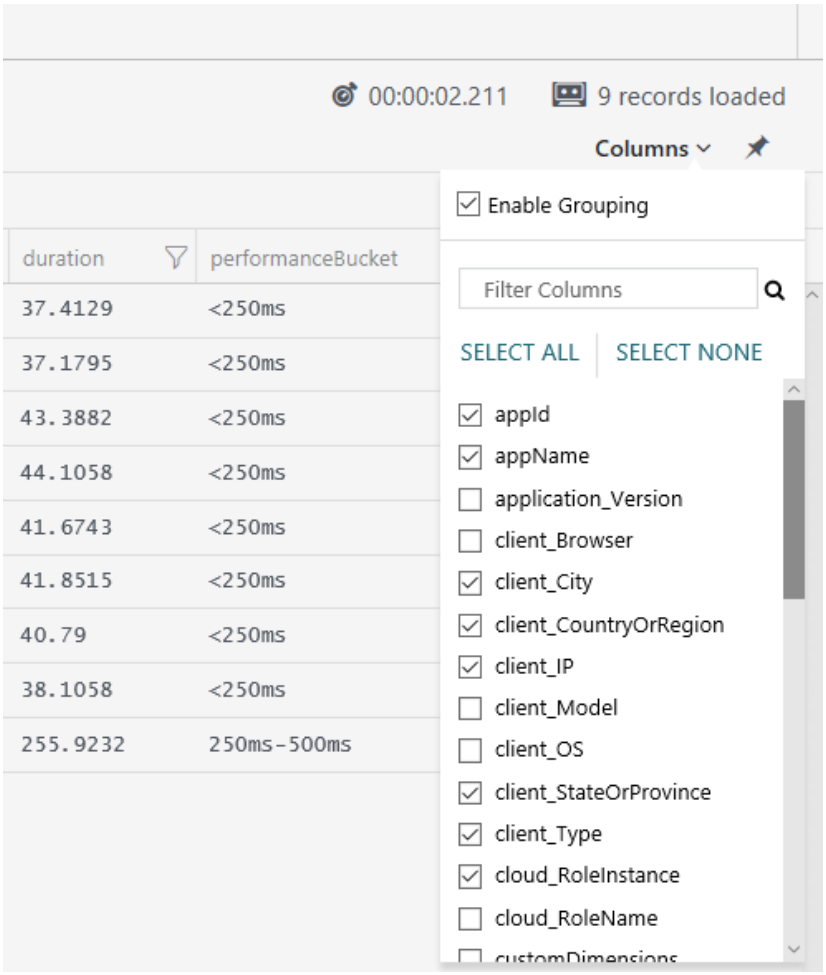


Figure 41: The Columns Selector

You can use both the **`project`** keyword and the **Columns** selector; however, the selector will only display the columns returned from the query. Columns that have been unselected are still in the result even if not displayed, and will be there if you export the whole query result.

You can filter the data in the grid as well, using the filter symbol in the header of every column. Every filter will open a little pop-up window for entering the filter value, as shown in the following figure.

www.dbooks.org

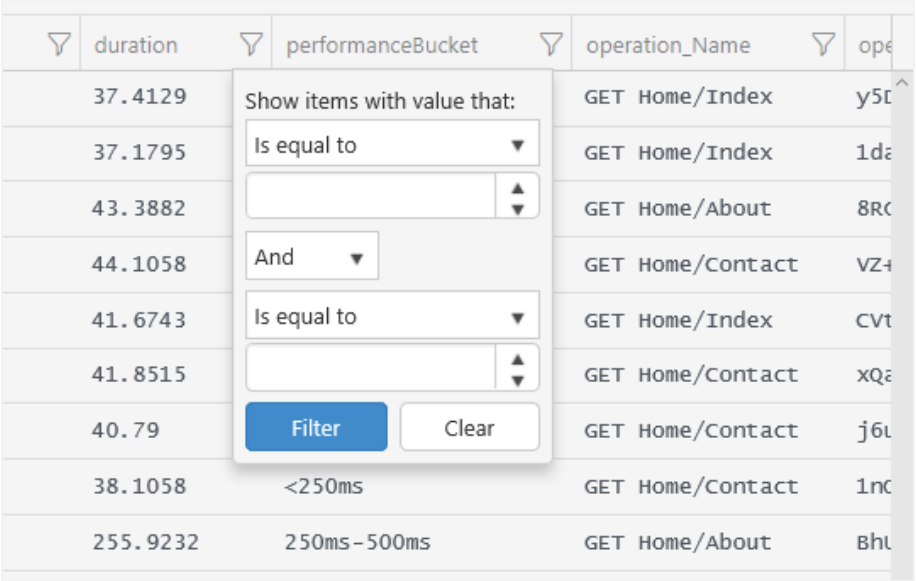


Figure 42: The Column Data Filter

This image is familiar for users who frequently manage data through a spreadsheet application such as Excel, and using such an application enables users to apply detailed filters to the columns to drill down into the results.

Date and time expressions

In almost every kind of metric, there is a timestamp related to the moment in which the event happened. And, of course, it is also possible to apply a filter on this timestamp.

For the time filter statement, there are some rules to know in order to be more productive when building your query. For instance, if you want to get the requests **for the last 24 hours**, you can write a query as follows.

Code Listing 16: A Query that Returns the Last Day's Requests

```
requests | where timestamp >= ago(24h)
```

If you want to get the requests starting from a specific date, you can write the following query.

Code Listing 17: A Query that Returns Requests from a Specific Date

```
requests | where timestamp >= datetime(2017-01-01)
```

You can be more precise in your filter by using a specific time range.

Code Listing 18: A Query that Returns Requests from a Specific Time Range

```
requests | where timestamp >= datetime(2017-01-01T00:00:00.000) and
timestamp <= datetime(2017-01-31T23:59:59.999)
```

There are quite a few time filters that can be used in your query; they are listed here alphabetically:

- **ago**
- **datepart**
- **dayofmonth**
- **dayofweek**
- **dayofyear**
- **endofday**
- **endofmonth**
- **endofweek**
- **endofyear**
- **getyear**
- **now**
- **startofday**
- **startofmonth**
- **startofweek**
- **startofyear**
- **todatetime**
- **totimespan**
- **weekofyear**

Managing the results

When you run a query, the results will be returned in a grid, as seen in Figure 38. It is a flat visualization that is useful for reading your data in a simple way.

But you might need to save your data, understand it with a different visualization, or perform more analysis on the results. In that case, you need to export that data or show it in a form other than a grid.

Exporting the grid

If you want to save your data, you can export it using the integrated **Export** menu in the top, right-hand side of the page.

www.dbooks.org

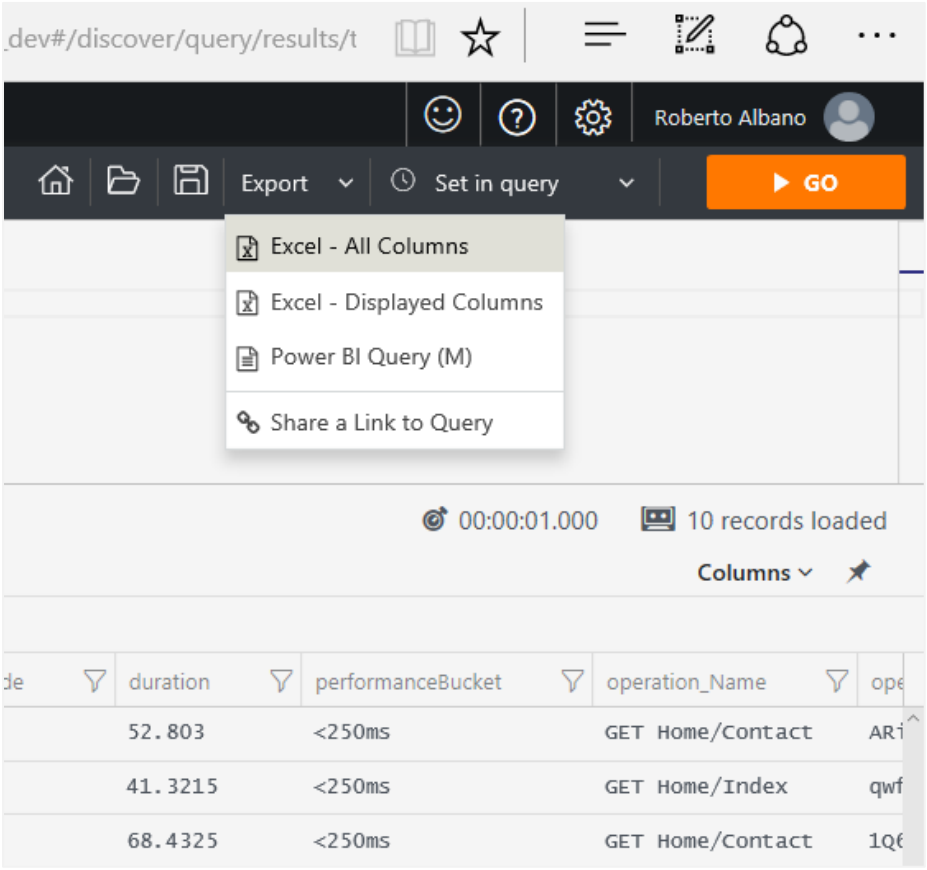


Figure 43: The Export Menu

You can choose to export it to an Excel file, using all the columns returned by the query, or only the displayed columns by choosing **Excel - All Columns** or **Excel - Displayed Columns**. The exported file will be saved with a .csv extension, so you can open it in Excel or a text editor, in case you don't have Excel installed.

From the same menu, you can also export using the **Power BI Query (M)** option (meaning a query in the M language, or the Power Query M formula language), so it can be used in Excel or with the Power BI Desktop application.

The exported file is saved with the .txt extension. It includes instructions on how to use Power Query within the Power BI Desktop app and get the results from the Analytics query directly into the application.

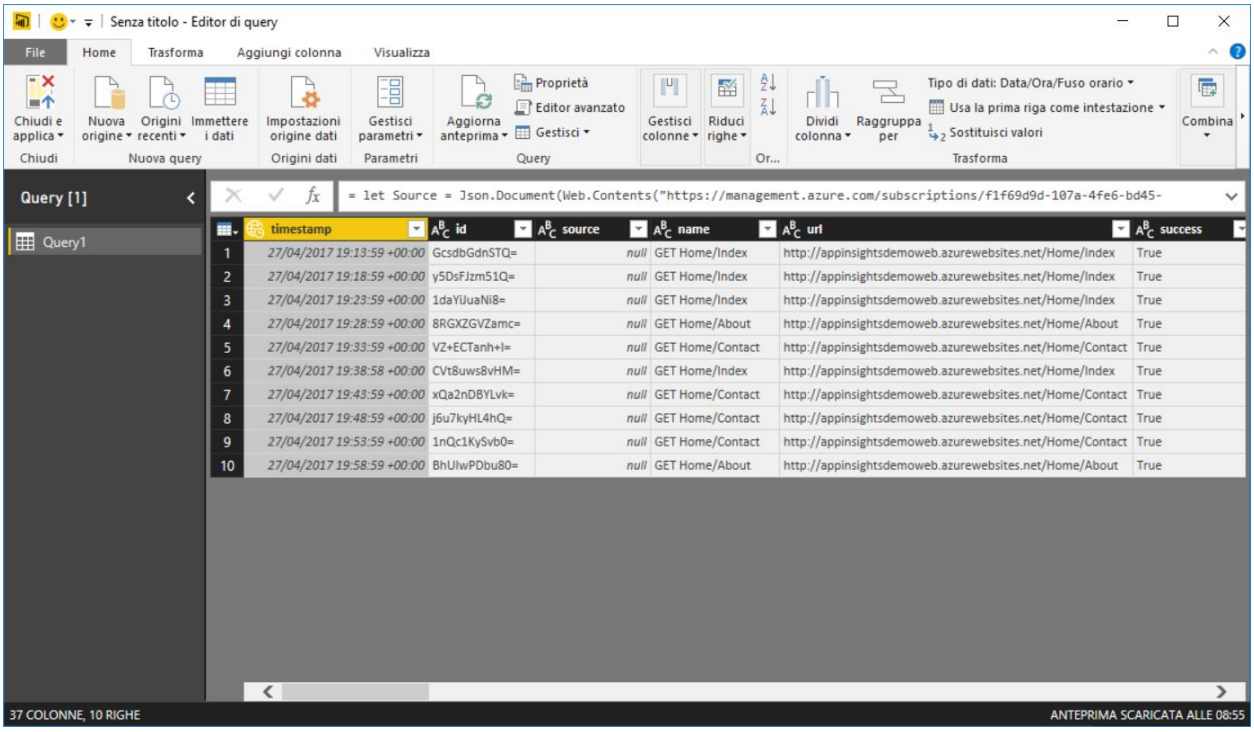


Figure 44: Power Query Running from the Power BI Desktop App

To execute the query from the Power BI app, the user must sign in with valid credentials on the Application Insights resource. To understand how to assign valid credentials to other users, refer to Chapter 5, Access Control.

In the same menu, you can find a **Share a Link to Query** item as well. It is not an export feature; instead, it copies a link to your clipboard that you can share with someone with valid credentials to run the query. Pasting this URL in a browser will bring the user to Application Insights Analytics, running the shared query and returning the same results. This feature can be advantageous if you need to share by email a set of data that updates every time the query is executed.

Shape your data

Sometimes it is difficult to analyze the data with only a grid. Perhaps you need to evaluate a particular KPI, and the best way to do it is with a graphical representation instead of a flat visualization as a grid. To achieve this, Application Insights Analytics offers an alternative chart representation to the grid.

Let's start with an example. Suppose you have a query like this:

Code Listing 19: A Query that Returns a Limited Number of Requests in a Specific Time Range

```
requests | where timestamp >= datetime(2017-04-27) and timestamp <=
datetime(2017-04-28)| limit 10
```


www.dbooks.org

This query will return a grid like the one in Figure 38. Above the grid, you can find two labels: **TABLE**, which stands for the grid visualization; and **CHART**, which represents a graphical display like the one in the following figure.

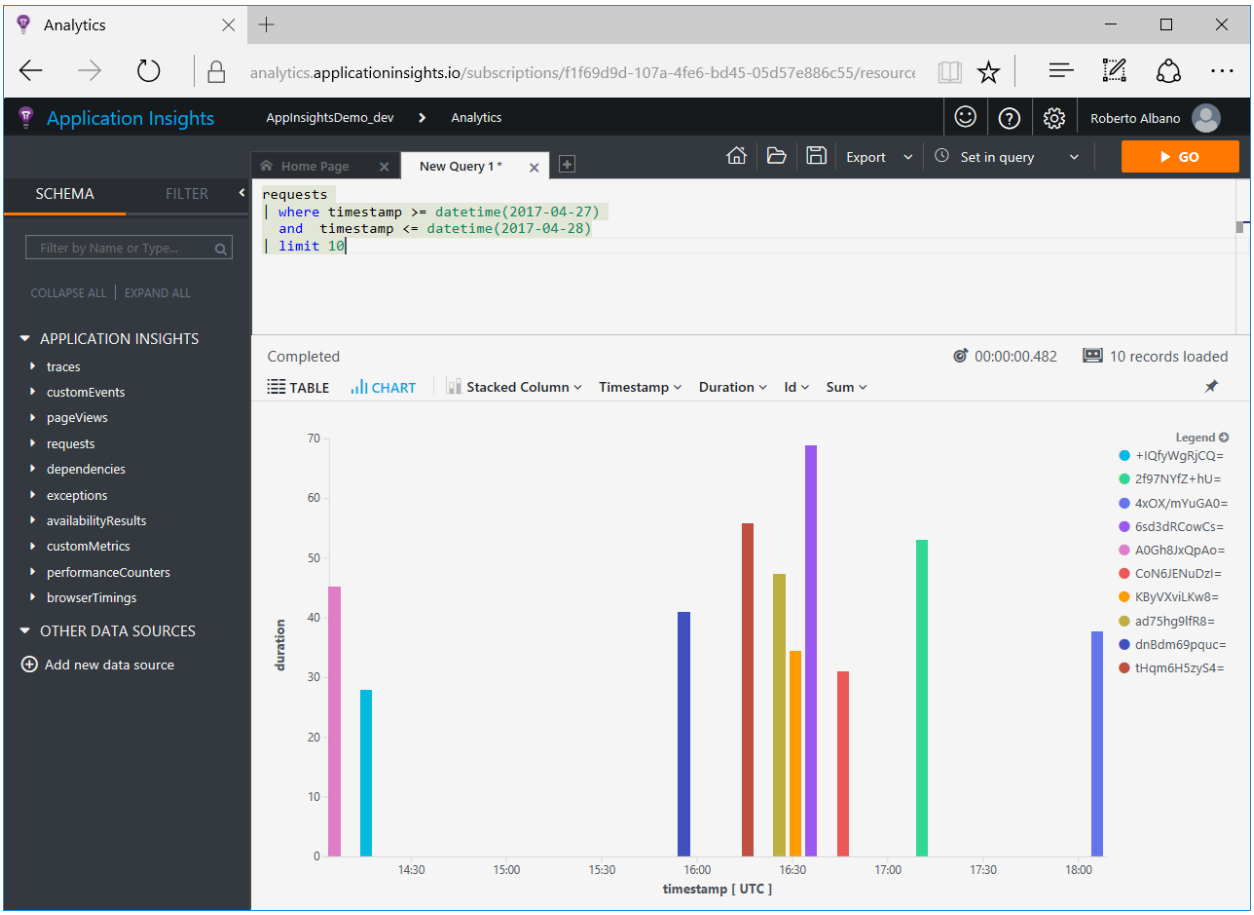


Figure 45: A Chart Visualization

Charts can be more useful than grids when you want to *see* the data instead of *read* it. The data can be arranged into different shapes using the bar options above the chart.

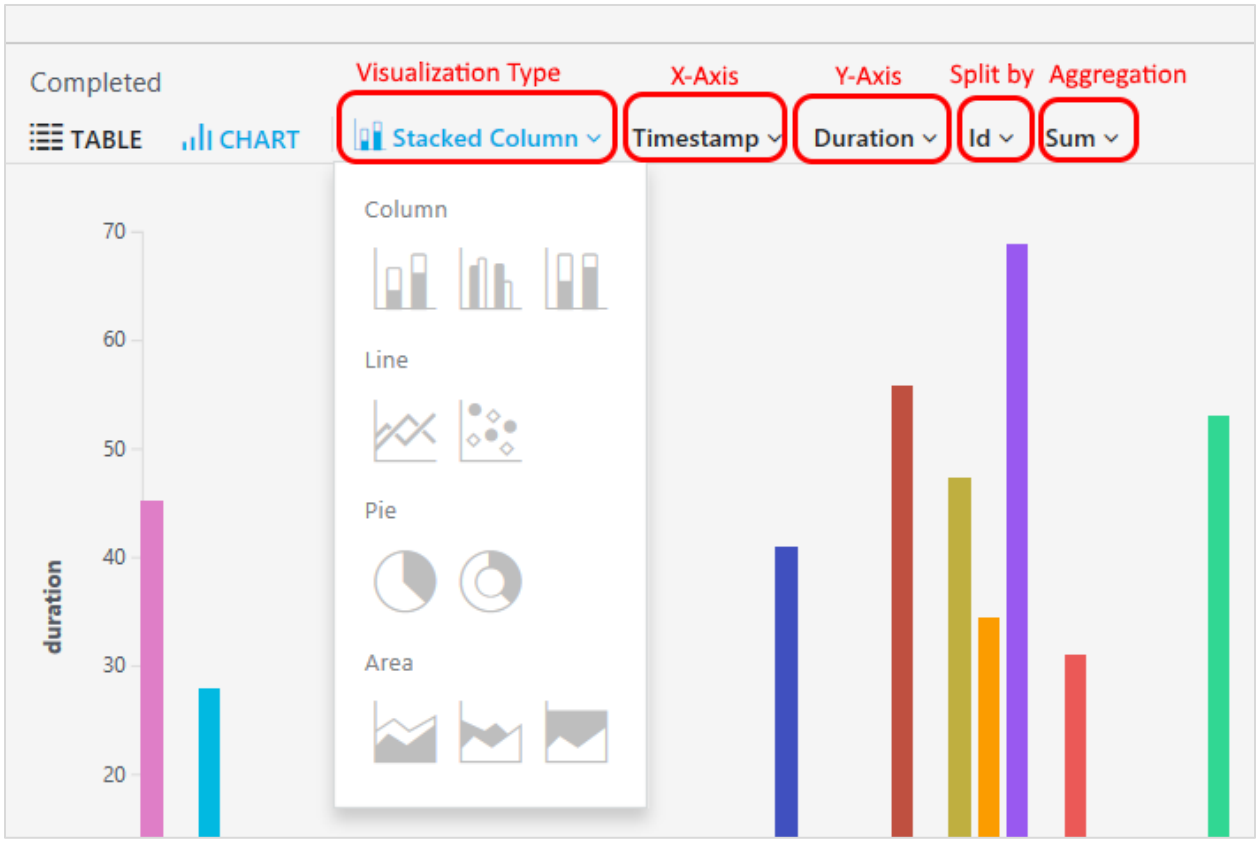


Figure 46: Chart Options

The available options are:

- Visualization Type
  - Here you can choose the kind of chart to display (as seen in Figure 46) with the following:
    - Column charts
      - Stacked column
      - Unstacked column
      - 100% stacked column
    - Line charts
      - Line
      - Scatter
    - Pie charts
      - Pie
      - Doughnut
    - Area charts
      - Stacked area
      - Unstacked area
      - 100% stacked area
- X-Axis
  - Here you can choose the field to use for the horizontal axis of the chart. The list shows the fields available in the query.

www.dbooks.org

- Y-Axis
  - Here you can choose the field to use for the vertical axis of the chart. The list shows the fields available in the query.
- Split by
  - Here you can choose one or more fields to represent within the chart. The list shows the fields available in the query, excluding the ones on the x-axis and y-axis.
- Aggregation
  - Here you can choose the aggregation mode you want to apply to the chart, with the following options:
    - Sum
    - Average
    - Minimum
    - Maximum

Figures 45 and 46 display charts using the **Stacked column** option. This is useful if you want to see at a glance which are the best and the worst results in a metric. The downside is that if you have many items, it could be confusing.

If you want to take a quick look at a summarization or a percentage of the complete data set, you can use the **Pie** option. For example, you can use it when you want to see the country of origin of your visitors, as shown in the following figure.

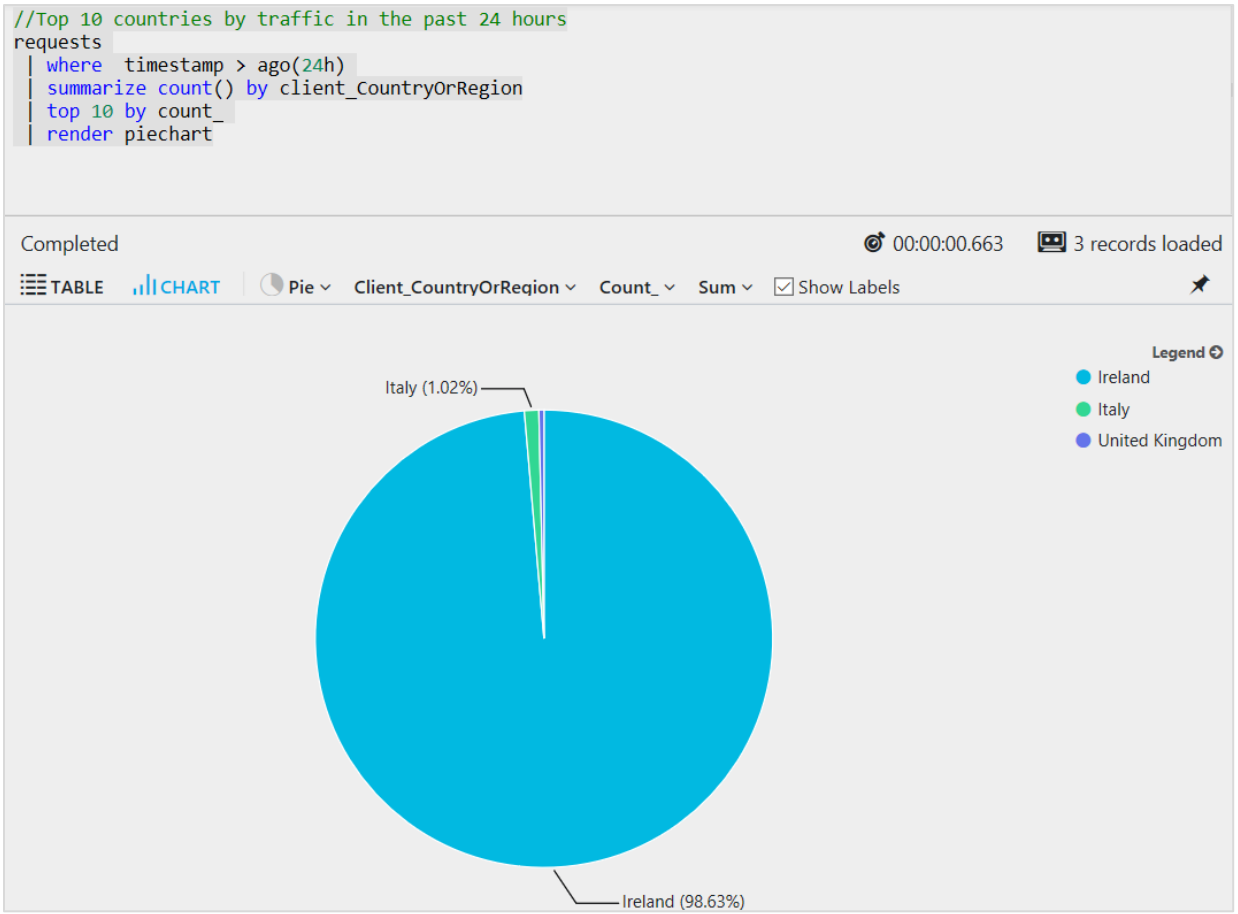


Figure 47: A Pie Chart

Conclusion

Application Insights is a fascinating tool for web-based businesses, providing advantages for both developers and marketing teams.

This tool can be handy for a development team, to help you understand how and where to increase the performance of a site, and where and why there could be issues to fix. In my personal experience, I have removed a large amount of logging by using an Application Insights resource instead. And in every metric, there is more information than a developer can write accurately with any kind of logging tool.

It can be a valuable tool for a marketing team as well, allowing you to learn what parts of the front end are more or less appreciated, and subsequently, where to invest. Using external tools such as PowerBI can give marketers the ability to see your data from different points of view.

The presence of an infrastructure like Azure adds to Application Insights an enormous power due to its ability to handle large amounts of data without problems.

I would recommend trying Application Insights if you have never used it before. It will surprise you if you are a skeptic, and will certainly help all the people involved in any part of your product's chain and organization.

www.dbooks.org