Get diagnostics & analytics for your ASP.NET Web App with Application Insights

Overview

Quickly learn how you can take advantage of diagnostics and analytics capabilities from Visual Studio Application Insights to identify, understand and solve problems in your ASP.NET Apps. Get started from Visual Studio and see how easy it is to enable Application Insights for your project and start sending out-of-the-box telemetry and custom instrumentation.

Application Insights is a solution for developers which provides fast, easy and non-intrusive analytics tools to interactively diagnose problems and answer tough questions. You can diagnose problems right from within your development environment and incorporate monitoring in your existing ALM workflows with Azure & Visual Studio Team Services integrations.

Objectives

- Create an Azure Account for Application Insights (if do not have one)
- Add Application Insights to an ASP.NET App using Visual Studio 2015
- Diagnose issues in Application Insights Azure Portal
- Ask deeper questions with Analytics for Application Insights

Prerequisites

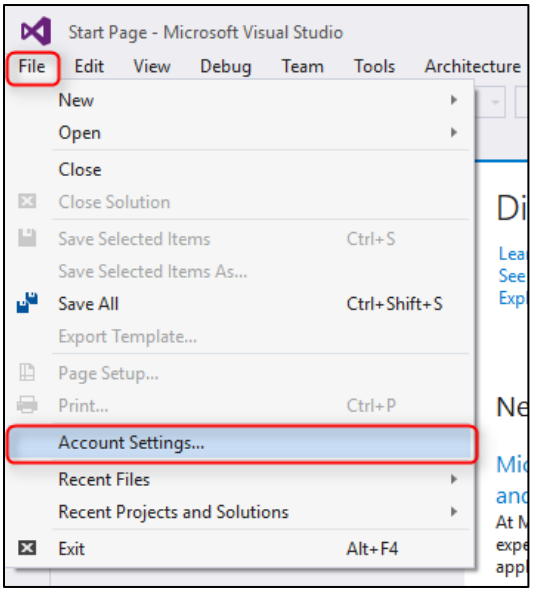
- Visual Studio 2015 Update 2
- Microsoft Azure Subscription
    - If you already have an Azure Subscription, use the Microsoft Account associated with that.
    - If not and you have an MSDN subscription, you already have access to Microsoft Azure; you simply need to activate it in your MSDN benefits portal, if you have not already.
    - You can also create a free trial Azure account, for which you would need a Credit Card for Security Verification

Intended Audience

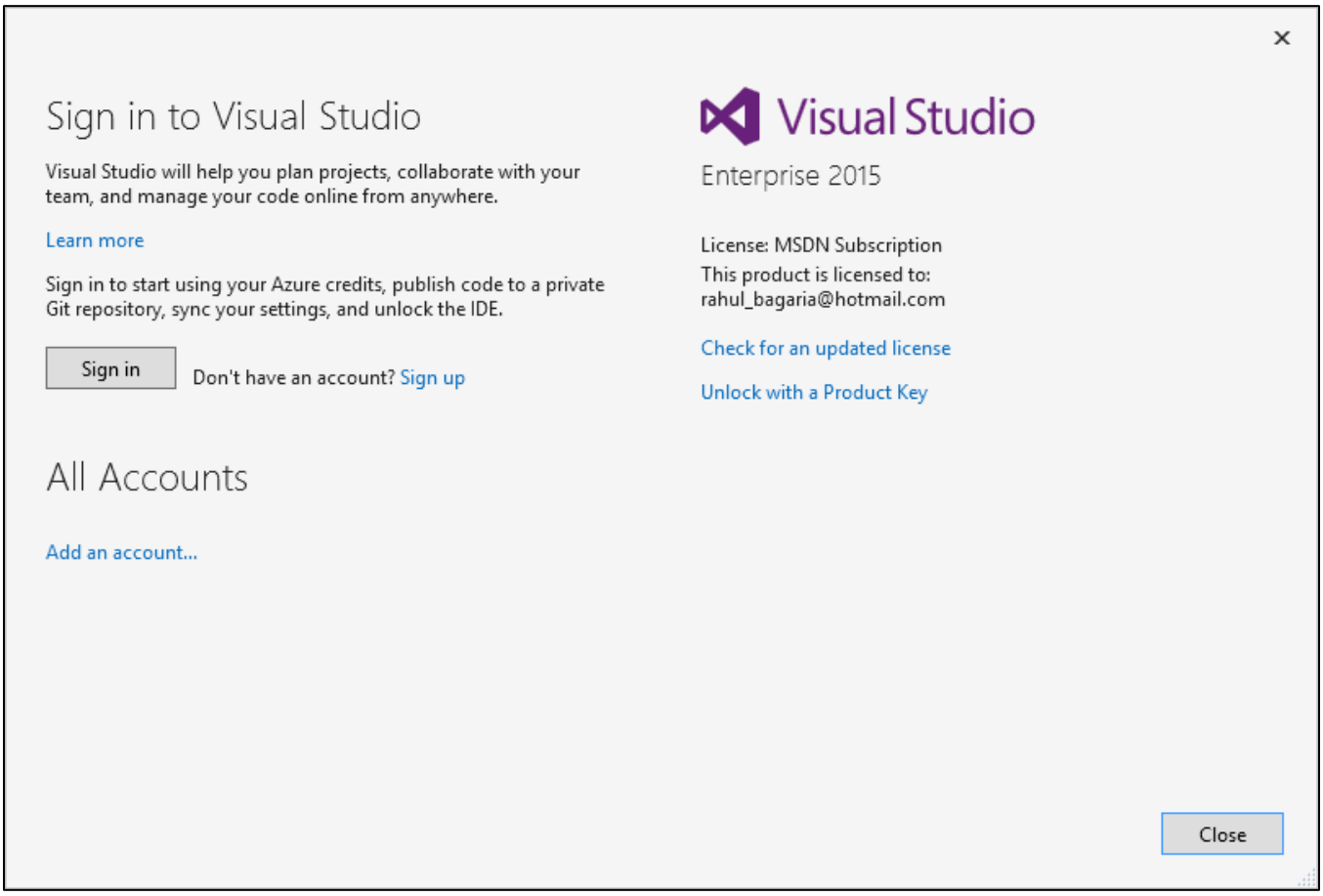
This Quick Start Challenge is intended for developers familiar with ASP.NET web apps and Visual Studio. It does not require prior experience with Application Insights (or diagnostics in general).

Step 0: Connect to an Azure Subscription in Visual Studio

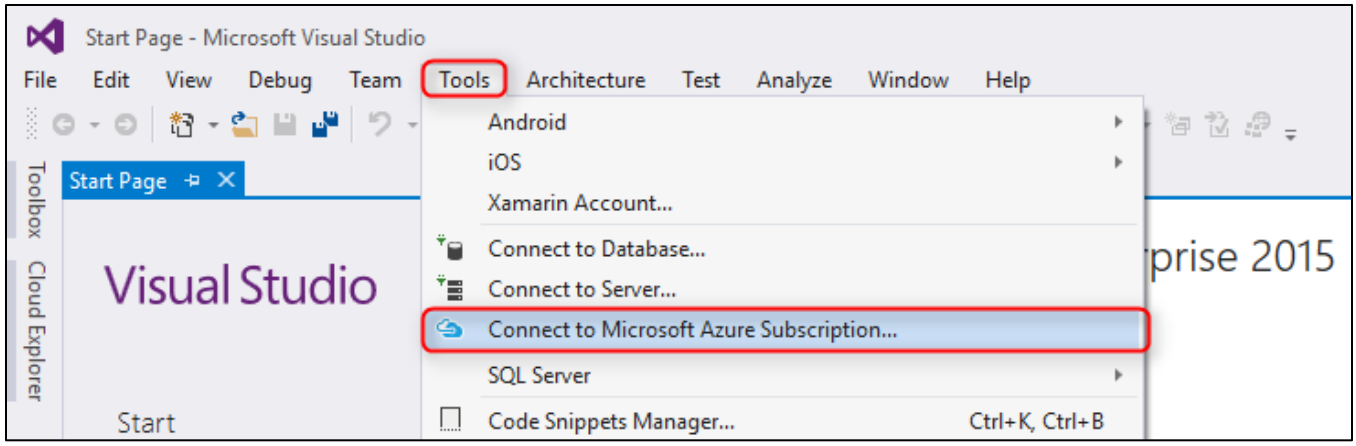
Launch **Visual Studio 2015** from the Start Menu. Navigate to File -> Accounts and ensure there are no existing accounts present on the system.



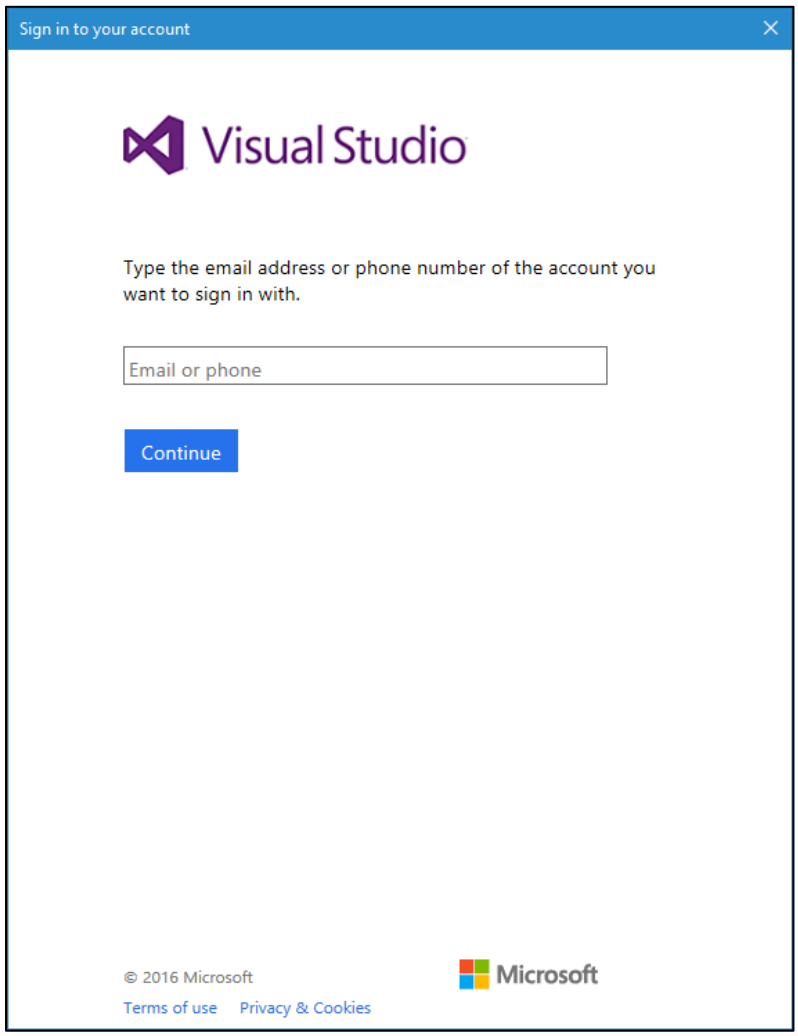
If there are, please remove them before starting.



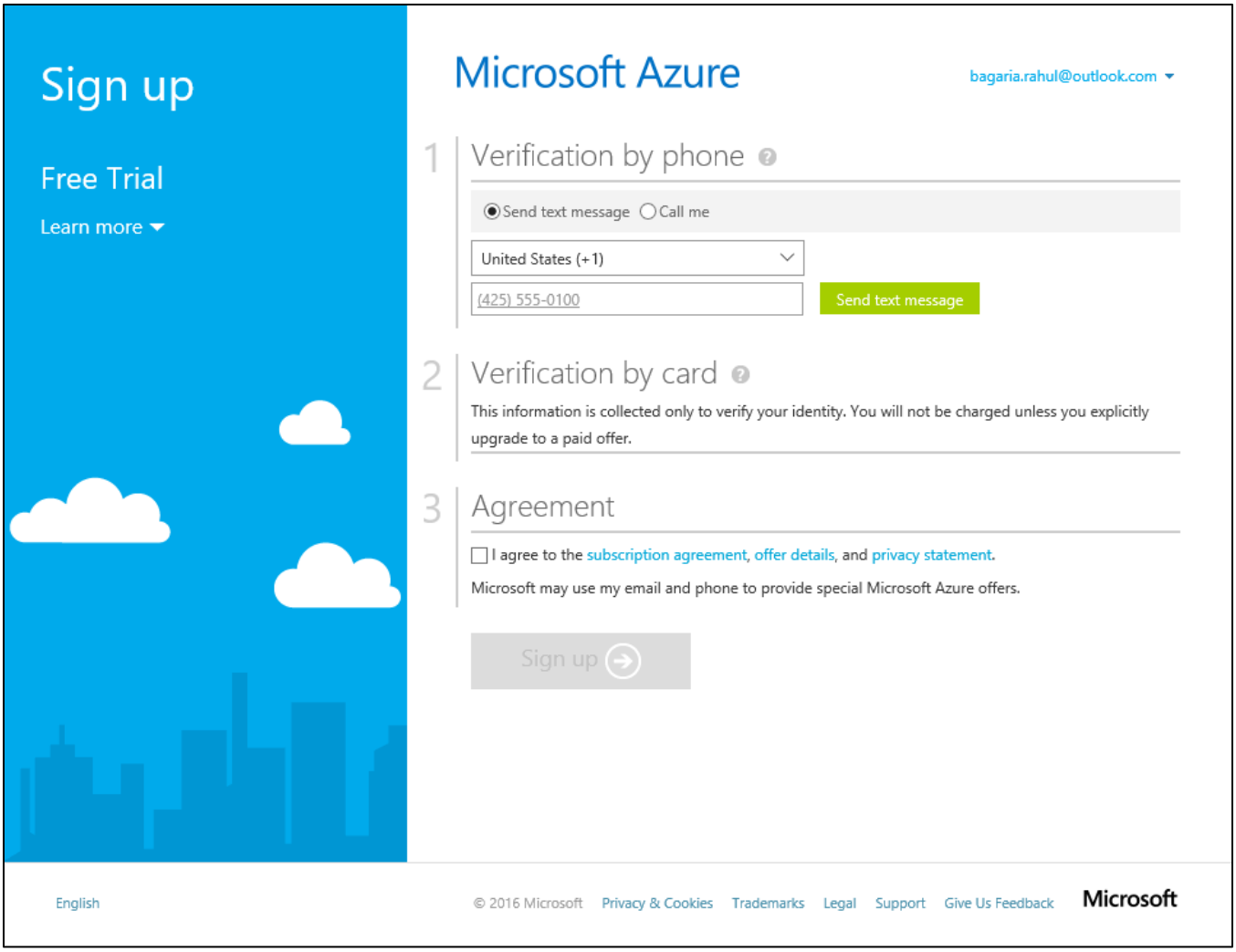
Close the above pup-up and navigate to Tools -> Connect to Microsoft Azure Subscription to get started.



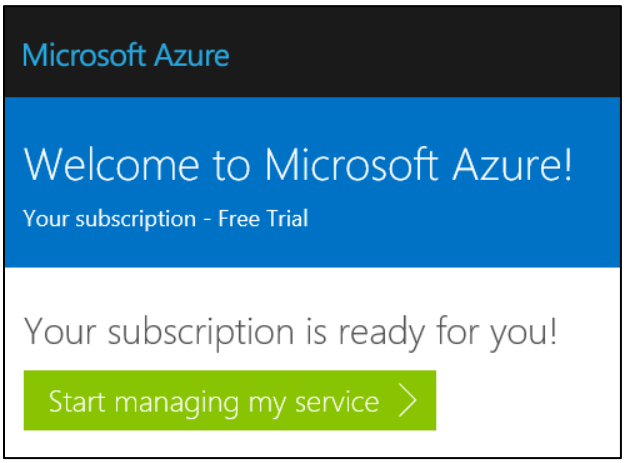
As you are not signed in, you will need to sign in with your Microsoft Account, that you will use for Azure as well (if you already have an Azure Subscription, use the Microsoft Account that has access to that):



Once you log in, you will have to sign up for an Azure subscription (if your Microsoft Account is not already associated with an Azure Subscription).

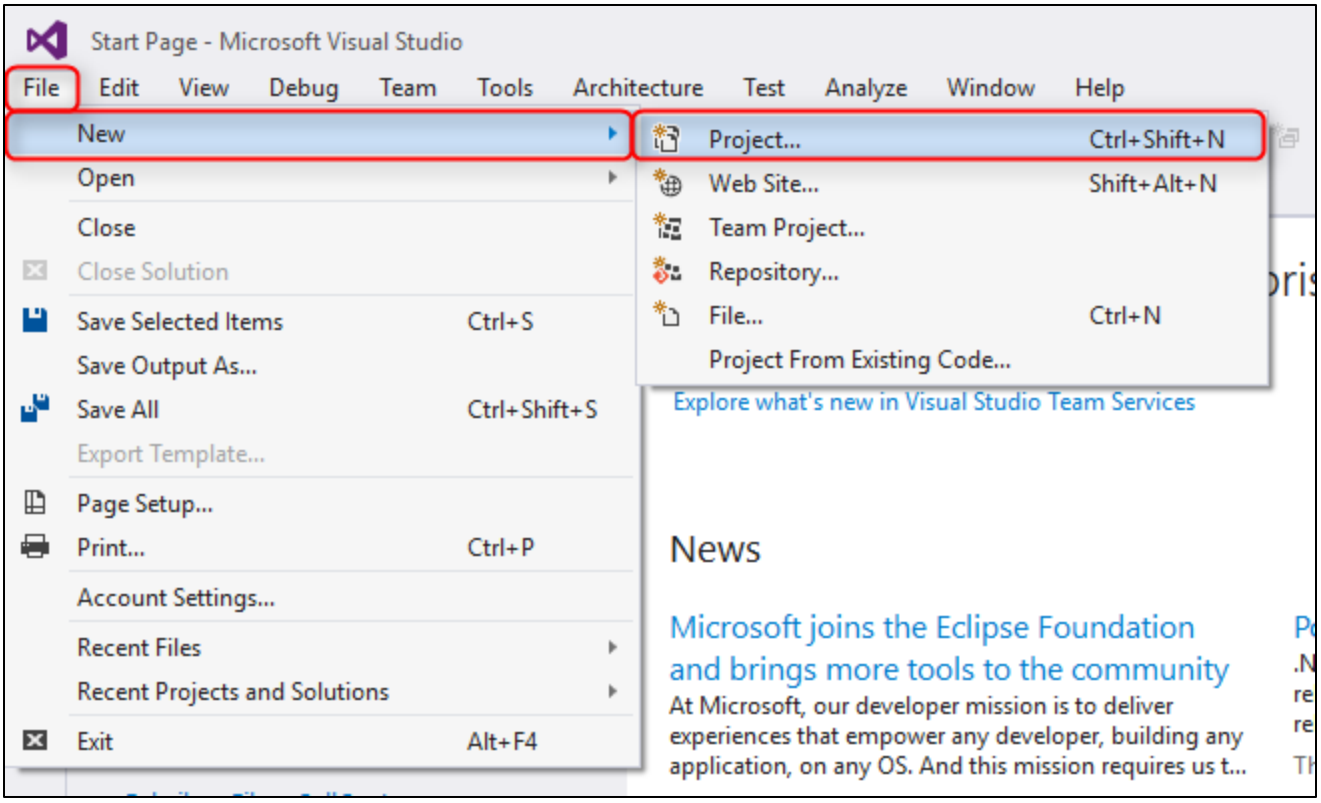


Complete the sign up process for a new FREE Azure Subscription (if needed):

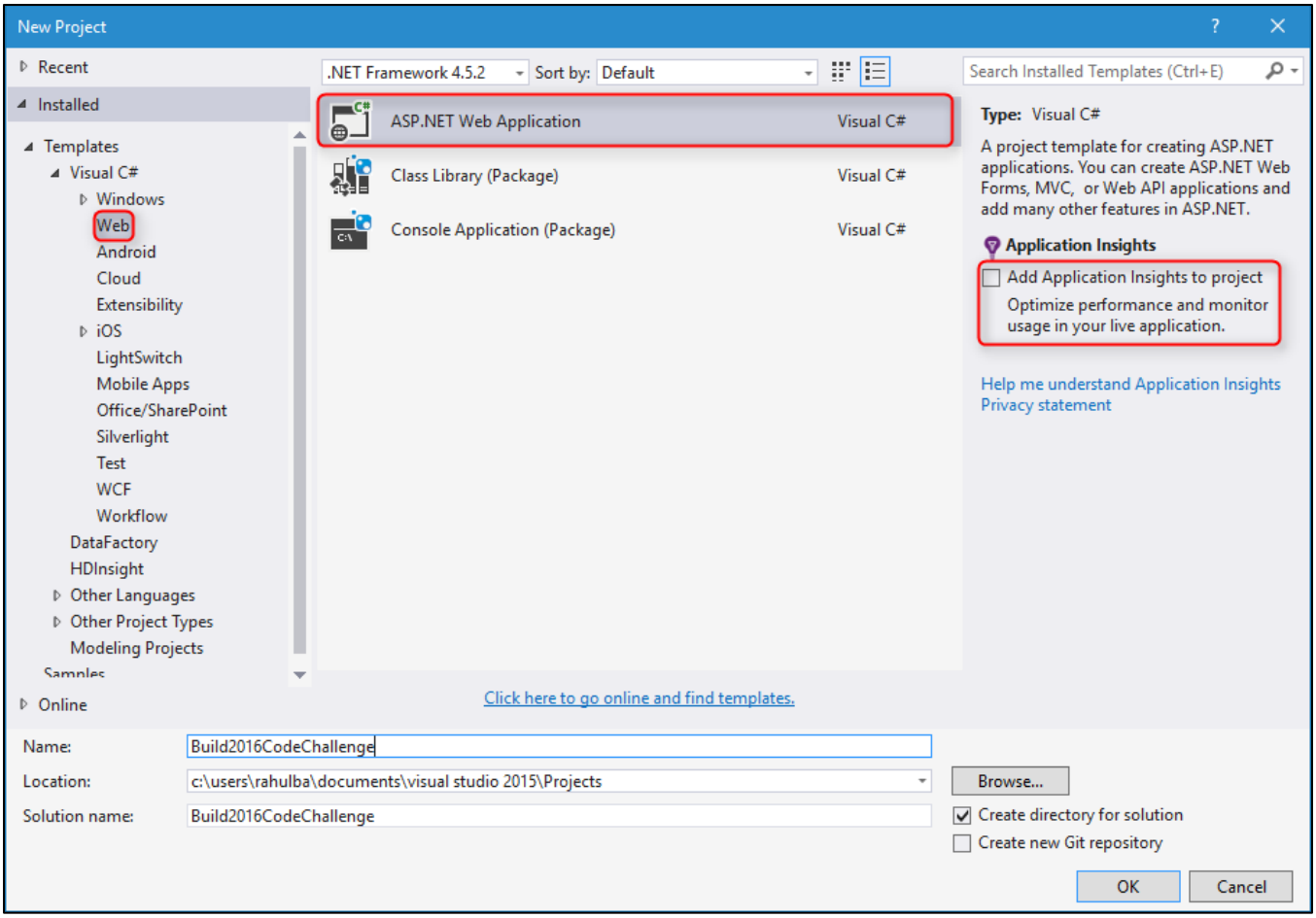


Step 1: Create new ASP.NET app in Visual Studio with Application Insights

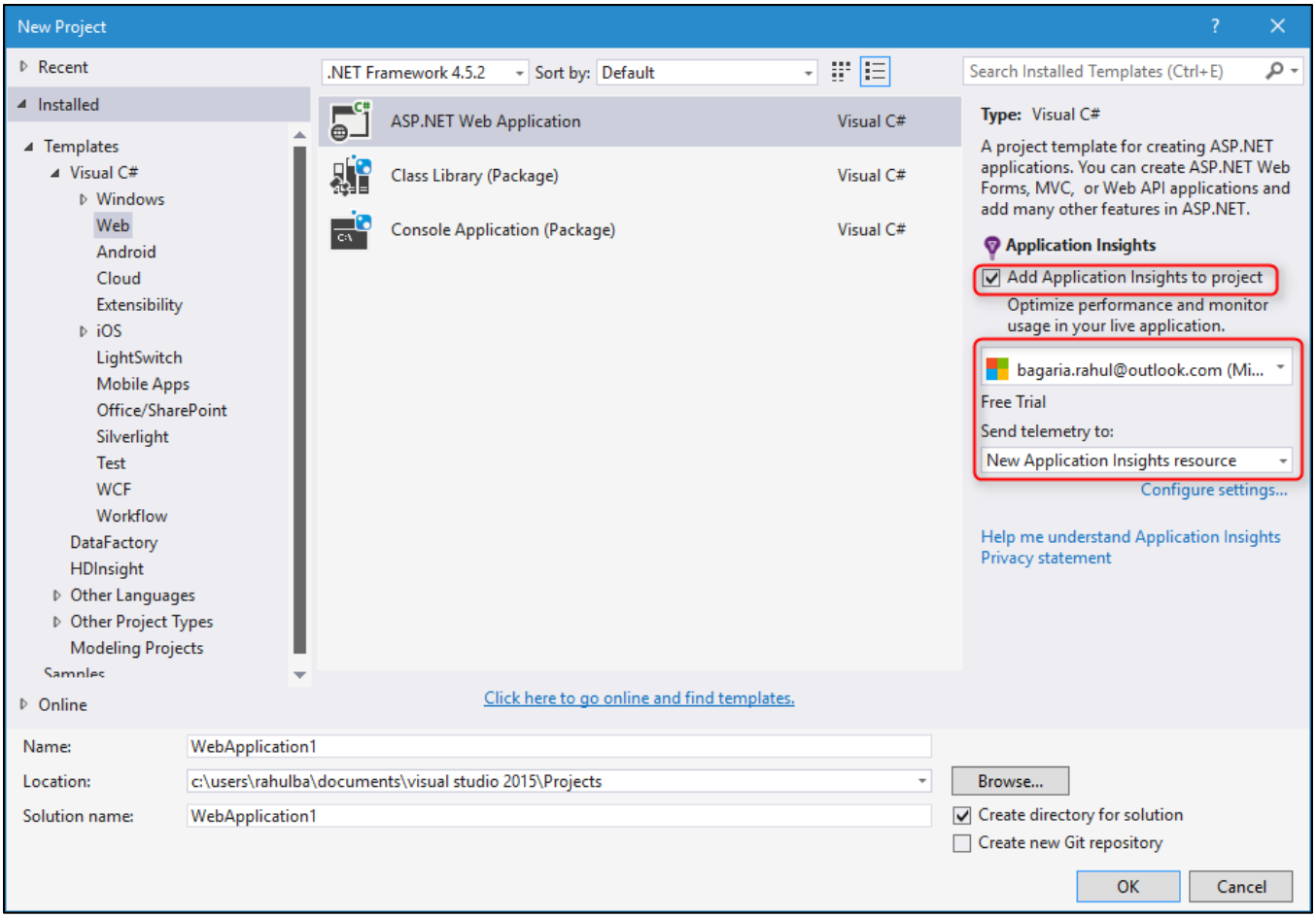
Launch **Visual Studio 2015** from the Start Menu. Then, create a new project:



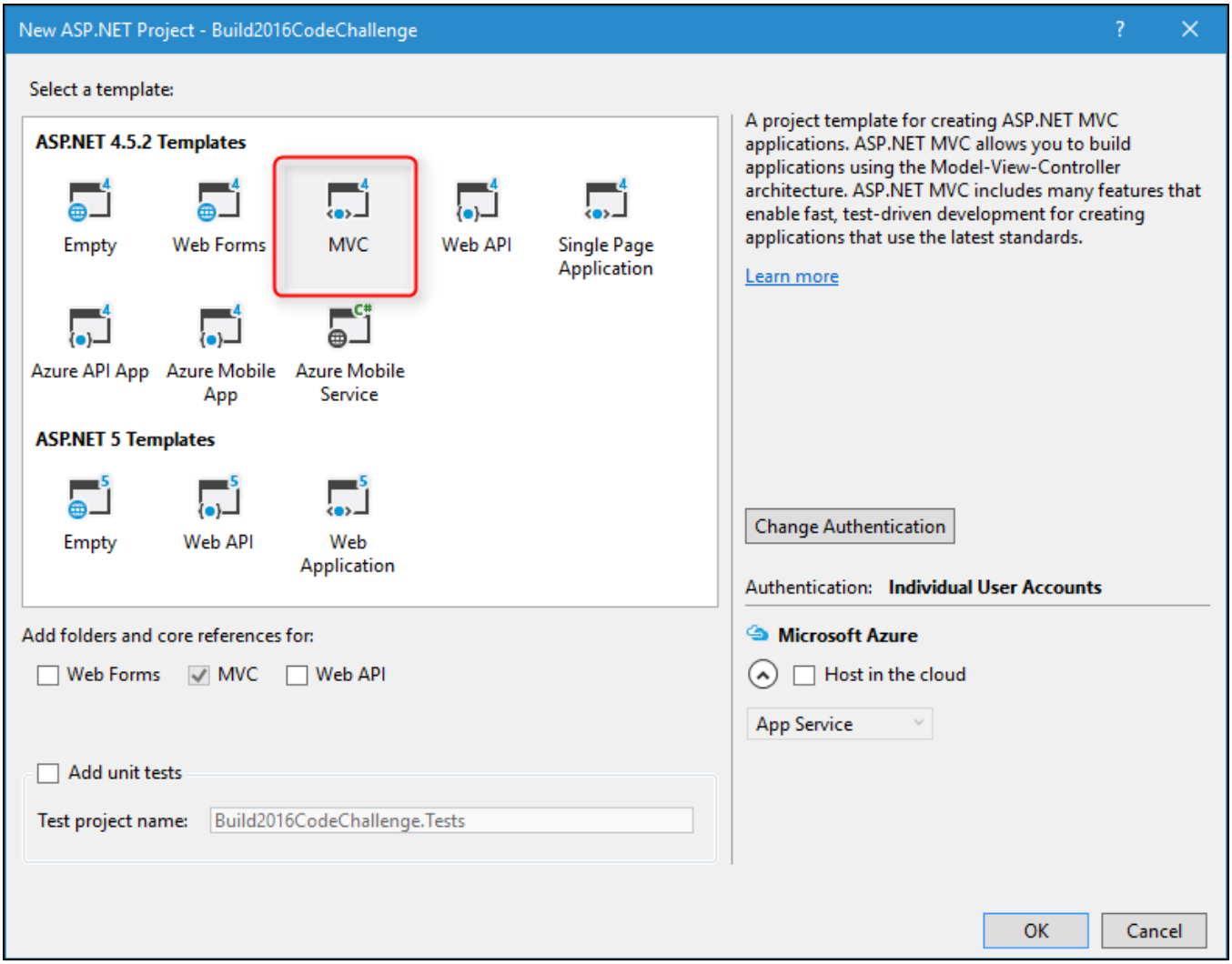
Select Installed → Templates → Visual C# → Web → ASP.NET Web Application for your project template:



If the checkbox on the right side panel to “Add Application Insights to project” is not selected, mark it checked. Since, you have already connected your Azure Subscription, you will see the details updated. Continue with Free Trial or choose your Azure Subscription you want to send telemetry to. You can let Application Insights send telemetry to a new Application Insights resource:

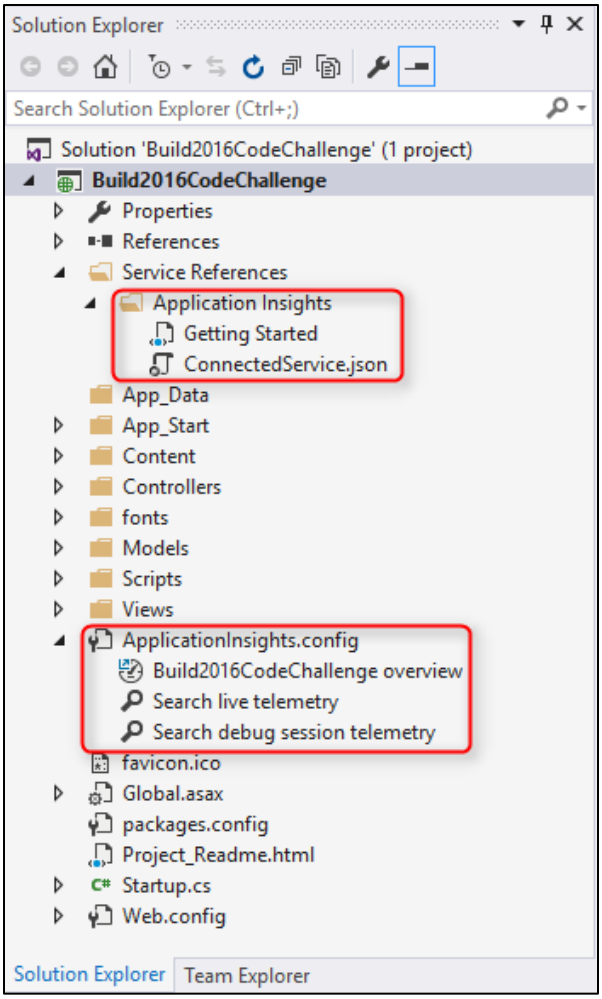


Give the project a name, for the lab you should use our sample name of “Build2016CodeChallenge”. Click OK to select the project type. Choose MVC as the project template and click OK to create the project.



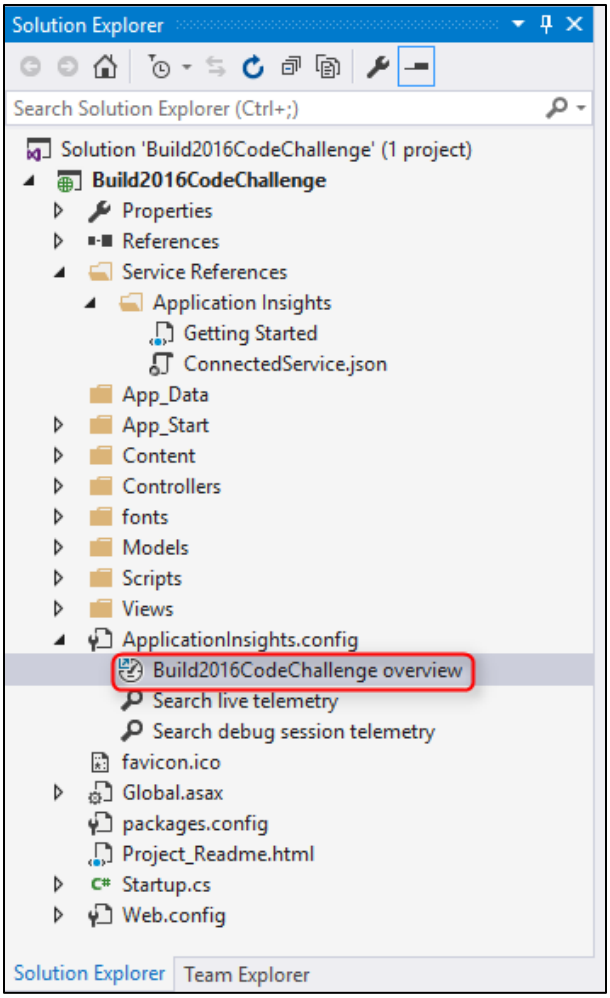
Step 2: Start getting out of the box telemetry with Application Insights

As the project is created, Application Insights will be automatically added to the project and configured to send telemetry to the resource you selected previously. You'll also notice that there are now new elements in your solution including a new ApplicationInsights.config file:

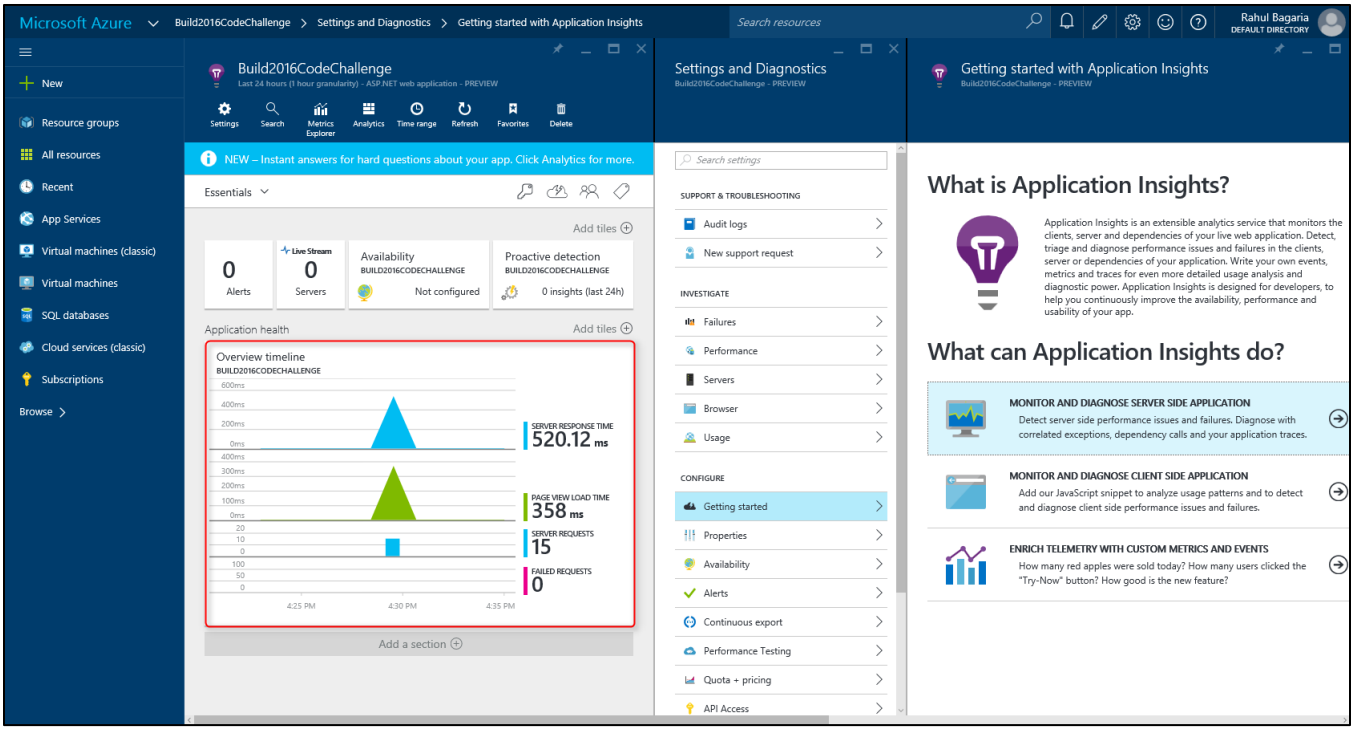


Press F5 to build and run your web app. Once it opens (it may take a minute or two for the first run), navigate to various pages, including Home, About, Register and Contact. When you're done, close the browser window.

When you return to Visual Studio, get to the Solution Explorer and double click on "Build2016CodeChallenge overview" under ApplicationInsights.config to open Application Insights portal:

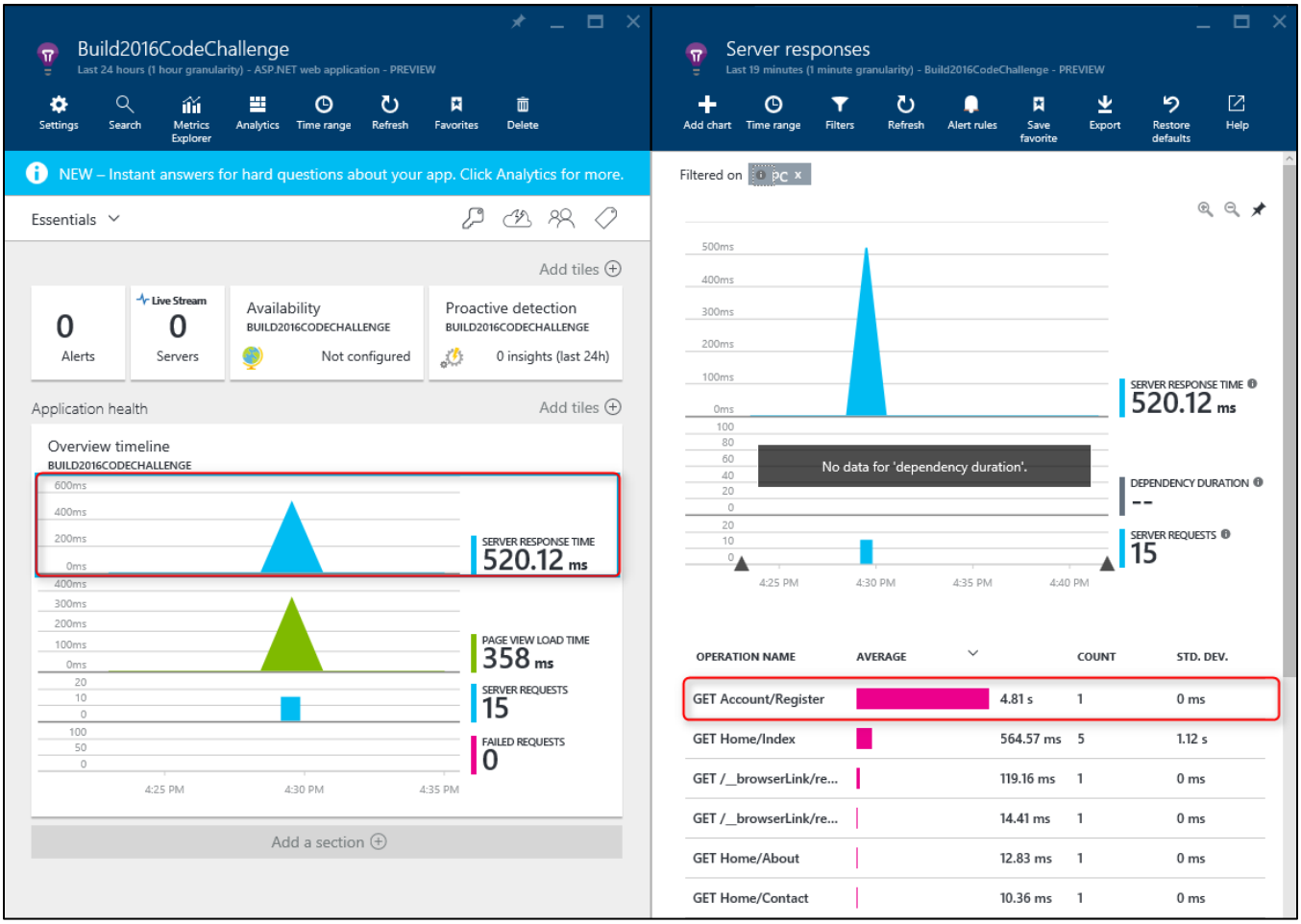


Sign into Azure using your credentials, and the overview blade of your Application Insights resource will open. You can close the Getting Started blade for now and read through it later at leisure.

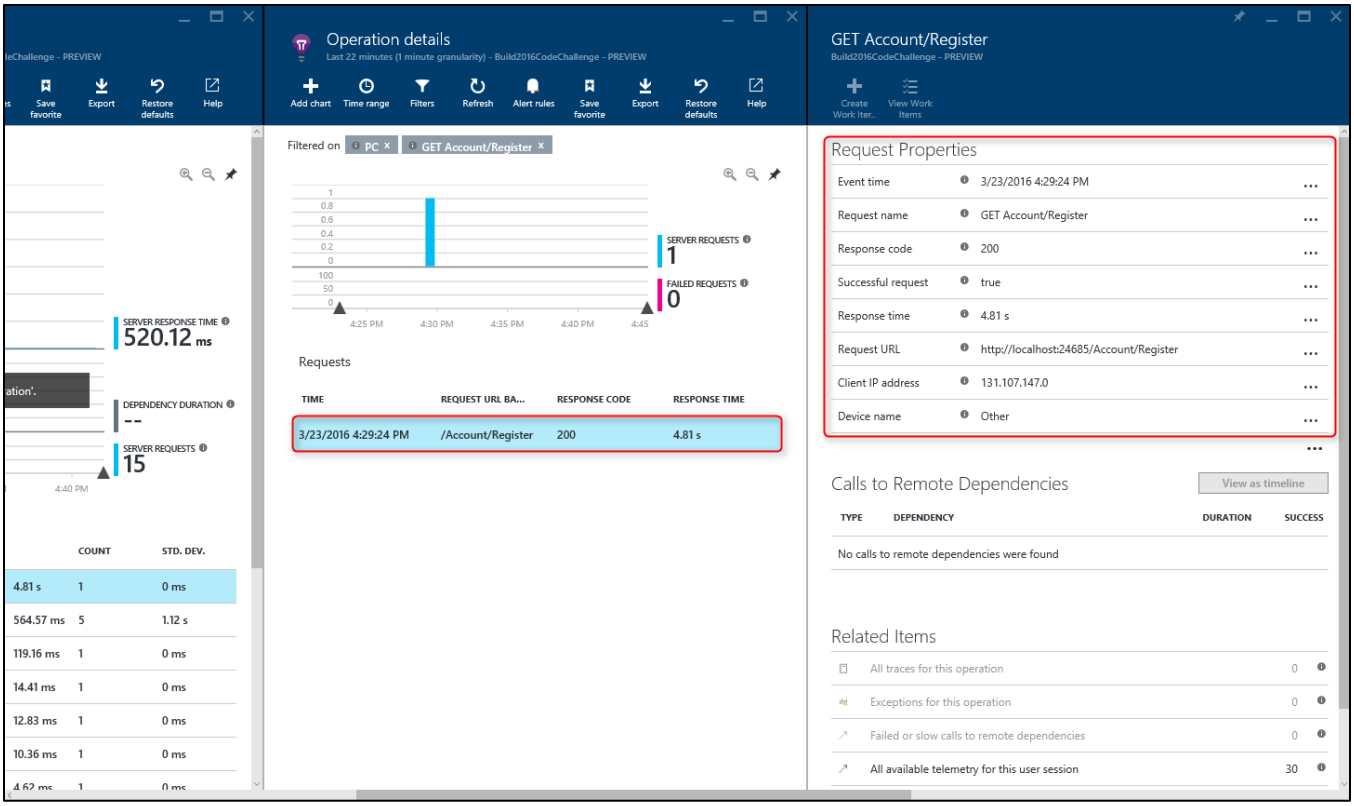


You now have out-of-the-box telemetry summarizing the requests processed by your web app, how quickly it responded to them and page load times at the client browser. You don't seem to have caught any exceptions so far since it was a new project.

Click on Server Response Time Chart to open a details blade on "Server responses". You will see aggregations on requests grouped by Operation Name, sorted on the request response time.



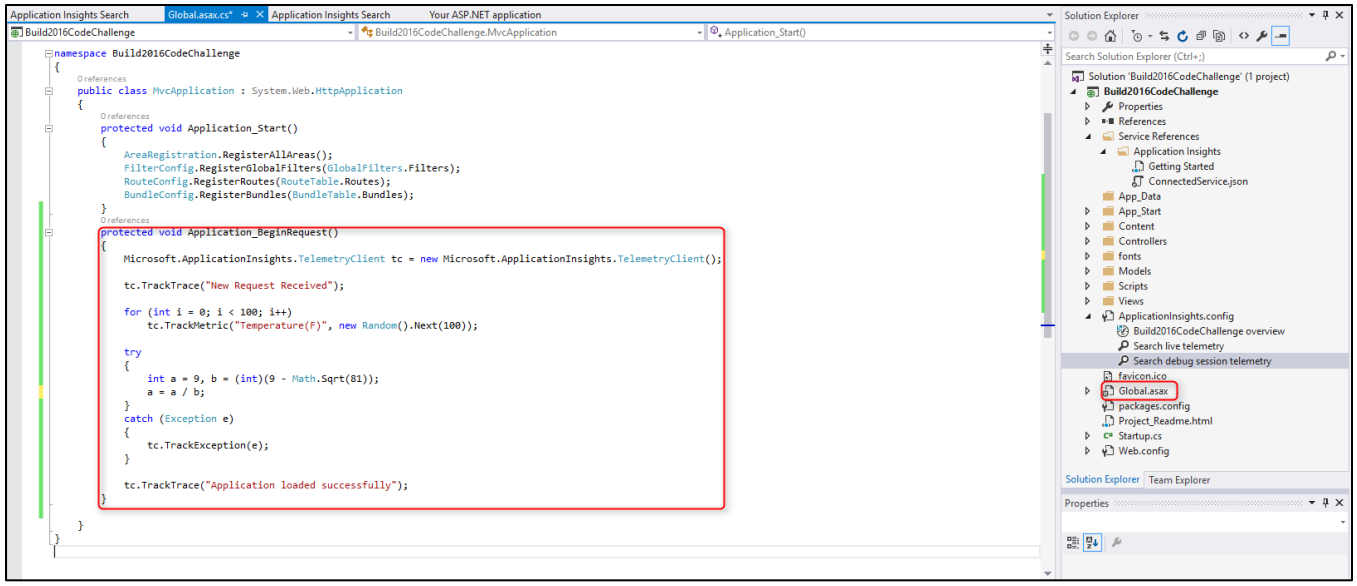
Click on the slowest request at the top (GET Account/Register) to navigate to Operation details. Clicking on the specific request instance will open the details blade with all captured properties.



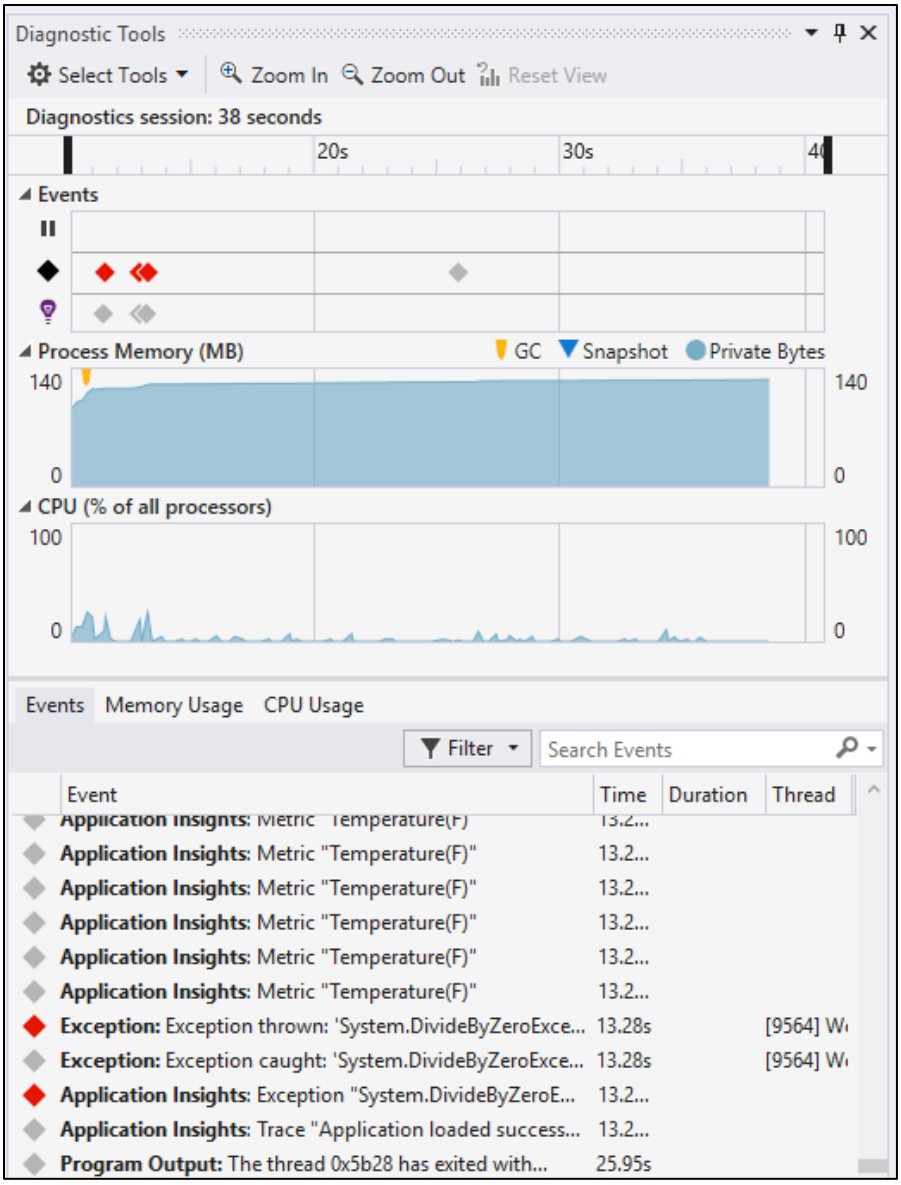
Step 3: Catch an exception and diagnose within Visual Studio

Come back to Visual Studio, navigate to the Global.asax.cs file and add the following code to the MvcApplication class:

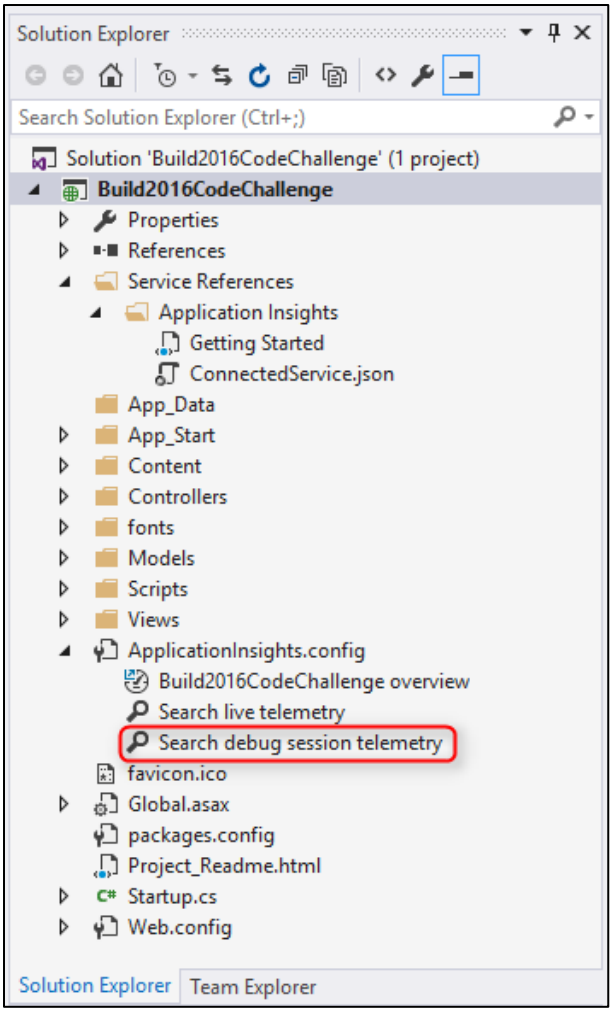
```
protected void Application_BeginRequest()
{
        Microsoft.ApplicationInsights.TelemetryClient tc = new
        Microsoft.ApplicationInsights.TelemetryClient();

        tc.TrackTrace("New Request Received");

        for(int i = 0; i<100; i++)
            tc.TrackMetric("Temperature(F)", new Random().Next(100));

        try
        {
            int a = 9, b = (int)(9-Math.Sqrt(81));
            a = a / b;
        }
        catch (Exception e)
        {
            tc.TrackException(e);
        }

        tc.TrackTrace("Application loaded successfully");
}
```



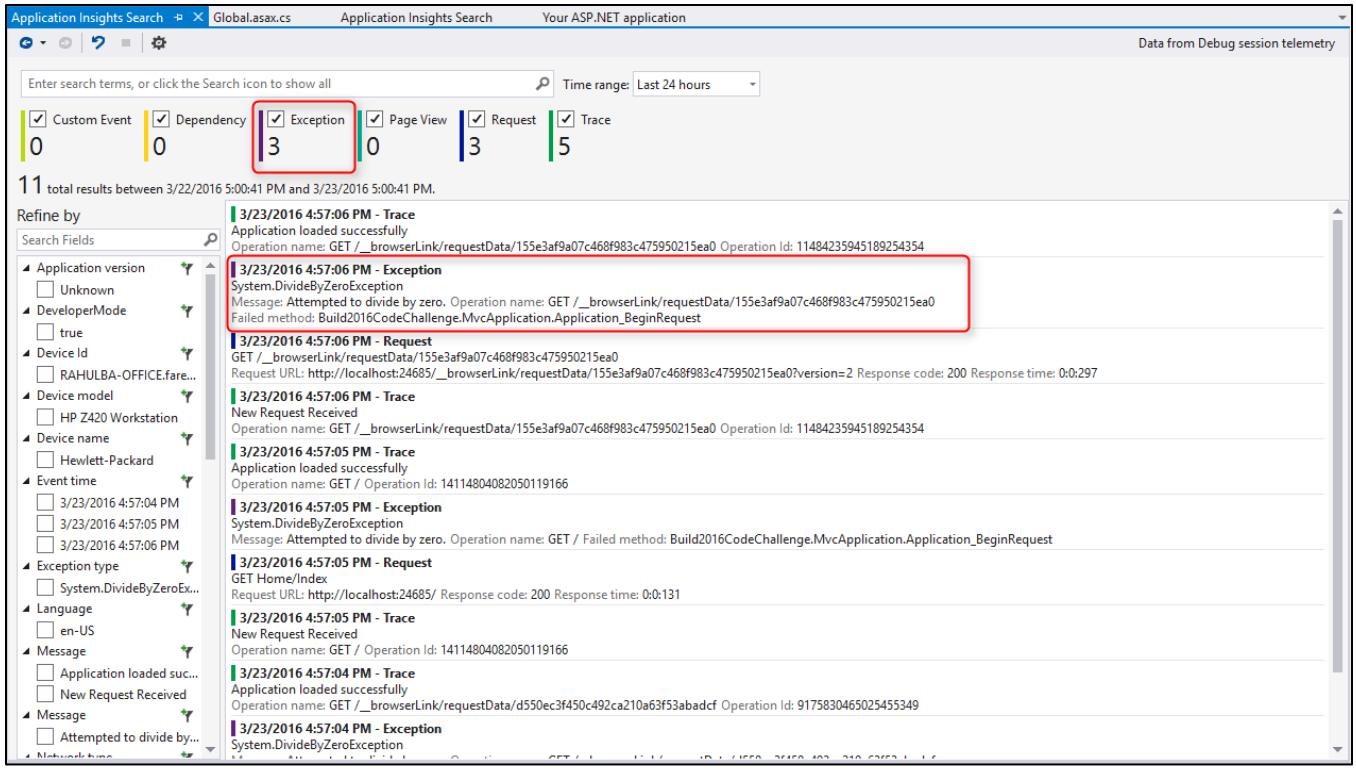
Use F5 to Build and Run the application again. Refresh the home page a few times. If you check out Visual Studio while the app is running, you will notice that the Diagnostics Tools show all the events captured by Application Insights.



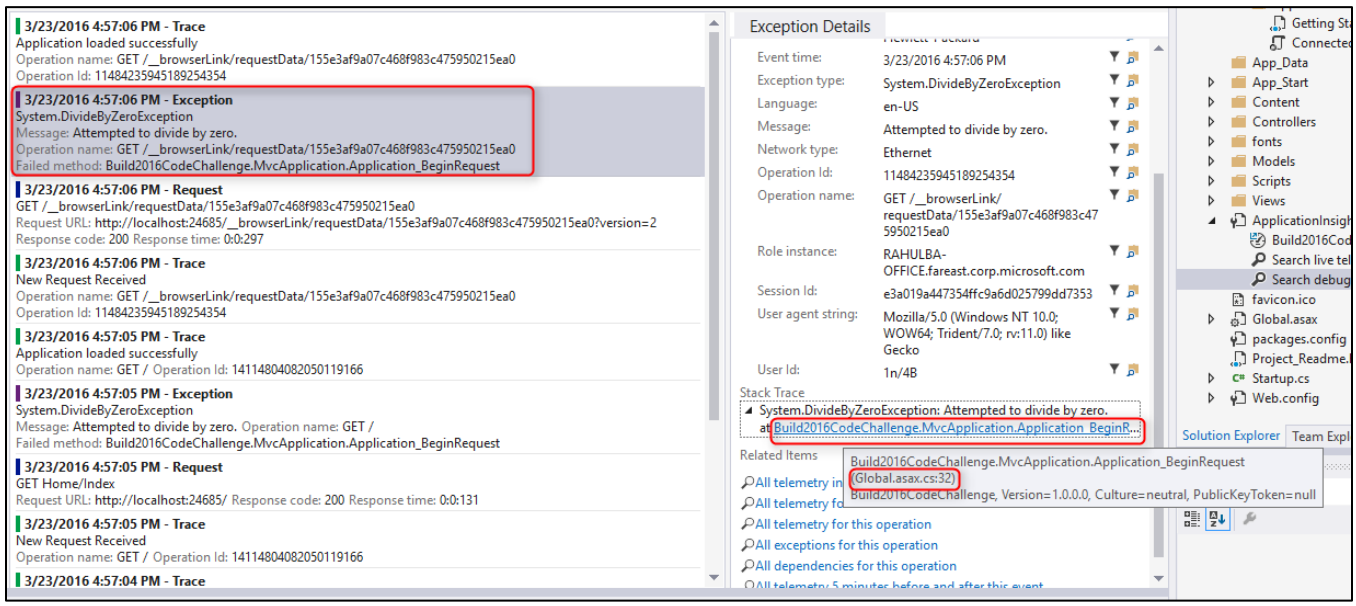
Close the web app, return to Visual Studio and double-click on “Search debug session telemetry” under ApplicationInsights.config to explore the data from the last run.



In the Application Insights Search page you will see all the events captured in the debug sessions so far. You will notice there are 3 Exceptions listed. Click on one of them to drill down and get more detailed information about what went wrong.



You will be able to see to the exact Stack Trace along with the method name and line of code.



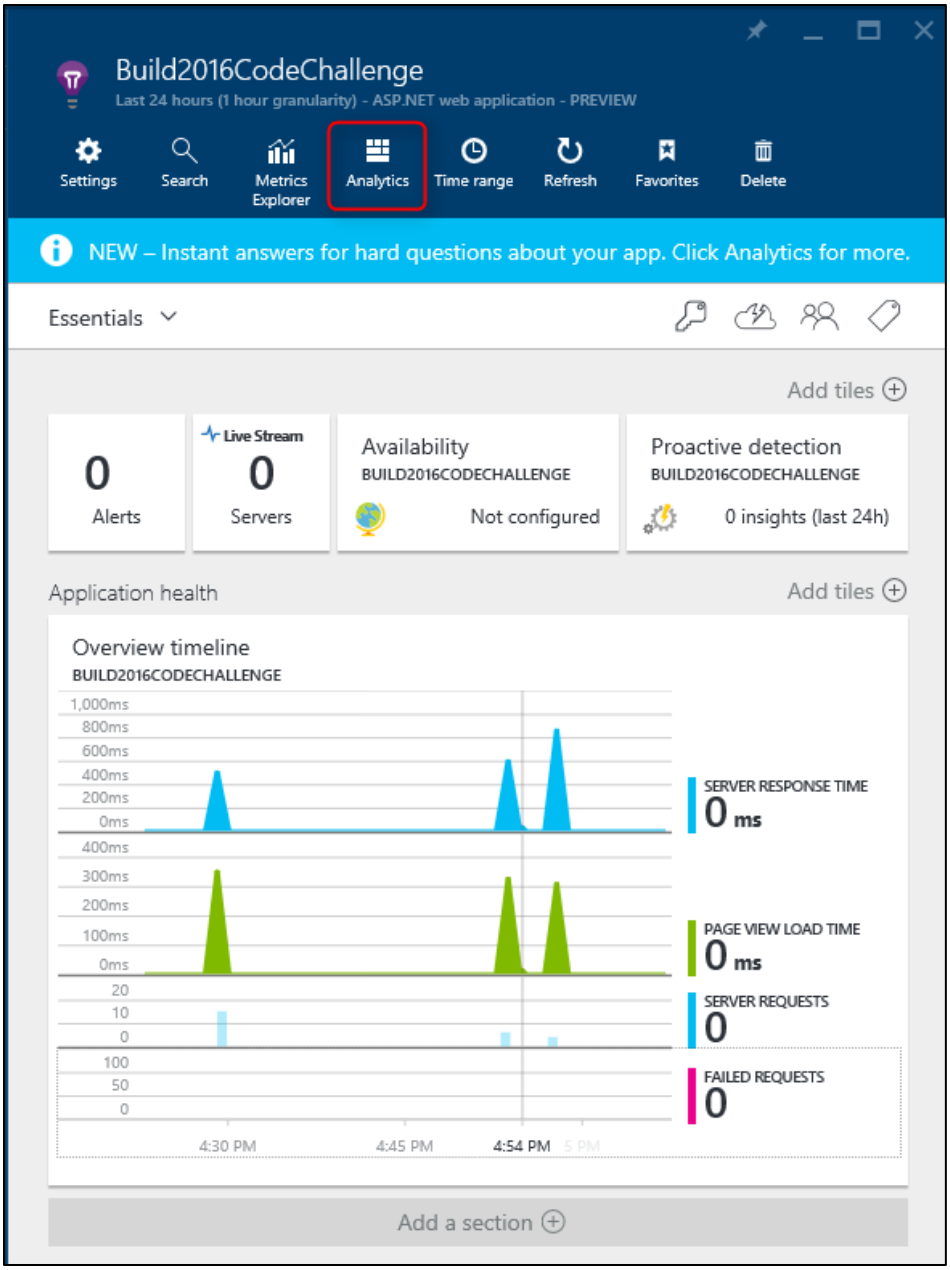
Click on the method name URL to jump right back to the particular method in code. Navigate to the line number mentioned in the tool tip above and you will see the “DivideByZero” exception in the try block.

```
namespace Build2016CodeChallenge
{
    0 references
    public class MvcApplication : System.Web.HttpApplication
    {
        0 references
        protected void Application_Start()
        {
            AreaRegistration.RegisterAllAreas();
            FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
            RouteConfig.RegisterRoutes(RouteTable.Routes);
            BundleConfig.RegisterBundles(BundleTable.Bundles);
        }
        0 references
        protected void Application_BeginRequest()
        {
            Microsoft.ApplicationInsights.TelemetryClient tc = new Microsoft.ApplicationInsights.TelemetryClient();

            tc.TrackTrace("New Request Received");

            for (int i = 0; i < 100; i++)
                tc.TrackMetric("Temperature(F)", new Random().Next(100));

            try
            {
                int a = 9, b = (int)(9 - Math.Sqrt(81));
                a = a / b;
            }
            catch (Exception e)
            {
                tc.TrackException(e);
            }
```

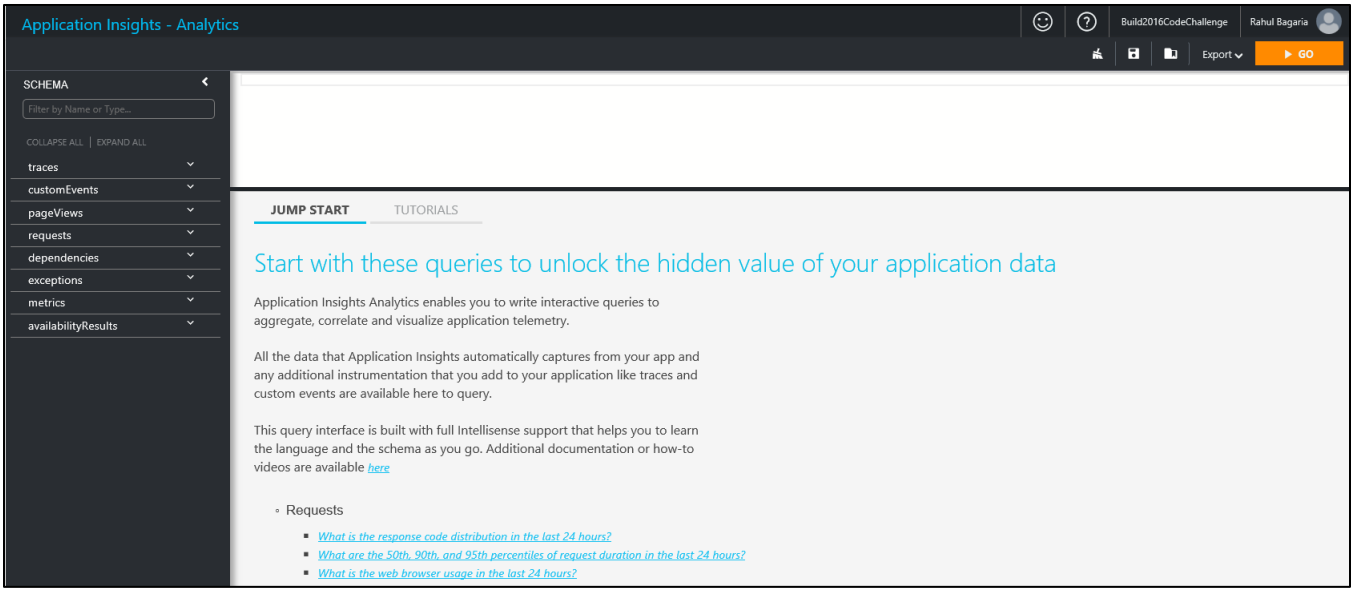
You get the same experience when your app is running live in production and with Application Insights you can capture and diagnose exceptions right from within Visual Studio or even from the Azure Portal experience.

Step 4: Run custom queries on your data with the new Analytics capability

In the browser, open the Application Insights blade for Build2016CodeChallenege resource in Azure Portal. Click on “Analytics” button at the top of the blade, which will take you to the Analytics UI.

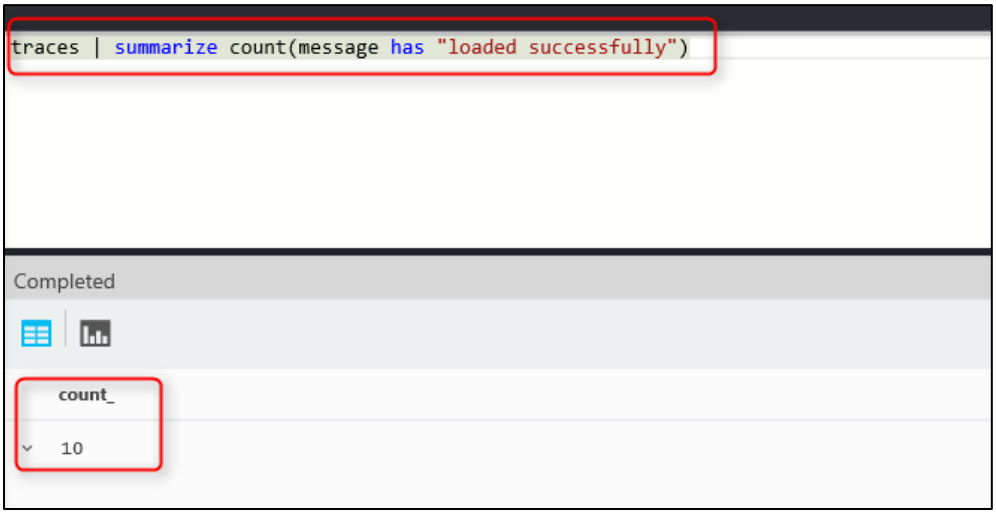


Here you can write powerful queries in a SQL like query language and ask tough questions on your data set for instant answers.



Write the following query in the query-box at the top and press Go (or Shift + Enter). You will instantly see the no. of trace messages having the phrase “loaded successfully”.

```
traces | summarize count(message has "loaded successfully")
```



Next you can write a query to visualize the frequency distribution of temperature (which was a custom metric you sent to Application Insights from the code). Just press enter to go to the next line, write the following query and press Go:

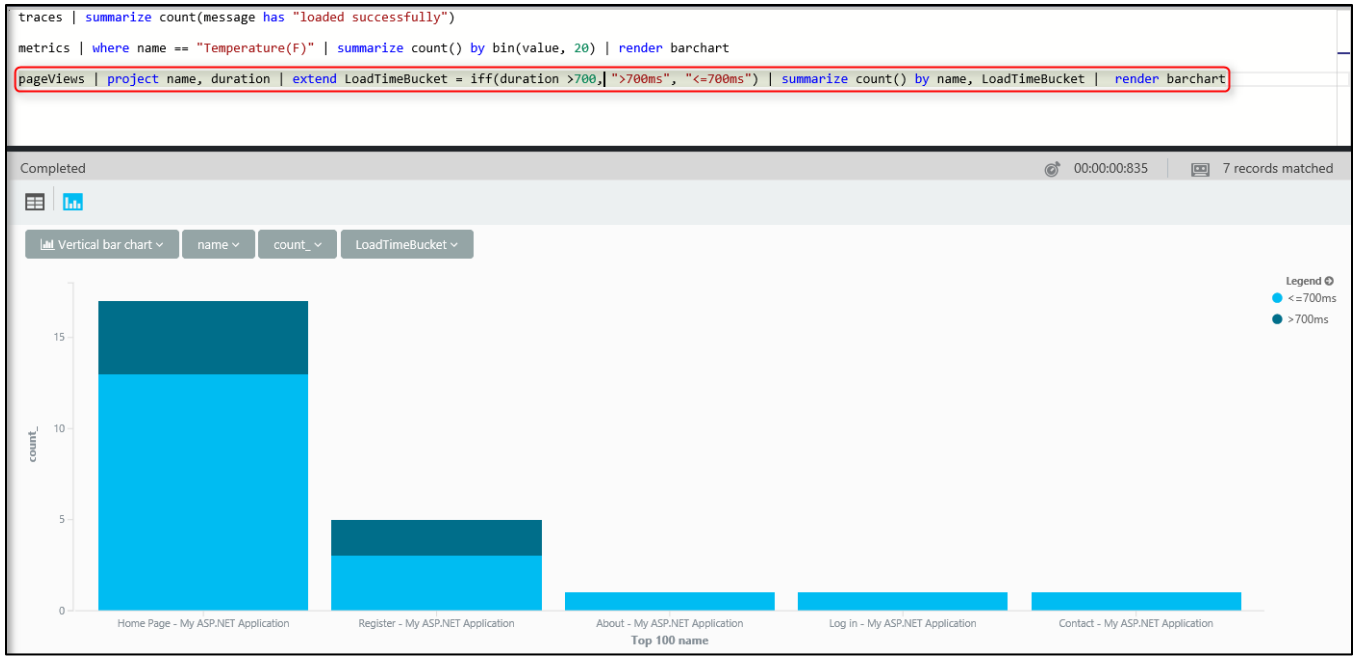
```
metrics | where name == "Temperature(F)" | summarize count() by bin(value, 20) | render barchart
```

You can also see all the pages you have viewed so far along with their page load duration:

```
pageViews | project name, duration
```

You can extend this query with bucketed duration to get a more intuitive visualization

```
pageViews | project name, duration | extend LoadTimeBucket = iff(duration >700, ">700ms",
"<=700ms") | summarize count() by name, LoadTimeBucket | render barchart
```



That is all for today! You are now ready to explore Application Insights on your own and discover how easy it is to set up alerts on your metrics, diagnose problems in client & server side performance, identify dependency issues, set up web tests, filter & segment data on out-of-the-box and custom properties, integrate with your existing DevOps processes, and so much more.