Enabling monitoring on your ASP.NET and ASP.NET Core based web applications running on [Azure App Services](https://docs.microsoft.com/en-us/azure/app-service/) is now easier than ever. Whereas previously you needed to manually install a site extension, the latest extension/agent is now built into the app service image by default. This article will walk you through enabling Application Insights monitoring as well as provide preliminary guidance for automating the process for large-scale deployments.

**Note**

Manually adding an Application Insights site extension via **Development Tools** > **Extensions** is deprecated. This method of extension installation was dependent on manual updates for each new version. The latest stable release of the extension is now [**preinstalled**](https://github.com/projectkudu/kudu/wiki/Azure-Site-Extensions) as part of the App Service image. The files are located in d:\Program Files (x86)\SiteExtensions\ApplicationInsightsAgent and are automatically updated with each stable release. If you follow the agent based instructions to enable monitoring below, it will automatically remove the deprecated extension for you.

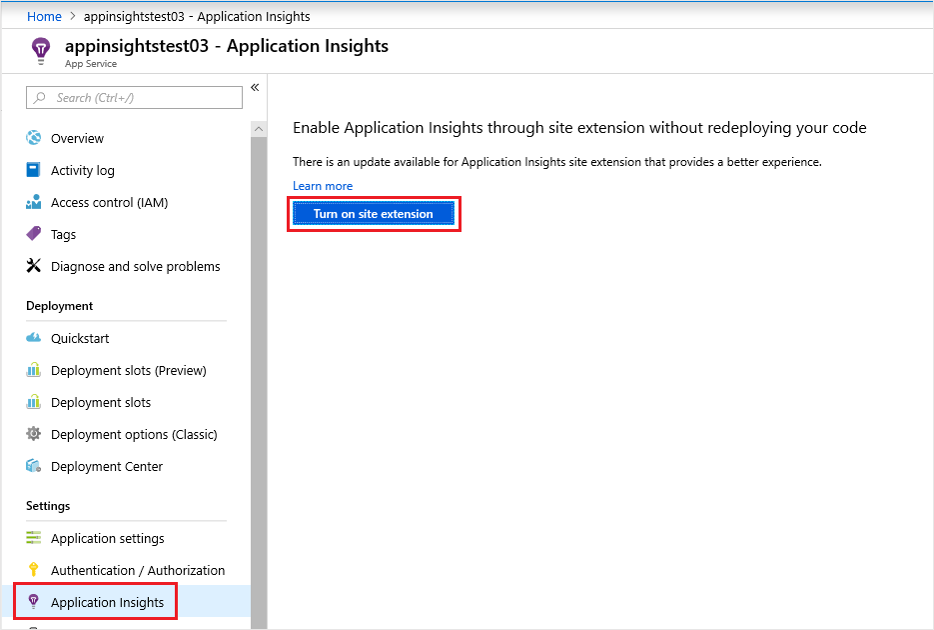
**Enable Application Insights**

There are two ways to enable application monitoring for Azure App Services hosted applications:

* **Agent-based application monitoring** (ApplicationInsightsAgent).
  + This method is the easiest to enable, and no advanced configuration is required. It is often referred to as "runtime" monitoring. For Azure App Services we recommend at a minimum enabling this level of monitoring, and then based on your specific scenario you can evaluate whether more advanced monitoring through manual instrumentation is needed.
* **Manually instrumenting the application through code** by installing the Application Insights SDK.
  + This approach is much more customizable, but it requires [adding a dependency on the Application Insights SDK NuGet packages](https://docs.microsoft.com/en-us/azure/azure-monitor/app/asp-net). This method, also means you have to manage the updates to the latest version of the packages yourself.
  + If you need to make custom API calls to track events/dependencies not captured by default with agent-based monitoring, you would need to use this method. Check out the [API for custom events and metrics article](https://docs.microsoft.com/en-us/azure/azure-monitor/app/api-custom-events-metrics) to learn more. This is also currently the only supported option for Linux based workloads.

**Note**

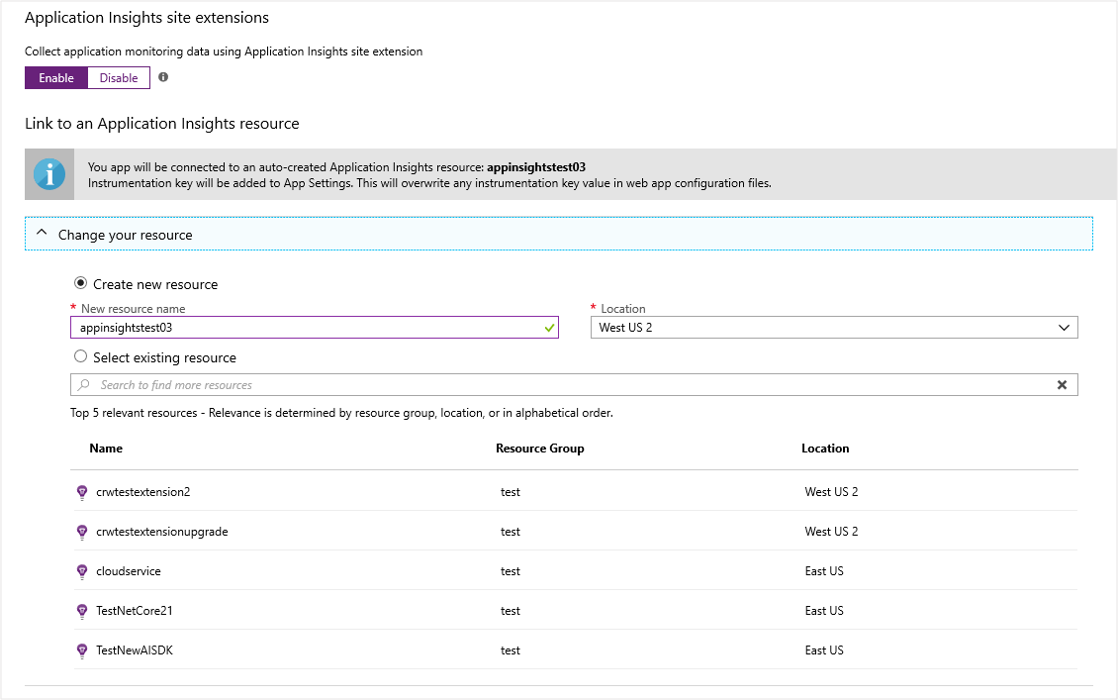
1. **Select Application Insights** in the Azure control panel for your app service.



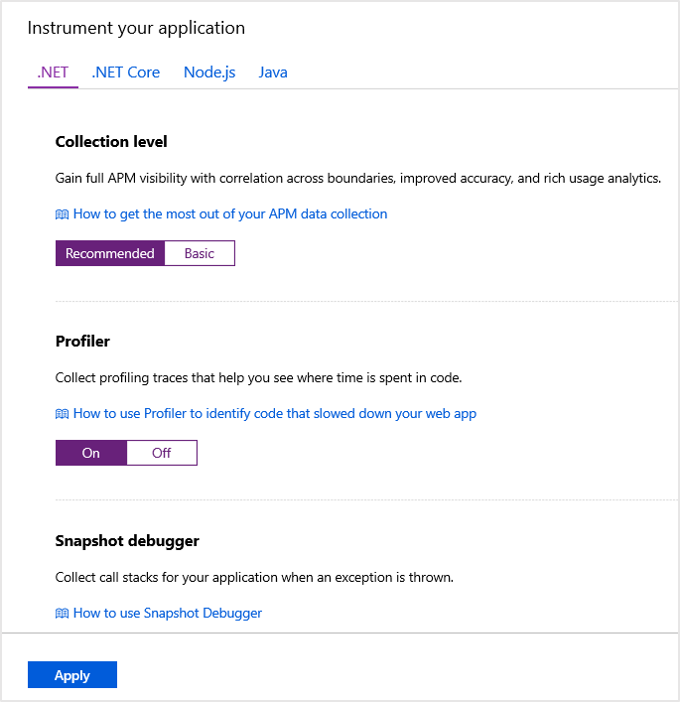
* + Choose to create a new resource, unless you already set up an Application Insights resource for this application.

**Note**

When you click **OK** to create the new resource you will be prompted to **Apply monitoring settings**. Selecting **Continue** will link your new Application Insights resource to your app service, doing so will also **trigger a restart of your app service**.



1. After specifying which resource to use, you can choose how you want application insights to collect data per platform for your application. ASP.NET app monitoring is on-by-default with two different levels of collection.



Below is a summary of data collected for each route:

| **TABLE 1** | | |
| --- | --- | --- |
| **Data** | **.NET Basic Collection** | **.NET Recommended collection** |
| Adds CPU, memory, and I/O usage trends | Yes | Yes |
| Collects usage trends, and enables correlation from availability results to transactions | Yes | Yes |
| Collects exceptions unhandled by the host process | Yes | Yes |
| Improves APM metrics accuracy under load, when sampling is used | Yes | Yes |
| Correlates micro-services across request/dependency boundaries | No (single-instance APM capabilities only) | Yes |

1. To configure settings like sampling, which you could previously control via the applicationinsights.config file you can now interact with those same settings via Application settings with a corresponding prefix.
   * For example, to change the initial sampling percentage, you can create an Application setting of: MicrosoftAppInsights\_AdaptiveSamplingTelemetryProcessor\_InitialSamplingPercentage and a value of 100.
   * For the list of supported adaptive sampling telemetry processor settings, you can consult the [code](https://github.com/microsoft/ApplicationInsights-dotnet/blob/master/BASE/Test/ServerTelemetryChannel.Test/TelemetryChannel.Tests/AdaptiveSamplingTelemetryProcessorTest.cs) and [associated documentation](https://docs.microsoft.com/en-us/azure/azure-monitor/app/sampling).

**Enable client-side monitoring**

* [.NET](https://docs.microsoft.com/en-us/azure/azure-monitor/app/azure-web-apps?tabs=net#tabpanel_CeZOj-G++Q-1_net)
* [.NET Core](https://docs.microsoft.com/en-us/azure/azure-monitor/app/azure-web-apps?tabs=net#tabpanel_CeZOj-G++Q-1_netcore)
* [Node.js](https://docs.microsoft.com/en-us/azure/azure-monitor/app/azure-web-apps?tabs=net#tabpanel_CeZOj-G++Q-1_nodejs)
* [Java](https://docs.microsoft.com/en-us/azure/azure-monitor/app/azure-web-apps?tabs=net#tabpanel_CeZOj-G++Q-1_java)
* [Python](https://docs.microsoft.com/en-us/azure/azure-monitor/app/azure-web-apps?tabs=net#tabpanel_CeZOj-G++Q-1_python)

Client-side monitoring is opt-in for ASP.NET. To enable client-side monitoring:

* **Settings** **>** **Configuration**
  + Under Application settings, create a **new application setting**:

Name: APPINSIGHTS\_JAVASCRIPT\_ENABLED

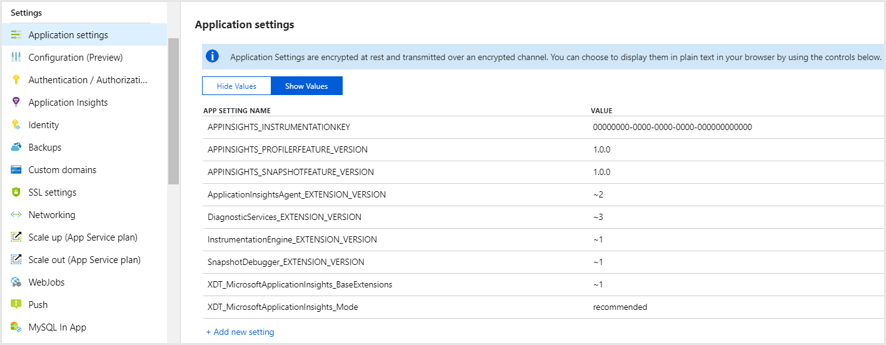
Value: true

* + **Save** the settings and **Restart** your app.

To disable client-side monitoring either remove the associated key value pair from the Application settings, or set the value to false.

**Automate monitoring**

In order to enable telemetry collection with Application Insights, only the Application settings need to be set:



**Application settin**