At the end of this lab, you can skip the **Clean Up** exercise directing you to remove the resources from your Subscription or Resource Group(s). The clean up is handled automatically, after ending your lab.

more...

Lab: Monitoring services that are deployed to Azure

Student lab manual

Lab scenario

You have created an API for your next big startup venture. Even though you want to get to market quickly, you have witnessed other ventures fail when they don’t plan for growth and have too few resources or too many users. To plan for this, you have decided to take advantage of the scale-out features of Microsoft Azure App Service, the telemetry features of Application Insights, and the performance-testing features of Azure DevOps.

Objectives

After you complete this lab, you will be able to:

- Create an Application Insights resource.
- Integrate Application Insights telemetry tracking into an ASP.NET web app and a resource built using the Web Apps feature of Azure App Service.

Lab setup

- Estimated time: **45 minutes**

Instructions

Before you start

Sign in to the lab virtual machine

Ensure that you're signed in to your Windows 10 virtual machine by using the following credentials:

- Username: **Admin**
- Password: **Pa55w.rd**

Review the installed applications

Find the taskbar on your Windows 10 desktop. The taskbar contains the icons for the applications that you'll use in this lab:

- Microsoft Edge
- File Explorer
- Visual Studio Code
- Windows PowerShell

Exercise 1: Create and configure Azure resources

Task 1: Open the Azure portal

1. On the taskbar, select the **Microsoft Edge** icon.
2. Sign in to the Azure portal (https://portal.azure.com).
3. At the sign-in page, enter the email address for your Microsoft account, and then select **Next**.
4. Enter the password for your Microsoft account, and then select **Sign in**.

   Note: If this is your first time signing in to the Azure portal, a dialog box will display offering a tour of the portal. Select **Get Started** to skip the tour and begin using the portal.

Task 2: Create an Application Insights resource

1. Create a new Application Insights account with the following details:
   - New resource group: **MonitoredAssets**
   - Name: **instrm*[yourname]***
   - Region: **(US) East US**
   - Resource Mode: **Classic**

   Note: Wait for Azure to finish creating the storage account before you move forward with the lab. You'll receive a notification when the account is created.

2. Access the **Properties** section of the **Application Insights** blade.
3. Get the value of the **Instrumentation Key** text box. This key is used by client applications to connect to Application Insights.

Task 3: Create a web app by using Azure App Services resource

1. Create a new **web app** with the following details:
   - Existing resource group: **MonitoredAssets**
   - Web App name: ***smpapi***[yourname]***
   - Publish: **Code**
   - Runtime stack: **.NET Core 3.1 (LTS)**
   - Operating System: **Windows**
   - Region: **East US**
   - New App Service plan: **MonitoredPlan**
   - SKU and size: **Standard (S1)**
   - Application Insights: **Enabled**
   - Existing Application Insights resource: **instrm*[yourname]***

Note: Wait for Azure to finish creating the web app before you move forward with the lab. You'll receive a notification when the app is created.

2. Access the web app with a prefix of **smpapi*** that you created earlier in this lab.
3. In the **Settings** section, go to the **Configuration** section.
4. Find and access the **Application Settings** tab in the **Configuration** section.
5. Get the value corresponding to the **APPINSIGHTS_INSTRUMENTATIONKEY** application settings key. This value was set automatically when you built your web app.
6. Access the **Properties** section of the **App Service** blade.
7. Record the value of the **URL** text box. You'll use this value later in the lab to make requests against the API.

Task 4: Configure web app autoscale options

1. Go to the **Scale out** section of the **App Services** blade.
2. In the **Scale out** section, enable **Custom autoscale** with the following details:
    - Name: **ComputeScaler**
    - In the **Scale mode** section, select **Scale based on a metric**.
    - Minimum instances: **2**
    - Maximum instances: **8**
    - Default instances: **3**
    - Scale rules: **Single scale-out rule with default values**
3. Save your changes to the **autoscale** configuration.

Review

In this exercise, you created the resources that you'll use for the remainder of the lab.

Exercise 2: Monitor a local web application by using Application Insights

Task 1: Build a .NET Web API project

1. Open Visual Studio Code.
2. In Visual Studio Code, open the **Allfiles (F):\Allfiles\Labs\12\Starter\Api** folder.
3. Use the Explorer pane in Visual Studio Code to open a new terminal that has the context set to the current working directory.
4. At the command prompt, create a new .NET Web API application named **SimpleApi** in the current directory:

```
dotnet new webapi --output . --name SimpleApi
```

5. Import version 2.14.0 of **Microsoft.ApplicationInsights** from NuGet to the current project:

```
dotnet add package Microsoft.ApplicationInsights --version 2.14.0
```

Note: The **dotnet add package** command will add the **Microsoft.ApplicationInsights**package from NuGet. For more information, go to Microsoft.ApplicationInsights.

6. Import version 2.14.0 of **Microsoft.ApplicationInsights.AspNetCore** from NuGet to the current project:

   ```
   dotnet add package Microsoft.ApplicationInsights.AspNetCore --version 2.14
   ```

   Note: The **dotnet add package** command will add the **Microsoft.ApplicationInsights.AspNetCore** package from NuGet. For more information, go to Microsoft.ApplicationInsights.AspNetCore.

7. Import version 2.14.0 of **Microsoft.ApplicationInsights.PerfCounterCollector** from NuGet to the current project:

   ```
   dotnet add package Microsoft.ApplicationInsights.PerfCounterCollector  --v
   ```

   Note: The **dotnet add package** command will add the **Microsoft.ApplicationInsights.PerfCounterCollector** package from NuGet. For more information, go to Microsoft.ApplicationInsights.PerfCounterCollector.

8. Build the .NET web app:

   ```
   dotnet build
   ```

Task 2: Update application code to disable HTTPS and use Application Insights

1. Use the Explorer in Visual Studio Code to open the **Startup.cs** file in the editor.

2. Find and delete the following line of code at line 39:

   ```
   1. []UseHttpsRedirection();
   ```

   Note: This line of code forces the web app to use HTTPS. For this lab, this is unnecessary.

3. In the **Startup** class, add a new static string constant named **INSTRUMENTATION_KEY** with its value set to the instrumentation key that you copied from the Application Insights resource you created earlier in this lab:

   ```
   private static string INSTRUMENTATION_KEY = "{your_instrumentation_key}";
   ```

   Note: For example, if your instrumentation key is `d2bb0eed-1342-4394-9b0c-8a56d21aaa43` , your line of code would be `private static string INSTRUMENTATION_KEY = "d2bb0eed-1342-4394-9b0c-8a56d21aaa43";`

4. Add a new line of code in the **ConfigureServices** method to configure Application Insights using the provided instrumentation key:

   ```
   services.AddApplicationInsightsTelemetry(INSTRUMENTATION_KEY);
   ```

5. Save the **Startup.cs** file.

6. If it's not already open, use the Explorer in Visual Studio Code to open a new terminal with the context set to the current working directory.

7. Build the .NET web app:

   ```
   dotnet build
   ```

Task 3: Test an API application locally

1. If it's not already open, use the Explorer in Visual Studio Code to open a new terminal with the context set to the current working directory.

2. Run the .NET web app.

   ```
   dotnet run
   ```

3. Open a new **Microsoft Edge** browser window.

4. In the open browser window, go to the **/weatherforecast** relative path of your test application that's hosted at **localhost** on port **5000**.

   > **Note**: The full URL is http://localhost:5000/weatherforecast.

5. Close the browser window that you recently opened.

6. Close the currently running Visual Studio Code application.

Task 4: Get metrics in Application Insights

1. Return to your currently open browser window that's displaying the Azure portal.

2. Access the **instrm*[yourname]*** Application Insights account that you created earlier in this lab.

3. From the **Application Insights** blade, find the metrics displayed in the tiles in the center of the blade. Specifically, find the number of server requests that have occurred and the average server response time.

   > **Note**: It can take up to five minutes to observe requests in the Application Insights metrics charts.

Review

In this exercise, you created an API by using ASP.NET and configured it to stream application metrics to Application Insights. You then used the Application Insights dashboard to get performance details about your API.

Exercise 3: Monitor a web app using Application Insights

Task 1: Deploy an application to the web app

1. Open Visual Studio Code.
2. In Visual Studio Code, open the **Allfiles (F):\Allfiles\Labs\12\Starter\Api** folder.
3. Use the Explorer in Visual Studio Code to open a new terminal with the context set to the current working directory.
4. Sign in to the Azure Command-Line Interface (CLI) by using your Azure credentials:

   ```
   az login
   ```

5. List all the apps in your **MonitoredAssets** resource group:

   ```
   az webapp list --resource-group MonitoredAssets
   ```

6. Find the apps that have the prefix **smpapi***:

   ```
   az webapp list --resource-group MonitoredAssets --query "[?starts_with(nam
   ```

7. Print only the name of the single app that has the prefix **smpapi***:

   ```
   az webapp list --resource-group MonitoredAssets --query "[?starts_with(nam
   ```

8. Change the current directory to the **Allfiles (F):\Allfiles\Labs\12\Starter** directory that contains the lab files:

   ```
   cd F:\Allfiles\Labs\12\Starter\
   ```

9. Deploy the **api.zip** file to the web app that you created earlier in this lab:

   ```
   az webapp deployment source config-zip --resource-group MonitoredAssets --
   ```

   Note: Replace the *name-of-your-api-app* placeholder with the name of the web app that you created earlier in this lab. You recently queried this app's name in the previous steps.

10. Return to your currently open browser window that's displaying the Azure portal.
11. Access the ***smpapi***[yourname]*** web app that you created earlier in this lab.
12. Open the ***smpapi***[yourname]*** web app in your browser.
13. Perform a GET request to the **/weatherforecast** relative path of the website, and then find the JavaScript Object Notation (JSON) array that's returned as a result of using the API.

    Note: For example, if your URL is https://smpapistudent.azurewebsites.net, the new URL would be https://smpapistudent.azurewebsites.net/weatherforecast.

Task 2: Configure in-depth metric collection for Web Apps

1. Return to your currently open browser window that's displaying the Azure portal.

2. Access the ***smpapi***[yourname]*** web app that you created earlier in this lab.

3. Go to the **Application Insights** configuration section.

4. **Enable** Application Insights instrumentation for **.NET** by using the following settings:

   - Collection level: **Recommended**
   - Profiler: **On**
   - Snapshot debugger: **Off**
   - SQL Commands: **Off**

5. Save your Application Insights instrumentation configuration.

6. Open the ***smpapi***[yourname]*** web app in your browser.

7. Perform a GET request to the **/weatherforecast** relative path of the website, and then observe the JSON array that's returned as a result of using the API.

   > **Note**: For example, if your URL is https://smpapistudent.azurewebsites.net, the new URL would be https://smpapistudent.azurewebsites.net/weatherforecast.

8. Record the URL that you used to access the JSON array.

   > **Note**: Using the example from the previous step, you would record the URL `https://smpapistudent.azurewebsites.net/weatherforecast`.

Task 3: Get updated metrics in Application Insights

1. Return to your currently open browser window that's displaying the Azure portal.

2. Access the **instrm*[yourname]*** Application Insights account that you created earlier in this lab.

3. From the **Application Insights** blade, find the metrics displayed in the tiles in the center of the blade. Specifically, find the number of server requests that have occurred and the average server response time.

   > **Note**: It can take up to five minutes to observe requests in the Application Insights metrics charts.

Task 4: View real-time metrics in Application Insights

1. Return to your currently open browser window that's displaying the Azure portal.

2. Access the **instrm*[yourname]*** Application Insights account that you created earlier in this lab.

3. Open the **Live Metrics Stream** blade for your account.

4. Open a new **Microsoft Edge** browser window, go to the URL that you recorded earlier in this lab, and then observe the JSON array result.

5. Return to your currently open browser window that's displaying the Azure portal, and then observe the updated **Live Metrics Stream** blade.

   > **Note**: The **Incoming Requests** section should update in seconds, showing the requests that you made to the web app.

Review

In this exercise, you deployed your web app to App Service and monitored your metrics from the same Application Insights instance.

Exercise 4: Clean up your subscription

Task 1: Open Azure Cloud Shell

1. In the Azure portal, select the **Cloud Shell** icon to open a new shell instance.
2. If Cloud Shell isn't already configured, configure the shell for Bash by using the default settings.

Task 2: Delete resource groups

1. Enter the following command, and then select Enter to delete the **MonitoredAssets** resource group:

   ```
   az group delete --name MonitoredAssets --no-wait --yes
   ```

2. Close the Cloud Shell pane in the portal.

Task 3: Close the active applications

1. Close the currently running Microsoft Edge application.
2. Close the currently running Visual Studio Code application.

Review

In this exercise, you cleaned up your subscription by removing the resource groups used in this lab.

Congratulations!

You have successfully completed this exercise. Click **End** to advance to the next exercise.