Task 1: Create a function app

- In the Azure portal, create a new **function app** with the following details:
  - **App name**: JourneyThumbsFunc
  - **Existing resource group**: JourneyWeb
  - **OS**: Windows
  - **Hosting Plan**: Consumption Plan
  - **Location**: East US
  - **Runtime Stack**: .NET
  - **Storage**: JourneyPhotoStore

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

Task 2: Author a function to process blobs

1. On the Azure portal left navigation pane, select **Resource groups**.
2. In the **Resource groups** blade, locate and select the **JourneyWeb** resource group that you created earlier in this lab.
3. In the **JourneyWeb** blade, select the **JourneyThumbsFunc** function app that you created earlier in this lab.
4. In the **Function App** blade, select **+ New function**.
5. In the **New Azure Function** quickstart, perform the following actions:
   1. Under the **Choose a Development Environment** header, select **In-Portal**.
   2. Select **Continue**.
   3. Under the **Create a Function** header, select **More templates…**.
   4. Select **Finish and view templates**.
   5. In the **Templates** list, select **Azure Blob Storage trigger**.
   6. In the **Extensions not Installed** window, select **Install**.

Note: It can take up to two minutes to install the extensions needed to work with Azure Storage blobs. If the portal does not refresh, simply close the **Extensions not Installed** pop-up window and select **Azure Blob Storage trigger** again.

7. Once the installation has succeeded, select **Continue**.
8. In the **New Function** window, in the **Name** field enter **ImageManager**.
9. In the **New Function** window, in the **Path** field enter **images/{name}**.
10. In the **New Function** window, in the **Storage account connection** list, select **AzureWebJobsStorage**.
11. In the **New Function** window, select **Create**.
6. On the right side of the function editor, select **View files** to open the tab.
7. In the **View files** tab, select **Upload**.

8. In this folder open **function.proj** file, and then select **Open**.
9. Back in the **View files** tab, select the **function.json** file to view the editor for the function's configuration.
10. Replace the entire contents of the JSON configuration file with the following JSON content:

```
{
  "bindings": [
    {
      "name": "inputBlob",
      "type": "blobTrigger",
      "direction": "in",
      "path": "images/{name}",
      "connection": "AzureWebJobsStorage"
    },
    {
      "type": "blob",
      "name": "outputBlob",
      "path": "images-thumbnails/{name}",
      "connection": "AzureWebJobsStorage",
      "direction": "out"
    }
  ]
}
```

12. In the JSON editor, select **Save** button to persist your changes to the configuration.
13. Back in the **View files** tab, select the **run.csx** file to return to the editor for the **ImageManager** function.
16. Within the editor, copy and paste the following code:

```
using SixLabors.ImageSharp;
using SixLabors.ImageSharp.PixelFormats;
using SixLabors.ImageSharp.Processing;
using SixLabors.ImageSharp.Formats.Jpeg;
using SixLabors.Primitives;

public static void Run(Stream inputBlob, Stream outputBlob, string name, ILogger log)
{
    log.LogInformation($"C# Blob trigger function Processed blob\n Name:{name} \n Size: {inputB
lob.Length} Bytes");
    using (Image<Rgba32> image = Image.Load(inputBlob))
    {
        image.Mutate(i =>
            i.Resize(new ResizeOptions { Size = new Size(200, 200), Mode = ResizeMode.Max })
        );
        image.Save(outputBlob, new JpegEncoder());
    }
}
```

24. In the editor, select **Save** to save the script and compile the code again.