**Lab: Building a web application on Azure Platform-as-a-Service offerings**

**Student lab answer key**

**Exercise 1: Build a back-end API by using Azure Storage and API Apps**

**Task 1: Open the Azure portal**

1. On the taskbar, select the **Microsoft Edge** icon.
2. In the open browser window, navigate to the [**Azure portal**](https://portal.azure.com/) (portal.azure.com).
3. At the sign-in page, enter the **email address** for your Microsoft account.
4. Select **Next**.
5. Enter the **password** for your Microsoft account.
6. 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 Azure Storage account**

1. In the left navigation pane of the Azure portal, select **All services**.
2. In the **All services** blade, select **Storage Accounts**.
3. In the **Storage accounts** blade, view your list of Storage instances.
4. At the top of the **Storage accounts** blade, select **Add**.
5. In the **Create storage account** blade, observe the tabs at the top of the blade, such as Basics, Tags, and Review+Create.

**Note**: Each tab represents a step in the workflow to create a new **storage account**. At any time, you can select **Review + create** to skip the remaining tabs.

1. Select the **Basics** tab, and within the tab area, perform the following actions:
   1. Leave the **Subscription** field set to its default value.
   2. In the **Resource group** section, select **Create new**, enter **ManagedPlatform**, and then select **OK**.
   3. In the **Storage account** **name** field, enter **imgstor[*your name in lowercase*]**.
   4. In the **Location** list, select the **(US) East US** region.
   5. In the **Performance** section, select **Standard**.
   6. In the **Account kind** list, select **StorageV2 (general purpose v2)**.
   7. In the **Replication** list, select **Locally-redundant storage (LRS)**.
   8. In the **Access tier (default)** section, ensure that **Hot** is selected.
   9. Select **Review + Create**.
2. In the **Review + Create** tab, review the options that you specified in the previous steps.
3. Select **Create** to create the storage account by using your specified configuration.
4. In the **Deployment** blade, Wait for the creation task to complete before moving forward with this lab.
5. Click the **Go to resource** button in the **Deployment** blade to go to the newly created storage account.
6. In the **Storage account** blade, on the left side of the blade, locate the **Settings** section and select **Access keys**.
7. In the **Access keys** blade, select any one of the keys and record the value of either of the **Connection string** fields. You will use this value later in this lab.

**Note**: It does not matter which connection string you choose. They are interchangeable.

**Task 3: Upload a sample blob**

1. On the Azure portal left navigation pane, select **Resource groups**.
2. In the **Resource groups** blade, select the **ManagedPlatform** resource group that you created earlier in this lab.
3. In the **ManagedPlatform** blade, select the **imgstor\*** storage account you created earlier in this lab.
4. In the **Storage Account** blade, on the left side of the blade, in the **Blob service** section, select the **Blobs** link.
5. In the **Blobs** section, select **+ Container**.
6. In the **New container** window, perform the following actions:
   1. In the **Name** field, enter **images**.
   2. In the **Public access level** list, select **Blob (anonymous read access for blobs only)**.
   3. Select **OK**.
7. In the **Blobs** section, select **+ Container** again.
8. In the **New container** window, perform the following actions:
   1. In the **Name** field, enter **images-thumbnails**.
   2. In the **Public access level** list, select **Blob (anonymous read access for blobs only)**.
   3. Select **OK**.
9. In the **Blobs** section, select the newly created **images** container.
10. In the **Container** blade, select **Upload**.
11. In the **Upload blob** window that appears, perform the following actions:
    1. In the **Files** section, select the **Folder** icon.
    2. In the File Explorer dialog box that appears, go to Images folder and select the **grilledcheese.jpg** file from the lab material that you downloaded from drive, and then select **Open**.
    3. Ensure that the **Overwrite if files already exist** check box is selected.
    4. Select **Upload**.
12. Wait for the blob to be uploaded before you continue with this lab.

**Task 4: Create an API app**

1. In the left navigation pane of the portal, select **+ Create a resource**.
2. At the top of the **New** blade, locate the **Search the Marketplace** field.
3. In the search field, enter **API** and press Enter.
4. In the **Everything** search results blade, select the **API App** result.
5. In the **API App** blade, select **Create**.
6. In the second **API App** blade, perform the following actions:
   1. In the **App name** field, enter **imgapi[*your name in lowercase*]**.
   2. Leave the **Subscription** field set to its default value.
   3. In the **Resource group** section, select **Use existing**, and then select **ManagedPlatform**.
   4. Leave the **App Service plan/Location** field set to its default value.
   5. Leave the **Application Insights** field set to its default value.
   6. Select **Create**.
7. Wait for the creation task to complete before you move forward with this lab.

**Task 5: Configure an API app**

1. In the left navigation pane of the portal, select **Resource groups**.
2. In the **Resource groups** blade, select the **ManagedPlatform** resource group that you created earlier in this lab.
3. In the **ManagedPlatform** blade, select the **imgapi\*** API app that you created earlier in this lab.
4. In the **API App** blade, on the left side of the blade in the **Settings** section, select the **Configuration** link.
5. In the **Configuration** section, perform the following actions:
   1. Select the **Application settings** tab.
   2. Select **+ New application setting**.
   3. In the **Add/Edit application setting** popup that appears, in the **Name** field, enter **StorageConnectionString**.
   4. In the **Value** field, enter the **Storage Connection String** you copied earlier in this lab.
   5. Leave the **deployment slot setting** field set to its default value.
   6. Select **OK** to close the popup and return to the **Configuration** section.
   7. Select **Save** at the top of the blade to persist your settings.
6. Wait for your application settings to persist before you move forward with the lab.
7. In the **API App** blade, on the left side of the blade in the **Settings** section, select the **Properties** link.
8. In the **Properties** section, copy the value of the **URL** field. You will use this value later in the lab.

**Task 6: Deploy an ASP.NET Core web application to API App**

1. Open visual studio.
2. On the **File** menu, select **Open Folder**.
3. In the File Explorer pane that opens, go to lab files and open **API** folder, and then select **Select Folder**.
4. In the **Solution Explorer** pane of the Visual Studio window, expand the **Controllers** folder and double-click the **ImagesController.cs** file to open the file in the editor.
5. In the editor, in the **ImagesController** class, on line 27, observe the **GetCloudBlobContainer** method and the code used to retrieve a container.
6. In the **ImagesController** class, on line 38, observe the **Get** method and the code used to retrieve all blobs asynchronously from the **images** container.
7. In the **ImagesController** class, on line 74, observe the **Post** method and the code used to persist an uploaded image to Azure Storage.
8. Now open CMD installed with Azure PoerShell CLI and enter the following command and press Enter to sign in to the Azure CLI:

az login

1. In the **Microsoft Edge** browser window that appears, perform the following actions:
   1. Enter the **email address** for your Microsoft account.
   2. Select **Next**.
   3. Enter the **password** for your Microsoft account.
   4. Select **Sign in**.
2. Return to the currently open **Command Prompt** application. Wait for the sign-in process to finish.
3. At the command prompt, enter the following command and press Enter to list all the **apps** in your **ManagedPlatform** resource group:

az webapp list --resource-group ManagedPlatform

1. Enter the following command and press Enter to find the **apps** that have the prefix **imgapi\***:

az webapp list --resource-group ManagedPlatform --query "[?starts\_with(name, 'imgapi')]"

1. Enter the following command and press Enter to print out only the name of the single app that has the prefix **imgapi\***:

az webapp list --resource-group ManagedPlatform --query "[?starts\_with(name, 'imgapi')].{Name:name}" --output tsv

1. Enter the following command and press Enter to change the current directory to the **Lab Material\API** directory that contains the lab files:

cd Lab Material\API\

1. Enter the following command and press Enter to deploy the **api.zip** file to the **API app** you created earlier in this lab:
2. az webapp deployment source config-zip --resource-group ManagedPlatform --src api.zip --name <name-of-your-api-app>

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

1. Wait for the deployment to complete before you move forward with this lab.
2. On the left side of the portal, select the **Resource groups** link.
3. In the **Resource groups** blade, locate and select the **ManagedPlatform** resource group that you created earlier in this lab.
4. In the **ManagedPlatform** blade, select the **imgapi\*** API App that you created earlier in this lab.
5. In the **API App** blade, select the **Browse** button.
6. Perform a **GET** request to the root of the website and observe the JSON array that is returned. This array should contain the URL for your single uploaded image in your **Azure Storage** account.
7. Return to your browser window showing the **Azure portal**.

**Review**

In this exercise, you created an API App in Azure and then deployed your ASP.NET Core web application to the API App by using the Azure CLI and Kudu’s zip deployment utility.

**Exercise 2: Build a front-end web application by using Azure Web Apps**

**Task 1: Create a web app**

1. In the Azure portal, on the left navigation pane, select **+ Create a resource**.
2. At the top of the **New** blade, locate the **Search the Marketplace** field.
3. In the search field, enter **Web** and press Enter.
4. In the **Everything** search results blade, select the **Web App** result.
5. In the **Web App** blade, select **Create**.
6. In the second **Web App** blade, perform the following actions:
   1. In the **App name** field, enter **imgweb[*your name in lowercase*]**.
   2. Leave the **Subscription** field set to its default value.
   3. In the **Resource group** section, select **Use existing**, and then select **ManagedPlatform**.
   4. In the **Publish** section, select **Code**.
   5. In the **Runtime stack** section, select **.NET Core 2.2**.
   6. In the **OS** section, select **Windows**.
   7. In the **Region** drop-down list, select **East US**
   8. Leave the **Plan (East US)** field set to its default value.
   9. Leave the **Sku and size** field set to its default value.
   10. Select **Review and create**.
7. In the **Review and create** tab, observe the settings then click **Create**.
8. Wait for the creation task to complete before you move forward with this lab.

**Task 2: Configure a web app**

1. On the left navigation pane of the portal, select **Resource groups**.
2. In the **Resource groups** blade, select the **ManagedPlatform** resource group that you created earlier in this lab.
3. In the **ManagedPlatform** blade, select the **imgweb\*** web app that you created earlier in this lab.
4. In the **Web App** bladeblade, on the left side of the blade in the **Settings** section, select the **Configuration** link.
5. In the **Configuration** section, perform the following actions:
   1. Select the **Application settings** tab.
   2. Select **+ New application setting**.
   3. In the **Add/Edit application setting** popup that appears, in the **Name** field, enter **ApiUrl**.
   4. In the **Value** field, enter the API app **URL** you copied earlier in this lab.
   5. Leave the **deployment slot setting** field set to its default value.
   6. Select **OK** to close the popup and return to the **Configuration** section.
   7. Select **Save** at the top of the blade to persist your settings.
6. Wait for your application settings to persist before you move forward with the lab.

**Task 3: Deploy an ASP.NET Core web application to web app**

1. On the taskbar, select the **Visual Studio Code** icon.
2. On the **File** menu, select **Open Folder**.
3. In the File Explorer pane that opens, go to **Lab Material\Web**, and then select **Select Folder**.
4. In the **Explorer** pane of the Visual Studio Code window, expand the **Pages** folder and double-click the **Index.cshtml.cs** file to open the file in the editor.
5. In the editor, in the **IndexModel** class, on line 30, observe the **OnGetAsync** method and the code used to retrieve the list of images from the API.
6. In the **IndexModel** class, on line 52, observe the **OnPostAsync** method and the code used to stream an uploaded image to the back-end API.
7. On the taskbar, select the **Windows** **PowerShell** icon.
8. In the open command prompt, enter the following command and press Enter to sign in to the Azure CLI:
9. az login
10. In the browser window that appears, perform the following actions:
    1. Enter the **email address** for your Microsoft account.
    2. Select **Next**.
    3. Enter the **password** for your Microsoft account.
    4. Select **Sign in**.
11. Return to the currently open **Command Prompt** application. Wait for the sign-in process to finish.
12. Enter the following command and press Enter to list all the **apps** in your **ManagedPlatform** resource group:

az webapp list --resource-group ManagedPlatform

1. Enter the following command and press Enter to find the **apps** that have the prefix **imgweb\***:

az webapp list --resource-group ManagedPlatform --query "[?starts\_with(name, 'imgweb')]"

1. Enter the following command and press Enter to print out only the name of the single app that has the prefix **imgweb\***:

az webapp list --resource-group ManagedPlatform --query "[?starts\_with(name, 'imgweb')].{Name:name}" --output tsv

1. Enter the following command and press Enter to change the current directory to the **Lab Material\Web** directory that contains the lab files:

cd F:\Lab Material\Web\

1. Enter the following command and press Enter to deploy the **web.zip** file to the **web app** you created earlier in this lab:

az webapp deployment source config-zip --resource-group ManagedPlatform --src web.zip --name <name-of-your-web-app>

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

1. Wait for the deployment to complete before you move forward with this lab.
2. On the left navigation pane of the portal, select **Resource groups**.
3. In the **Resource groups** blade, select the **ManagedPlatform** resource group you created earlier in this lab.
4. In the **ManagedPlatform** blade, select the **imgweb\*** web app that you created earlier in this lab.
5. In the **Web App** blade, select **Browse**.
6. Observe the list of images in the gallery. The gallery should list a single image that was uploaded to Azure Storage earlier in the lab.
7. At the top of the **Contoso Photo Gallery** webpage, locate the **Upload a new image** section and perform the following actions:
   1. Select **Browse**.
   2. In the File Explorer dialog box that opens, go to  **Lab Material \Images**, select the **bahnmi.jpg** file, and then select **Open**.
   3. Select **Upload**.
8. Observe that the list of gallery images has been updated with your new image.

**Note**: In some rare cases, you might need to refresh your browser window for the new image to appear.

1. Return to your browser window showing the **Azure portal**.

**Review**

In this exercise, you created an Azure Web App and deployed an existing web application’s code to the resource in the cloud.

**Exercise 3: Build a background processing job by using Azure Storage and Azure Functions**

**Task 1: Create a function app**

1. On the Azure portal left navigation pane, select **+ Create a resource**.
2. At the top of the **New** blade, locate the **Search the Marketplace** field.
3. In the search field, enter **Function** and press Enter.
4. In the **Everything** search results blade, select the **Function App** result.
5. In the **Function App** blade, select **Create**.
6. In the second **Function App** blade, perform the following actions:
   1. In the **App name** field, enter **imgfunc[*your name in lowercase*]**.
   2. Leave the **Subscription** field set to its default value.
   3. In the **Resource group** section, select **Use existing**, and then select **ManagedPlatform**.
   4. In the **OS** section, select **Windows**.
   5. In the **Hosting Plan** list, select **Consumption Plan**.
   6. In the **Location** list, select **East US**.
   7. In the **Runtime Stack** list, select **.NET Core**.
   8. In the **Storage** section, select **Use existing**, and then select the **imgstor\*** storage account you created earlier in this lab.
   9. Leave the **Application Insights** field set to its default value.
   10. Select **Create**.
7. Wait for the creation task to complete before moving on with this lab.

**Task 2: Create a .NET Core application setting**

1. On the navigation menu located on the left side of the portal, select the **Resource groups** link.
2. In the **Resource groups** blade, locate and select the **ManagedPlatform** resource group that you created earlier in this lab.
3. In the **ManagedPlatform** blade, select the **imgfunc\*** function app that you created earlier in this lab.
4. In the **Function App** blade, select the **Platform features** tab.
5. In the **Platform features** tab, select the **Configuration** link located in the **General Settings** section.
6. In the **Configuration** section, perform the following actions:
   1. Select the **Application settings** tab.
   2. Select **+ New application setting**.
   3. In the **Add/Edit application setting** popup that appears, in the **Name** field, enter **DOTNET\_SKIP\_FIRST\_TIME\_EXPERIENCE**.
   4. In the **Value** field, enter **true**.

**Note**: The DOTNET\_SKIP\_FIRST\_TIME\_EXPERIENCE application setting tells .NET Core to disable it's built-in NuGet package caching mechanisms. On a temporary compute instance, this would effectively be a waste of time and cause build issues with your Azure Function.

* 1. Leave the **deployment slot setting** field set to its default value.
  2. Select **OK** to close the popup and return to the **Configuration** section.
  3. Select **Save** at the top of the blade to persist your settings.

1. Wait for your application settings to persist before you move forward with the lab.

**Task 3: Author a function to process blobs**

1. On the left navigation pane of the portal, select **Resource groups**.
2. In the **Resource groups** blade, locate and select the **ManagedPlatform** resource group that you created earlier in this lab.
3. In the **ManagedPlatform** blade, select the **imgfunc\*** 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.

* 1. Once the installation has succeeded, select **Continue**.
  2. In the **New Function** window, in the **Name** field, enter **ImageManager**.
  3. In the **New Function** window, in the **Path** field, enter **images/{name}**.
  4. In the **New Function** window, in the **Storage account connection** list, select **AzureWebJobsStorage**.
  5. In the **New Function** window, select **Create**.

1. On the right side of the function editor, select **View files** to open the tab.
2. In the **View files** tab, select **Add**.
3. In the filename dialog that appears, enter **function.proj**.
4. In the file editor, insert this configuration content:

<Project Sdk="Microsoft.NET.Sdk">

<PropertyGroup>

<TargetFramework>netstandard2.0</TargetFramework>

</PropertyGroup>

<ItemGroup>

<PackageReference Include="SixLabors.ImageSharp" Version="1.0.0-beta0006" />

</ItemGroup>

</Project>

1. In the editor, select **Save** button to persist your changes to the configuration.

**Note**: This **.proj** file contains the NuGet package reference necessary to import the [SixLabors.ImageSharp](https://www.nuget.org/packages/SixLabors.ImageSharp/1.0.0-beta0006) package.

1. Back in the **View files** tab, select the **function.json** file to view the editor for the function’s configuration.
2. In the JSON editor, observe the current configuration:

{

"bindings": [

{

"name": "myBlob",

"type": "blobTrigger",

"direction": "in",

"path": "images/{name}",

"connection": "AzureWebJobsStorage"

}

],

"disabled": false

}

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"

}

]

}

1. In the editor, select **Save** to persist your changes to the configuration.
2. Back in the **View files** tab, select the **run.csx** file to return to the editor for the **ImageManager** function.
3. Minimize the **View files** tab.

**Note**: You can minimize the tab by selecting the arrow immediately to the right of the tab header.

1. In the function editor, observe the example function script:

public static void Run(Stream myBlob, string name, ILogger log)

{

log.LogInformation($"C# Blob trigger function Processed blob\n Name:{name} \n Size: {myBlob.Length} Bytes");

}

1. **Delete** all the example code.
2. Within the editor, copy and paste the following placeholder function:

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)

{

}

1. Select **Save** to save the script and compile the code.
2. Add the following line of code within the **Run** method to log information about the function execution:

log.LogInformation($"C# Blob trigger function Processed blob\n Name:{name} \n Size: {inputBlob.Length} Bytes");

1. Add the following **using** statement to load the **Stream** for the input blob into the image library:

using (Image<Rgba32> image = Image.Load(inputBlob))

{

}

1. Add the following lines of code within the **using** statement to mutate the image by resizing the image and applying a grayscale filter:

image.Mutate(i =>

i.Resize(new ResizeOptions { Size = new Size(250, 250), Mode = ResizeMode.Max }).Grayscale()

);

1. Add the following line of code to save the new image to the **Stream** for the output blob:
2. image.Save(outputBlob, new JpegEncoder());
3. Your **Run** method should now resemble this:

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: {inputBlob.Length} Bytes");

using (Image<Rgba32> image = Image.Load(inputBlob))

{

image.Mutate(i =>

i.Resize(new ResizeOptions { Size = new Size(250, 250), Mode = ResizeMode.Max }).Grayscale()

);

image.Save(outputBlob, new JpegEncoder());

}

}

1. Select **Save** to save the script and compile the code again.

**Task 4: Validate the web solution**

1. On the left navigation pane of the portal, select **Resource groups**.
2. In the **Resource groups** blade, select the **ManagedPlatform** resource group that you created earlier in this lab.
3. In the **ManagedPlatform** blade, select the **imgstor\*** storage account that you created earlier in this lab.
4. In the **Storage Account** blade, on the left side of the blade, in the **Blob service** section, select the **Blobs** link.
5. In the **Blobs** section, select the **images** container.
6. In the **Container** blade, select **Upload**.
7. In the **Upload blob** window that appears, perform the following actions:
   1. In the **Files** section, select the **Folder** icon.
   2. In the File Explorer dialog box that opens, go to **Allfiles (F):\Allfiles\Labs\02\Starter\Images**, select the **veggie.jpg** file, and then select **Open**.
   3. Ensure the **Overwrite if files already exist** check box is selected.
   4. Select **Upload**.
8. Wait for the blob to be uploaded before you continue with this lab.
9. Close the **Container** blade.
10. Back in the **Blobs** section, select the **images-thumbnails** container.
11. In the **Container** blade, observe the newly created **veggie.jpg** file in the **images-thumbnails** container.

**Note**: It might take one to five minutes for the new image to appear.

1. Select the **veggie.jpg** blob in the **images-thumbnails** container.
2. In the **Blob** blade, select the **Edit blob** tab.
3. Observe the contents of the blob. The webpage will render the image that was uploaded to the container.
4. On the left navigation pane of the portal, select **Resource groups**.
5. In the **Resource groups** blade, select the **ManagedPlatform** resource group that you created earlier in this lab.
6. In the **ManagedPlatform** blade, select the **imgweb\*** web app that you created earlier in this lab.
7. In the **Web App** blade, select **Browse**.
8. Observe the list of images in the gallery. The list of thumbnails should now be updated with a new thumbnail image.
9. At the top of the **Contoso Photo Gallery** webpage, locate the **Upload a new image** section and perform the following actions:
   1. Select **Browse**.
   2. In the File Explorer dialog box that opens, go to **Allfiles (F):\Allfiles\Labs\02\Starter\Images**, select the **blt.jpg** file, and then select **Open**.
   3. Select **Upload**.
10. At the top of the **Contoso Photo Gallery** webpage, locate the **Upload a new image** section and perform the following actions:
    1. Select **Browse**.
    2. In the File Explorer dialog box that opens, go to **Allfiles (F):\Allfiles\Labs\02\Starter\Images**, select the **sub.jpg** file, and then select **Open**.
    3. Select **Upload**.
11. At the top of the **Contoso Photo Gallery** webpage, locate the **Upload a new image** section and perform the following actions:
    1. Select **Browse**.
    2. In the File Explorer dialog box that opens, go to **Allfiles (F):\Allfiles\Labs\02\Starter\Images**, select the **burger.jpg** file, and then select **Open**.
    3. Select **Upload**.
12. Observe that the list of gallery images has been updated with your new image.
13. Observe the list of thumbnails at the top of the page. Refresh your page every minute until your thumbnails have been generated.

**Review**

In this exercise, you created a background processing job in Azure Functions to handle the computationally intensive task of modifying and resizing images.

**Exercise 4: Clean up subscription**

**Task 1: Open Cloud Shell**

1. At the top of the portal, select the **Cloud Shell** icon to open a new shell instance.
2. In the **Cloud Shell** command prompt at the bottom of the portal, type in the following command and press Enter to list all resource groups in the subscription:
3. az group list
4. Type in the following command and press Enter to view a list of possible commands to delete a resource group:
5. az group delete --help

**Task 2: Delete resource groups**

1. Type the following command and press Enter to delete the **ManagedPlatform** resource group:
2. az group delete --name ManagedPlatform --no-wait --yes
3. Close the **Cloud Shell** pane at the bottom of the portal.

**Task 3: Close 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.