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

V 1.4

## Microsoft Azure user interface

Given the dynamic nature of Microsoft cloud tools, you might experience Azure user interface (UI) changes after the development of this training content. These changes might cause the lab instructions and steps to not match up.

The Microsoft Worldwide Learning team updates this training course as soon as the community brings needed changes to our attention. However, because cloud updates occur frequently, you might encounter UI changes before this training content is updated. **If this occurs, adapt to the changes and work through them in the labs as needed.**

## Instructions

### Before you start

#### Review installed applications

* Following application need to be installed on your laptop and will use in this lab:
  + Microsoft Edge
  + File Explorer
  + **Visual Studio Code**

#### Lab files

* Lab files located in the folder **Starter**

### 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 **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 **Local-redundant storage (LRS)**.
   8. In the **Access tier** 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. Wait for the creation task to complete before moving forward with this lab.
5. In the left navigation pane of the Azure portal, select **All services**.
6. In the **All services** blade, select **Storage Accounts**.
7. In the **Storage accounts** blade that displays, select the **imgstor\*** storage account that you created earlier in this lab.
8. In the **Storage account** blade, on the left side of the blade, locate the **Settings** section and select **Access keys**.
9. 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:
9. In the **Name** field, enter **images-thumbnails**.
10. In the **Public access level** list, select **Blob (anonymous read access for blobs only)**.
11. Select **OK**.
12. In the **Blobs** section, select the newly created **images** container.
13. In the **Container** blade, select **Upload**.
14. In the **Upload blob** window that appears, perform the following actions:
15. In the **Files** section, select the **Folder** icon.
16. In the File Explorer dialog box that appears, go to **\Starter\Images**, select the **grilledcheese.jpg** file, and then select **Open**.
17. Ensure that the **Overwrite if files already exist** check box is selected.
18. Select **Upload**.
19. 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 **Application settings** section, perform the following actions:
   1. Scroll down until you see the **Application settings** subsection.
   2. Select **+ Add new setting**.
   3. In the **Enter a name** field, enter **StorageConnectionString**.
   4. In the **Enter a value** field, enter the **Storage Connection String** you copied earlier in this lab.
   5. Leave the **Slot Setting** field set to its default value.
   6. Select **Save** at the top of the blade.
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: Observe an ASP.NET Core web application to API 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 **\Starter\API**, and then select **Select Folder**.
4. In the **Explorer** pane of the Visual Studio Code 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 37, 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.

#### Task 7: Deploy an ASP.NET Core web application to API App

1. Return to the Portal page and open **Cloud Shell**. Select **batch** command prompt.
2. 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. Upload archive **api.zip** to the cloud shell. Button **upload** available on the top header of the Cloud Shell window.
2. Enter the following command and press Enter to deploy the **api.zip** file to the **API app** you created earlier in this lab:

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>** 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. Leave the **App Service plan/Location** field set to its default value.
   8. Leave the **Application Insights** field set to its default value.
   9. Select **Create**.
7. 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** blade, on the left side of the blade, in the **Settings** section, select the **Application settings** link.
5. In the **Application settings** section, perform the following actions:
   1. Scroll down until you see the **Application settings** subsection.
   2. Select **+ Add new setting**.
   3. In the **Enter a name** field, enter **ApiUrl**.
   4. In the **Enter a value** field, enter the API app **URL** you copied earlier in this lab.
   5. Leave the **Slot Setting** field set to its default value.
   6. Select **Save** at the top of the blade.
6. Wait for your application settings to persist before you move forward with the lab.

#### Task 3: Observe an ASP.NET Core web application to web app

1. On the taskbar, select the **Visual Studio Code**.
2. On the **File** menu, select **Open Folder**.
3. In the File Explorer pane that opens, go to **\Starter\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.

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

1. Return to the Portal page and open **Cloud Shell**. Select batch command prompt.
2. 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. Upload archive **web.zip** to the cloud shell. The button **upload** is available on the top header of the Cloud Shell window.
2. 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>** 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 **\Starter\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**.
   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: 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 **+ to create 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**.
6. On the right side of the function editor, select **View files** to open the tab.
7. Click on +Add to add new file. Name the file **function.proj**
8. In the JSON editor, provide following code to refer Nuget package System.Drawing :

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

<PropertyGroup>

<TargetFramework>netstandard2.0</TargetFramework>

</PropertyGroup>

<ItemGroup>

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

</ItemGroup>

</Project>

1. Save changes
2. Get Back in the **View files** tab create or select **function.json** file to view the editor for the function’s configuration.
3. Replace the entire contents of the JSON configuration file with the following JSON content to set up input and output binings:

{

"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)

{

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());

}

}

graphics.InterpolationMode = InterpolationMode.HighQualityBicubic;

graphics.SmoothingMode = SmoothingMode.AntiAlias;

graphics.PixelOffsetMode = PixelOffsetMode.HighQuality;

graphics.DrawImage(originalImage, 0, 0, width, height);

}

thumbnailImage.Save(outputBlob, ImageFormat.Jpeg);

}

finally

{

if (thumbnailImage != null)

{

thumbnailImage.Dispose();

}

}

}

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

#### Task 3: Validate a 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:
8. In the **Files** section, select the **Folder** icon.
9. In the File Explorer dialog box that opens, go to **\Starter\Images**, select the **veggie.jpg** file, and then select **Open**.
10. Ensure the **Overwrite if files already exist** check box is selected.
11. Select **Upload**.
12. Wait for the blob to be uploaded before you continue with this lab.
13. Close the **Container** blade.
14. Back in the **Blobs** section, select the **images-thumbnails** container.
15. In the **Container** blade, observe the newly created **veggie.jpg** file in the **images-thumbnails** container.

**Note**: It might take one to two minutes for the new image to appear. If the image is not appeared it means that configuration of function or code was not correct. You need to return back to the function and provide following text in **request body** filed of the **test** tab:

**Images/veggie.jpg**

Then click on **Run** button. Function execution should appear with blue text on the black screen log below the function. If there are red error lines the function will not work unless error been fixed.

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 **\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 \**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 **\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 all four of 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:

az group list

1. Type in the following command and press Enter to view a list of possible commands to delete a resource group:

az group delete --help

#### Task 2: Delete resource groups

1. Type the following command and press Enter to delete the **ManagedPlatform** resource group:

az group delete --name ManagedPlatform --no-wait --yes

1. 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.