

Tutorial: Upload and analyze a file with Azure Functions and Blob Storage

Article • 03/09/2023

In this tutorial, you'll learn how to upload an image to Azure Blob Storage and process it using Azure Functions and Computer Vision. You'll also learn how to implement Azure Function triggers and bindings as part of this process. Together, these services will analyze an uploaded image that contains text, extract the text out of it, and then store the text in a database row for later analysis or other purposes.

Azure Blob Storage is Microsoft's massively scalable object storage solution for the cloud. Blob Storage is designed for storing images and documents, streaming media files, managing backup and archive data, and much more. You can read more about Blob Storage on the [overview page](#).

Azure Functions is a serverless computer solution that allows you to write and run small blocks of code as highly scalable, serverless, event driven functions. You can read more about Azure Functions on the [overview page](#).

In this tutorial, you will learn how to:

- ✓ Upload images and files to Blob Storage
- ✓ Use an Azure Function event trigger to process data uploaded to Blob Storage
- ✓ Use Cognitive Services to analyze an image
- ✓ Write data to Table Storage using Azure Function output bindings

Prerequisites

- An Azure account with an active subscription. [Create an account for free](#) .
- [Visual Studio 2022](#) installed.

Create the storage account and container

The first step is to create the storage account that will hold the uploaded blob data, which in this scenario will be images that contain text. A storage account offers several different services, but this tutorial utilizes Blob Storage and Table Storage.

Azure portal

Sign in to the [Azure portal](#) .

1. In the search bar at the top of the portal, search for *Storage* and select the result labeled **Storage accounts**.
2. On the **Storage accounts** page, select **+ Create** in the top left.
3. On the **Create a storage account** page, enter the following values:
 - **Subscription:** Choose your desired subscription.
 - **Resource Group:** Select **Create new** and enter a name of `msdocs-storage-function`, and then choose **OK**.
 - **Storage account name:** Enter a value of `msdocsstoragefunction`. The Storage account name must be unique across Azure, so you may need to add numbers after the name, such as `msdocsstoragefunction123`.
 - **Region:** Select the region that is closest to you.
 - **Performance:** Choose **Standard**.
 - **Redundancy:** Leave the default value selected.

Home > Storage accounts >

Create a storage account

Basics | Advanced | Networking | Data protection | Encryption | Tags | Review + create

Azure Storage is a Microsoft-managed service providing cloud storage that is highly available, secure, durable, scalable, and redundant. Azure Storage includes Azure Blobs (objects), Azure Data Lake Storage Gen2, Azure Files, Azure Queues, and Azure Tables. The cost of your storage account depends on the usage and the options you choose below. [Learn more about Azure storage accounts](#)

Project details

Select the subscription in which to create the new storage account. Choose a new or existing resource group to organize and manage your storage account together with other resources.

Subscription *

Resource group * [Create new](#)

Instance details

If you need to create a legacy storage account type, please click [here](#).

Storage account name *

Region *

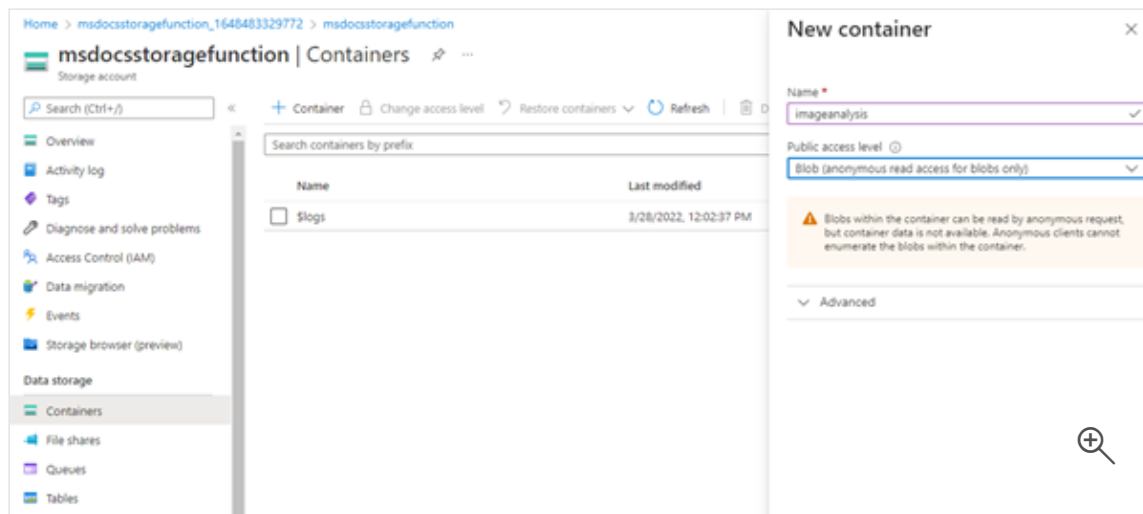
Performance * Standard: Recommended for most scenarios (general-purpose v2 account)
 Premium: Recommended for scenarios that require low latency.

Redundancy * Make read access to data available in the event of regional unavailal.

4. Select **Review + Create** at the bottom and Azure will validate the information you entered. Once the settings are validated, choose **Create** and Azure will begin provisioning the storage account, which might take a moment.

Create the container

1. After the storage account is provisioned, select **Go to Resource**. The next step is to create a storage container inside of the account to hold uploaded images for analysis.
2. On the navigation panel, choose **Containers**.
3. On the **Containers** page, select **+ Container** at the top. In the slide out panel, enter a **Name** of *imageanalysis*, and make sure the **Public access level** is set to **Blob (anonymous read access for blobs only)**. Then select **Create**.

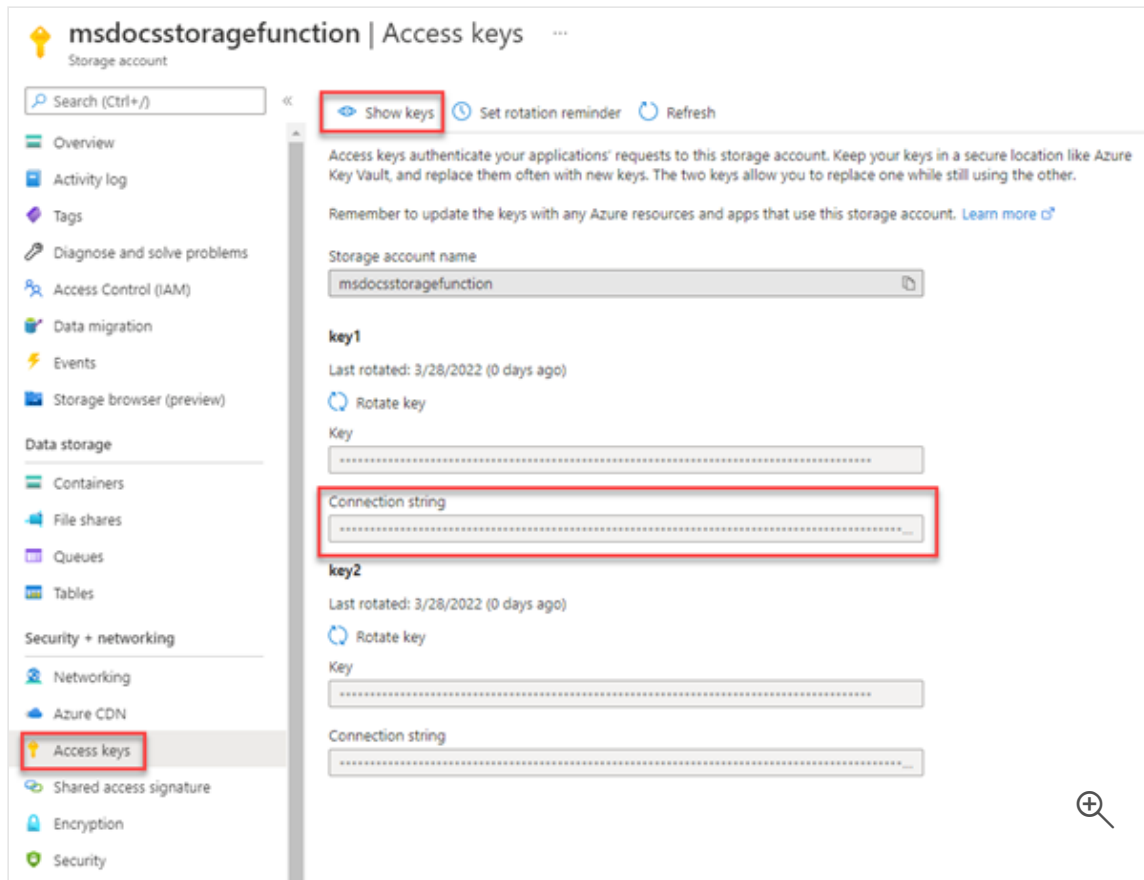


You should see your new container appear in the list of containers.

Retrieve the connection string

The last step is to retrieve our connection string for the storage account.

1. On the left navigation panel, select **Access Keys**.
2. On the **Access Keys** page, select **Show keys**. Copy the value of the **Connection String** under the **key1** section and paste this somewhere to use for later. You'll also want to make a note of the storage account name `msdocsstoragefunction` for later as well.



The screenshot displays the 'Access keys' page for the storage account 'msdocsstoragefunction'. The page includes a search bar, a 'Show keys' button (highlighted with a red box), and a 'Refresh' button. Below this, there is a section for 'key1' and 'key2'. Each key section contains a 'Rotate key' button, a 'Key' field, and a 'Connection string' field. The 'Connection string' field for 'key1' is highlighted with a red box. The left sidebar shows the 'Access keys' menu item highlighted.

These values will be necessary when we need to connect our Azure Function to this storage account.

Create the Computer Vision service

Next, create the Computer Vision service account that will process our uploaded files. Computer Vision is part of Azure Cognitive Services and offers a variety of features for extracting data out of images. You can learn more about Computer Vision on the [overview page](#).

Azure portal

1. In the search bar at the top of the portal, search for *Computer* and select the result labeled **Computer vision**.
2. On the **Computer vision** page, select **+ Create**.
3. On the **Create Computer Vision** page, enter the following values:

- **Subscription:** Choose your desired Subscription.
- **Resource Group:** Use the `msdocs-storage-function` resource group you created earlier.
- **Region:** Select the region that is closest to you.
- **Name:** Enter in a name of `msdocscomputervision`.
- **Pricing Tier:** Choose **Free** if it is available, otherwise choose **Standard S1**.
- Check the **Responsible AI Notice** box if you agree to the terms

Home > Cognitive Services >

Create Computer Vision

Basics Network Identity Tags Review + create

Boost content discoverability, accelerate text extraction, and create products that more people can use by embedding vision capabilities in your apps. Use visual data processing to label content (from objects to concepts), extract printed and handwritten text, recognize familiar subjects like brands and landmarks, and moderate content. No machine learning expertise is required.

[Learn more](#)

Project Details

Subscription *

Resource group * [Create new](#)

Instance Details

Region

Name *

Pricing tier *

[View full pricing details](#)

Responsible AI Notice

Microsoft provides technical documentation regarding the appropriate operation applicable to this Cognitive Service that is made available by Microsoft. Customer acknowledges and agrees that they have reviewed this documentation and will use this service in accordance with it. This Cognitive Services is intended to process Customer Data that includes Biometric Data (as may be further described in product documentation) that Customer may incorporate into its own systems used for personal identification or other purposes. Customer acknowledges and agrees that it is responsible for complying with the Biometric Data obligations contained in the Online Services DPA.

[Online Services DPA](#)

[Responsible Use of AI documentation for Spatial Analysis](#)

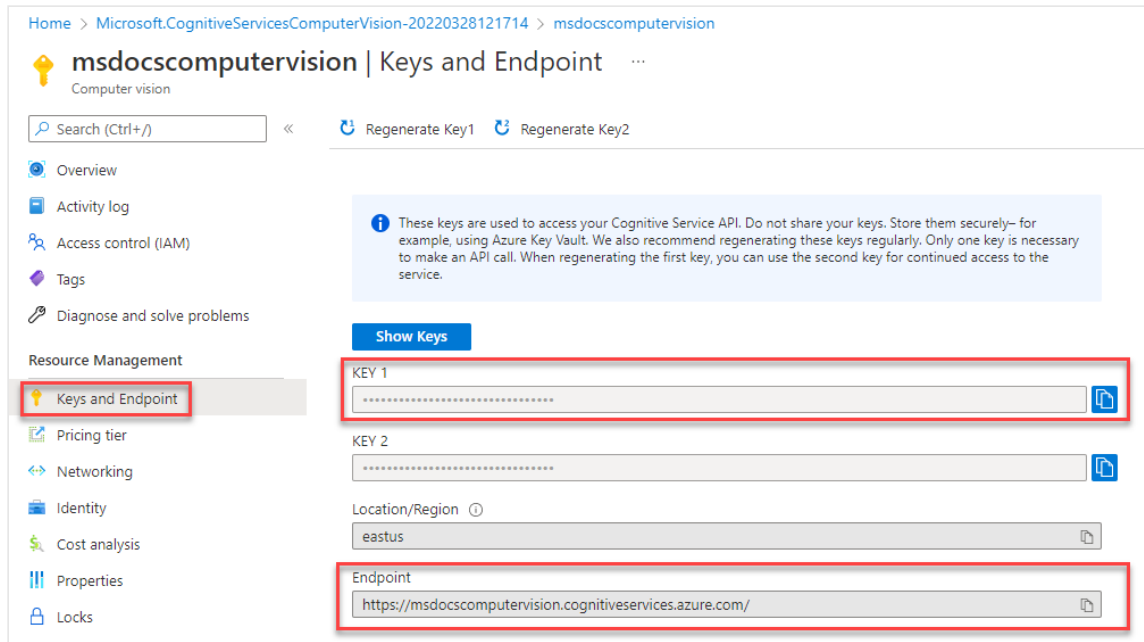
By checking this box I certify that I have reviewed and acknowledge the all the terms above.

4. Select **Review + Create** at the bottom. Azure will take a moment validate the information you entered. Once the settings are validated, choose **Create** and Azure will begin provisioning the Computer Vision service, which might take a moment.
5. When the operation has completed, select **Go to Resource**.

Retrieve the keys

Next, we need to find the secret key and endpoint URL for the Computer Vision service to use in our Azure Function app.

1. On the **Computer Vision** overview page, select **Keys and Endpoint**.
2. On the **Keys and EndPoint** page, copy the **Key 1** value and the **EndPoint** values and paste them somewhere to use for later.



The screenshot shows the Azure portal interface for a Computer Vision resource named 'msdocscomputervision'. The left-hand navigation pane is visible, with 'Keys and Endpoint' highlighted. The main content area displays the 'Keys and Endpoint' configuration page. A blue information box at the top states: 'These keys are used to access your Cognitive Service API. Do not share your keys. Store them securely—for example, using Azure Key Vault. We also recommend regenerating these keys regularly. Only one key is necessary to make an API call. When regenerating the first key, you can use the second key for continued access to the service.' Below this, there are two 'Regenerate' buttons. The 'Show Keys' button is clicked, revealing two key fields: 'KEY 1' and 'KEY 2', both containing masked values. The 'Location/Region' dropdown is set to 'eastus'. The 'Endpoint' field is highlighted with a red box and contains the URL 'https://msdocscomputervision.cognitiveservices.azure.com/'.

Download and configure the sample project

The code for the Azure Function used in this tutorial can be found in [this GitHub repository](#) . You can also clone the project using the command below.

terminal

```
git clone https://github.com/Azure-Samples/msdocs-storage-bind-function-service.git \  
cd msdocs-storage-bind-function-service/dotnet
```

The sample project code accomplishes the following tasks:

- Retrieves environment variables to connect to the storage account and Computer Vision service

- Accepts the uploaded file as a blob parameter
- Analyzes the blob using the Computer Vision service
- Sends the analyzed image text to a new table row using output bindings

Once you have downloaded and opened the project, there are a few essential concepts to understand in the main `Run` method shown below. The Azure function utilizes Trigger and Output bindings, which are applied using attributes on the `Run` method signature.

The `Table` attribute uses two parameters. The first parameter specifies the name of the table to write the parsed image text value returned by the function. The second `Connection` parameter pulls a Table Storage connection string from the environment variables so that our Azure function has access to it.

The `BlobTrigger` attribute is used to bind our function to the upload event in Blob Storage, and supplies that uploaded blob to the `Run` function. The blob trigger has two parameters of its own - one for the name of the blob container to monitor for uploads, and one for the connection string of our storage account again.

```
C#

// Azure Function name and output Binding to Table Storage
[FunctionName("ProcessImageUpload")]
[return: Table("ImageText", Connection = "StorageConnection")]
// Trigger binding runs when an image is uploaded to the blob container below
public async Task<ImageContent> Run([BlobTrigger("imageanalysis/{name}",
    Connection = "StorageConnection")]Stream myBlob, string name, ILogger
log)
{
    // Get connection configurations
    string subscriptionKey =
Environment.GetEnvironmentVariable("ComputerVisionKey");
    string endpoint =
Environment.GetEnvironmentVariable("ComputerVisionEndpoint");
    string imgUrl = $"https://{
Environment.GetEnvironmentVariable("StorageAccountName")}
.blob.core.windows.net/imageanalysis/{name}";

    ComputerVisionClient client = new ComputerVisionClient(
        new ApiKeyServiceClientCredentials(subscriptionKey)) { Endpoint = end-
point };

    // Get the analyzed image contents
    var textContext = await AnalyzeImageContent(client, imgUrl);

    return new ImageContent {
        PartitionKey = "Images",
```

```
        RowKey = Guid.NewGuid().ToString(), Text = textContext
    };
}

public class ImageContent
{
    public string PartitionKey { get; set; }
    public string RowKey { get; set; }
    public string Text { get; set; }
}
```

This code also retrieves essential configuration values from environment variables, such as the storage account connection string and Computer Vision key. We'll add these environment variables to our Azure Function environment after it's deployed.

The `ProcessImage` function also utilizes a second method called `AnalyzeImage`, seen below. This code uses the URL Endpoint and Key of our Computer Vision account to make a request to that server to process our image. The request will return all of the text discovered in the image, which will then be written to Table Storage using the output binding on the `Run` method.

```
C#

static async Task<string> ReadFileUrl(ComputerVisionClient client, string url-
File)
{
    // Analyze the file using Computer Vision Client
    var textHeaders = await client.ReadAsync(urlFile);
    string operationLocation = textHeaders.OperationLocation;
    Thread.Sleep(2000);

    // Complete code omitted for brevity, view in sample project

    return text.ToString();
}
```

Running locally

If you'd like to run the project locally, you can populate the environment variables using the `local.settings.json` file. Inside of this file, fill in the placeholder values with the values you saved earlier when creating the Azure resources.

Although the Azure Function code will run locally, it will still connect to the live services out on Azure, rather than using any local emulators.

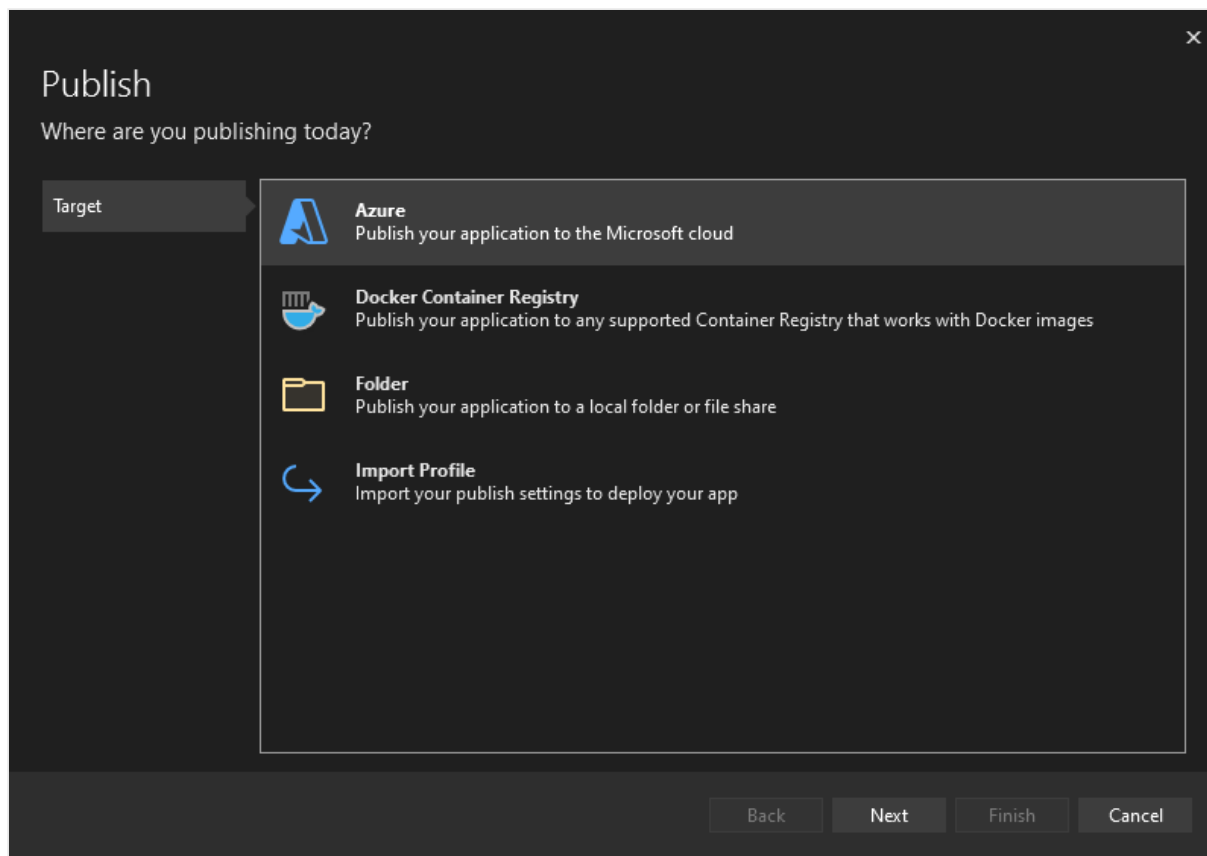
JavaScript

```
{
  "IsEncrypted": false,
  "Values": {
    "AzureWebJobsStorage": "UseDevelopmentStorage=true",
    "FUNCTIONS_WORKER_RUNTIME": "dotnet",
    "StorageConnection": "your-storage-account-connection-string",
    "StorageAccountName": "your-storage-account-name",
    "ComputerVisionKey": "your-computer-vision-key",
    "ComputerVisionEndPoint": "your-computer-vision-endpoint"
  }
}
```

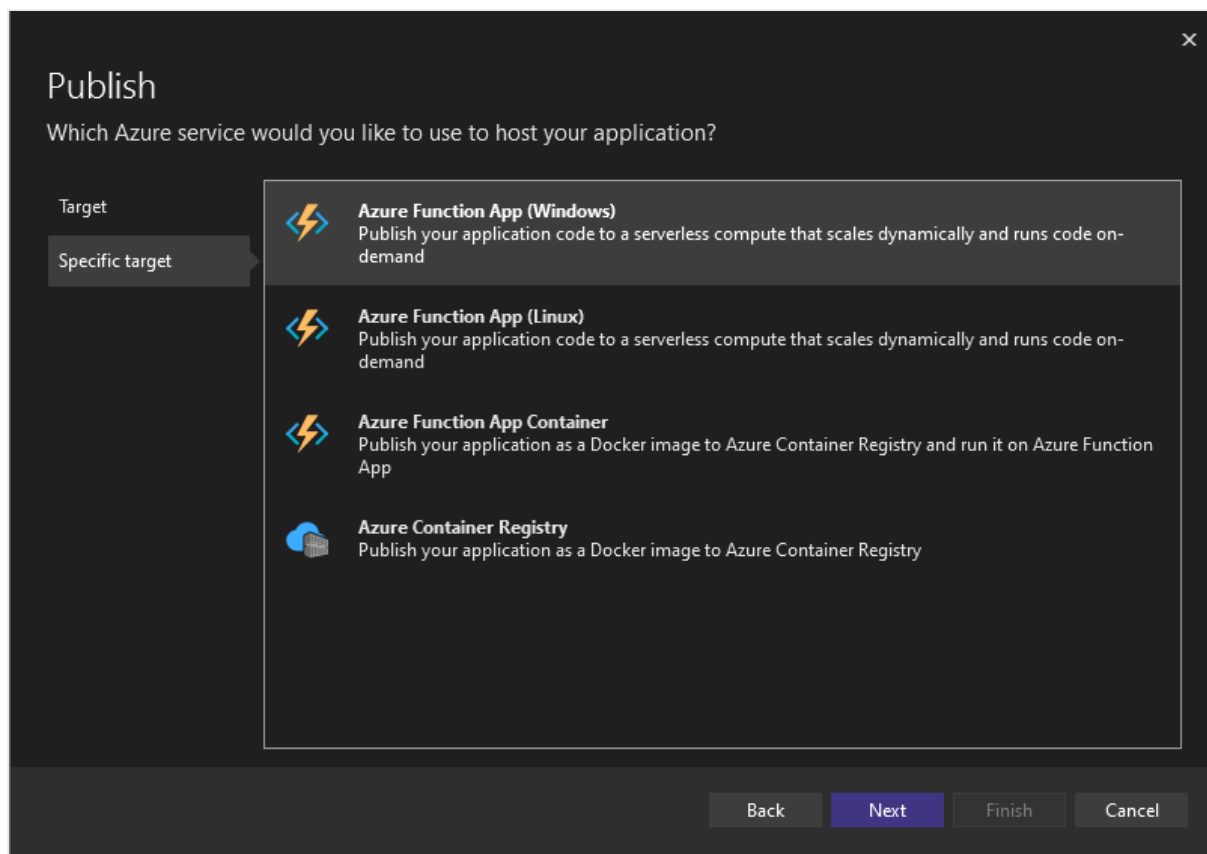
Deploy the code to Azure Functions

You are now ready to deploy our application to Azure by using Visual Studio. You can also create the Azure Functions app in Azure at the same time as part of the deployment process.

1. To begin, right select the **ProcessImage** project node and select **Publish**.
2. On the **Publish** dialog screen, select Azure and choose **Next**.



3. Select **Azure Function App (Windows)** or **Azure Function App (Linux)** on the next screen, and then choose **Next** again.



4. On the **Functions instance** step, make sure to choose the subscription you'd like to deploy to. Next, select the green + symbol on the right side of the dialog.
5. A new dialog will open. Enter the following values for your new Function App.
 - **Name:** Enter *msdocsprocessimage* or something similar.
 - **Subscription Name:** Choose whatever subscription you'd like to use.
 - **Resource Group:** Choose the *msdocs-storage-function* resource group you created earlier.
 - **Plan Type:** Select **Consumption**.
 - **Location:** Choose the region closest to you.
 - **Azure Storage:** Select the storage account you created earlier.

Function App (Windows) Create new

Microsoft

Name
ProcessImage

Subscription name
C&L Cross Service Content Team Testing

Resource group
msdocs-storage-function (East US) New...

Plan Type
Consumption

Location
Central US

Azure Storage
msdocsstoragefunction (East US) New...

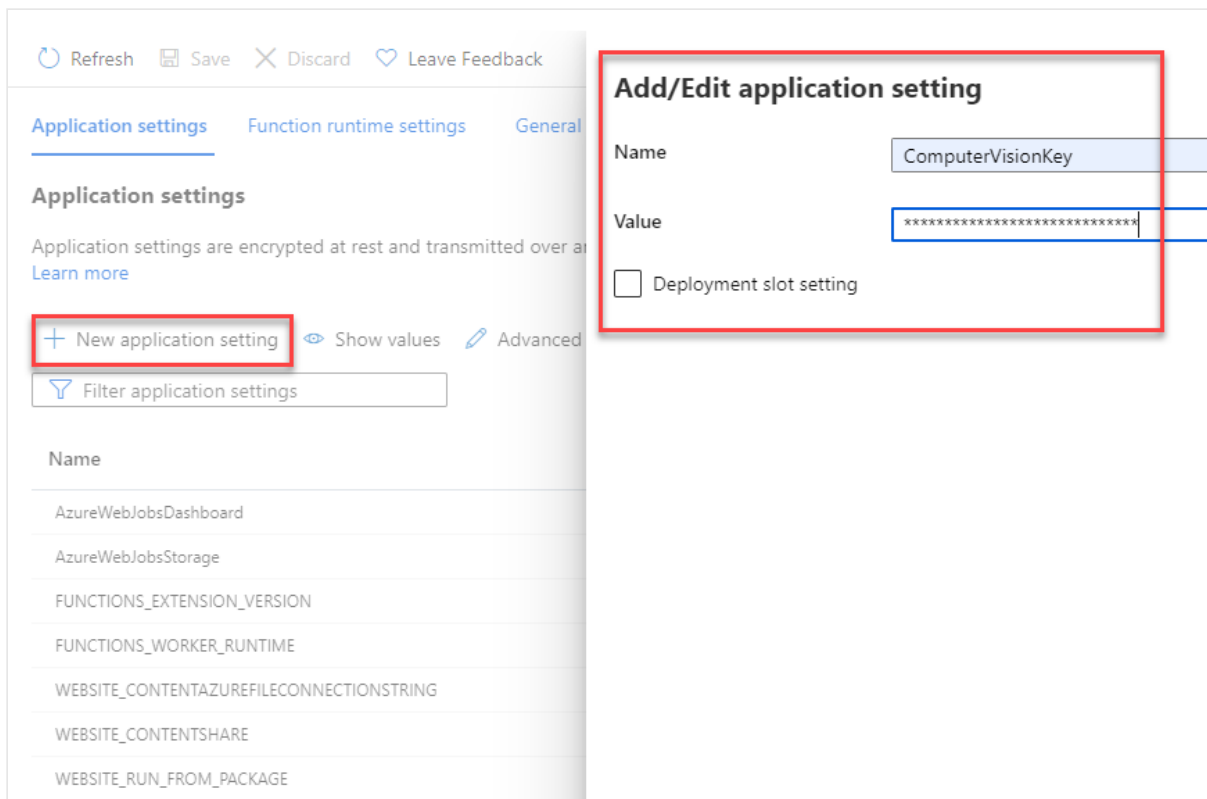
Export... Create Cancel

6. Once you have filled in all of those values, select **Create**. Visual Studio and Azure will begin provisioning the requested resources, which will take a few moments to complete.
7. Once the process has finished, select **Finish** to close out the dialog workflow.
8. The final step to deploy the Azure Function is to select **Publish** in the upper right of the screen. Publishing the function might also take a few moments to complete. Once it finishes, your application will be running on Azure.

Connect the services

The Azure Function was deployed successfully, but it cannot connect to our storage account and Computer Vision services yet. The correct keys and connection strings must first be added to the configuration settings of the Azure Functions app.

1. At the top of the Azure portal, search for *function* and select **Function App** from the results.
2. On the **Function App** screen, select the Function App you created in Visual Studio.
3. On the **Function App** overview page, select **Configuration** on the left navigation. This will open up a page where we can manage various types of configuration settings for our app. For now, we are interested in **Application Settings** section.
4. The next step is to add settings for our storage account name and connection string, the Computer Vision secret key, and the Computer Vision endpoint.
5. On the **Application settings** tab, select + **New application setting**. In the flyout that appears, enter the following values:
 - **Name:** Enter a value of *ComputerVisionKey*.
 - **Value:** Paste in the Computer Vision key you saved from earlier.
6. Click **OK** to add this setting to your app.



7. Next, let's repeat this process for the endpoint of our Computer Vision service, using the following values:

- **Name:** Enter a value of *ComputerVisionEndpoint*.
- **Value:** Paste in the endpoint URL you saved from earlier.

8. Repeat this step again for the storage account connection, using the following values:

- **Name:** Enter a value of *StorageConnection*.
- **Value:** Paste in the connection string you saved from earlier.

9. Finally, repeat this process one more time for the storage account name, using the following values:

- **Name:** Enter a value of *StorageAccountName*.
- **Value:** Enter in the name of the storage account you created.

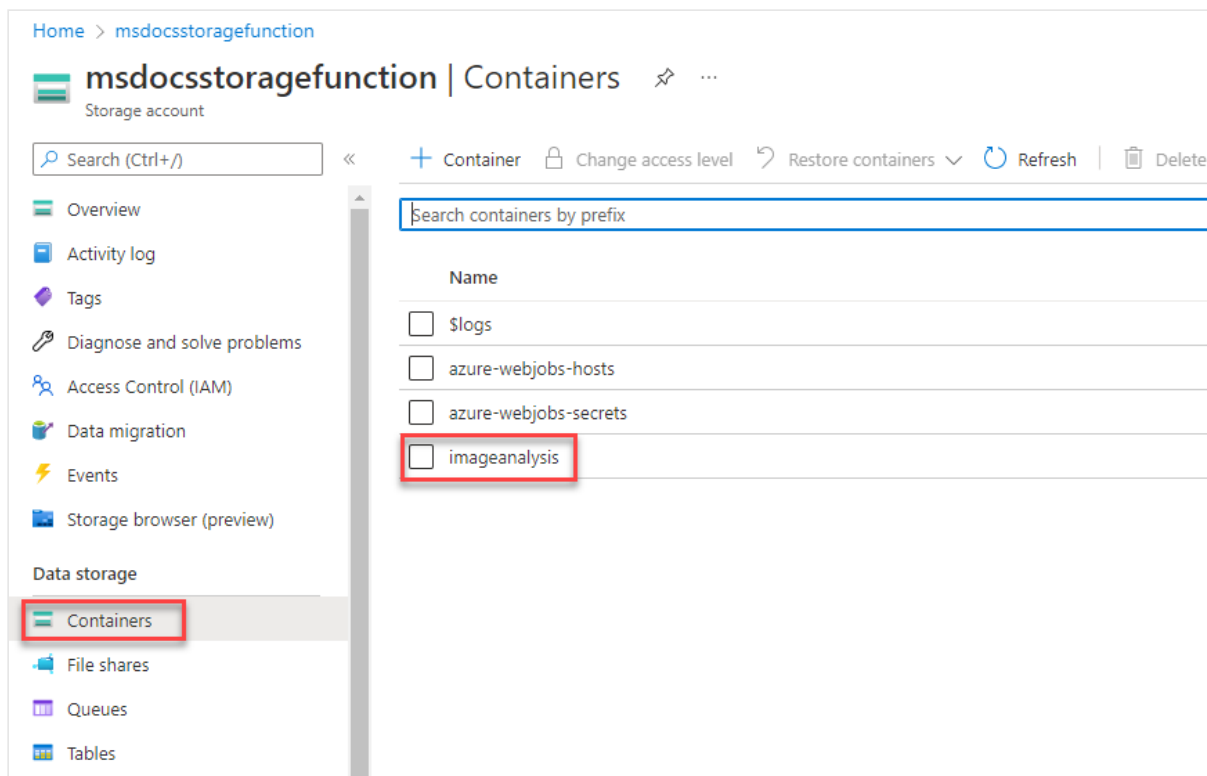
10. After you have added these application settings, make sure to select **Save** at the top of the configuration page. When the save completes, you can hit **Refresh** as well to make sure the settings are picked up.

All of the required environment variables to connect our Azure function to different services are now in place.

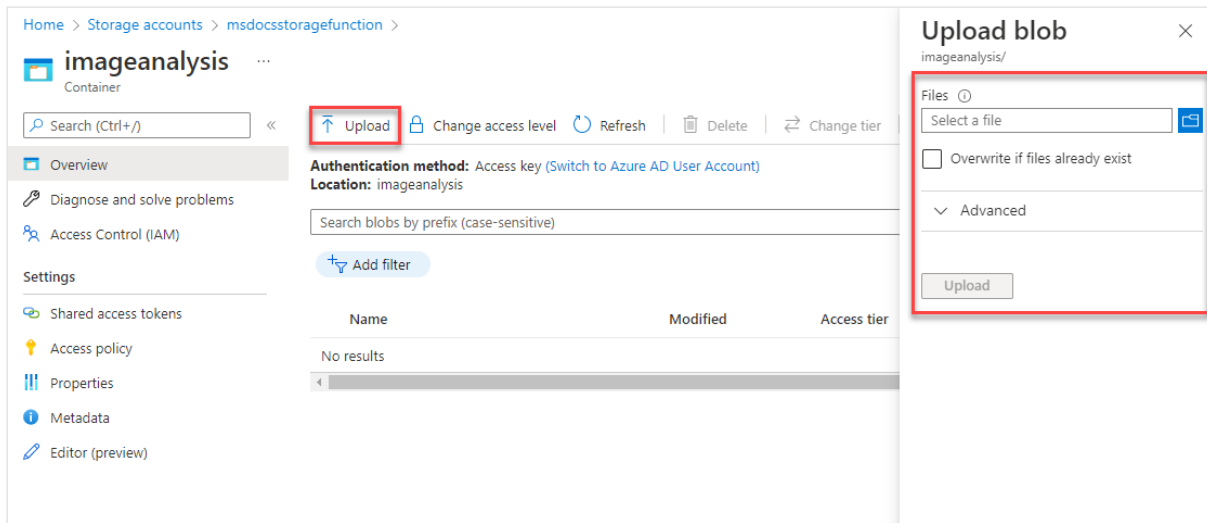
Upload an image to Blob Storage

You are now ready to test out our application! You can upload a blob to the container, and then verify that the text in the image was saved to Table Storage.

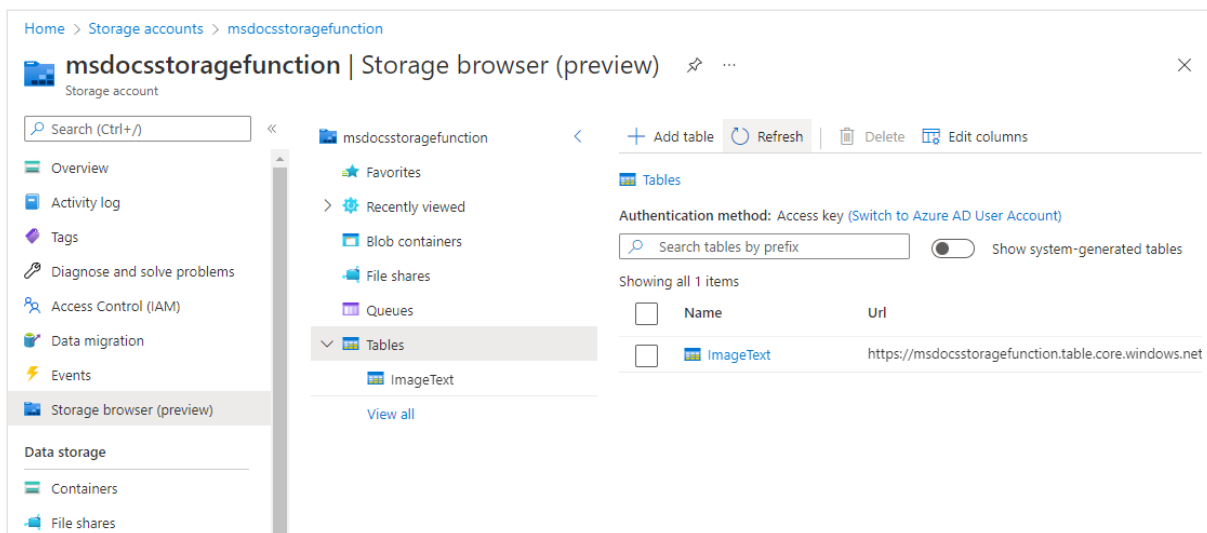
1. First, at the top of the Azure portal, search for *Storage* and select **storage account**. On the **storage account** page, select the account you created earlier.
2. Next, select **Containers** on the left nav, and then navigate into the **ImageAnalysis** container you created earlier. From here you can upload a test image right inside the browser.



3. You can find a few sample images included in the **images** folder at the root of the downloadable sample project, or you can use one of your own.
4. At the top of the **ImageAnalysis** page, select **Upload**. In the flyout that opens, select the folder icon on the right to open up a file browser. Choose the image you'd like to upload, and then select **Upload**.



5. The file should appear inside of your blob container. Next, you can verify that the upload triggered the Azure Function, and that the text in the image was analyzed and saved to Table Storage properly.
6. Using the breadcrumbs at the top of the page, navigate up one level in your storage account. Locate and select **Storage browser** on the left nav, and then select **Tables**.
7. An **ImageText** table should now be available. Click on the table to preview the data rows inside of it. You should see an entry for the processed image text of our upload. You can verify this using either the Timestamp, or by viewing the content of the **Text** column.



Congratulations! You succeeded in processing an image that was uploaded to Blob Storage using Azure Functions and Computer Vision.

Clean up resources

If you're not going to continue to use this application, you can delete the resources you created by removing the resource group.

1. Select **Resource groups** from the main navigation
2. Select the `msdocs-storage-function` resource group from the list.
3. Select the **Delete resource group** button at the top of the resource group overview page.
4. Enter the resource group name `msdocs-storage-function` in the confirmation dialog.
5. Select delete. The process to delete the resource group may take a few minutes to complete.